Learning to Rerank Schema Matches

Avigdor Gal, Senior Member, IEEE, Haggai Roitman, and Roe Shraga, Student Member, IEEE

Abstract—Schema matching is at the heart of integrating structured and semi-structured data with applications in data warehousing, data analysis recommendations, Web table matching, etc. Schema matching is known as an uncertain process and a common method to overcome this uncertainty introduces a human expert with a ranked list of possible schema matches to choose from, known as top-K matching. In this work we propose a learning algorithm that utilizes an innovative set of features to rerank a list of schema matches and improves upon the ranking of the best match. We provide a bound on the size of an initial match list, tying the number of matches with a desired level of confidence in finding the best match. We also propose the use of matching predictors as features in a learning task, and tailored nine new matching predictors for this purpose. The proposed algorithm assists the matching process by introducing a quality set of alternative matches to a human expert. It also serves as a step towards eliminating the involvement of human experts as decision makers in a matching process altogether. A large scale empirical evaluation with real-world benchmark shows the effectiveness of the proposed algorithmic solution.

Index Terms—Schema Matching, Heterogeneous Data Integration, Uncertainty, Learning to Rerank

1 INTRODUCTION

Schema matching [1] is the task of providing correspondences between concepts describing the meaning of data in various heterogeneous, distributed data sources. Schema matching is one of the basic operations required by the process of data integration, and thus has great effect on its outcomes. Schema matching research has been going on for more than 30 years now, focusing on the identification of high quality matchers, automatic tools for identifying correspondences among database attributes. Initial heuristic attempts (e.g., COMA [4]) were followed by theoretical grounding (e.g., [1], [9]). Recently, the use of matching predictors to assess the quality of matchers in the absence of a reference match was proposed [32] and implemented in tools for dynamic ensemble weight setting and process matching [38].

Data integration in general, and matching in particular, were recognized in the literature to be uncertain processes [5], [9] due to enormous ambiguity and heterogeneity of data description concepts. A matcher may be required to consider many probable correspondences, and therefore, its choice may be wrong.

For these reasons, researchers proposed the use of a ranked list of the “best” K matches a matcher can generate (top-K) [28], creating a search space in uncertain settings and assigning probabilities in probabilistic schema matching [31].

Gal et al. provide a theoretical grounding to the use of top-K matches in the form of the monotonicity principle [10]. A monotonic matcher ranks matches according to a similarity measure whose behavior is closely related to a measure of choice (e.g., precision). Therefore, monotonicity ensures that matches in the top-K list are sufficiently close to a reference match.

Top-K match lists are currently used for human matchers, who choose the best match from a list of (hopefully) powerful set of matches. In this work we investigate an alternative, in which top-K match list serves as a starting point for a rerank learning system, pushing better matches towards the top of the list. The implications of automatically reranking a match list are far reaching. Practically, reranking allows an automatic (rather than human-based) selection of a best candidate for a match in a setting where human experts are still valued as final decision makers and validators of any automatic match result. It is therefore a significant step forward into fully automating the matching process.

We tackle the challenge of identifying the best match within a top-K list using machine learning and propose a novel reranking approach, applied as a post-processing step of top-K matching. Our contribution is threefold:

- We provide a bound on the size of a top-K match list, tying the number of matches with a desired level of confidence in finding the best match (Section 2.3).
- Using matching predictors as innovative features, we rerank the best match in a top-K list higher (Section 3).
- Using large scale experiments with real-world benchmark ontology and schema sets, as well as synthetic data, we show the effectiveness of the proposed algorithmic solution for reranking top-K matches (Section 4).

This paper expands on our earlier work [12], providing a new set of matching predictors (Section 3.1) to extend our proposed algorithm, a bound on the size of a top-K match list (Section 2.3), a validation for the use of NDCG as the optimization function (Section 4.4.2), and an extension of the empirical evaluation to include ranking results and a robustness analysis. Building blocks of our model are given in Section 2 and related work in matching prediction and top-K schema matching is given in Section 5. We conclude in Section 6.
2 Model

We next provide the basic building blocks of the learn-to-rerank model (matching, top-K matches, and schema matching predictors) and a bound on the value of $K$.

2.1 Matching Model

Let $S, S'$ be two schemata with attributes $\{a_1, a_2, \ldots, a_n\}$ and $\{b_1, b_2, \ldots, b_m\}$, respectively. A matching process matches schemata by aligning their attributes using matching algorithms (matchers for short), which deduce similarity using schema characteristics, e.g., attribute labels. A matcher’s output can be conceptualized as a similarity matrix, hereinafter $M(S, S')$ (or simply $M$), having entry $M_{i,j}$ (typically a real number in $[0, 1]$) represents a degree of similarity between $a_i \in S$ and $b_j \in S'$. If all $M_{i,j}$ are 0 or 1 then $M$ is binary.

For any matched pair $(S, S')$, a match, denoted $\sigma$, is a subset of $M$’s entries and $\Sigma \subseteq 2^S \times 2^{S'}$ is the set of all possible matches. Matches (those in $\Sigma$) are subject to application semantics using constrains such as 1:1 matching. Also, while the similarity matrix abstraction “flattens” schema descriptions, it still captures rich structures (including taxonomies, ontologies, and others [9]).

Example 1. Table 1 provides an example of a similarity matrix over two simplified purchase order schemata [4], one with four attributes: poDay and poTime, representing a timestamp of the order, poCode represents a purchase order number, and the shipment city (city). The other schema has three attributes, namely orderDate issuing date of the order, orderNumber for the order number, and city for shipment city.

![Table 1: Similarity Matrix Example](image)

Let $f(\sigma, M)$ denote a schema pair similarity function, assigning an overall value to a match $\sigma$ given individual similarity values of all matrix $M$ entries. Typically, matching similarity functions are monotone submodular, e.g.,

$$f(\sigma, M) = \sum_{(i,j) \in \sigma} M_{i,j}$$

(1)

Whenever the similarity matrix $M$ is clear from the context, we shall refer to $f(\sigma, M)$ simply as $f(\sigma)$.

The similarity matrix of Example 1 is the outcome of Term [9], a string-based matcher. A Maximum Weight Bipartite Graph algorithm [13] was applied to generate a match, $\sigma = \langle \text{orderDate, poDay}, \text{orderNumber, poCode}, \text{city, city} \rangle$, marked by the boldface matrix entries, with $f(\sigma) = 1.31$, using Eq. 1.

Let $M^+$ be a binary matrix, which represents a reference match such that $M_{i,j} = 1$ whenever the correspondence $(a_i, b_j)$ is part of the reference match pair $(S, S')$ and $M_{i,j} = 0$ otherwise. We define $G_{M^+} : \Sigma \rightarrow [0, 1]$ to be a relevance function, assigning scores to matches according to their ability to identify correspondences in the reference match. We next define the precision (P) and recall (R) relevance functions, as follows:

$$P_{M^+}(\sigma) = \frac{|\sigma \cap M^+|}{|\sigma|}, R_{M^+}(\sigma) = \frac{|\sigma \cap M^+|}{|M^+|}$$

(2)

where $M^+$ represent the non-zero entries of $M^+$ and recalling that $\sigma$ is a subset of $M^+$’s entries. Using precision and recall, we further define the F1 measure, $F_{M^+}(\sigma)$, calculated as the harmonic mean of $P_{M^+}(\sigma)$ and $R_{M^+}(\sigma)$. Whenever the reference match is clear from the context, we shall refer to $P_{M^+}(\sigma), R_{M^+}(\sigma)$, and $F_{M^+}(\sigma)$ simply as $P(\sigma), R(\sigma)$, and $F(\sigma)$, respectively. Henceforth, we shall use $F(\sigma)$ as our relevance function of choice.

Continuing with Example 1, let the reference match be $\langle \text{orderDate, poDay}, \text{orderNumber, poCode}, \text{city, city} \rangle$. Then, we have that $P(\sigma) = 0.67, R(\sigma) = 0.5$, and $F(\sigma) = 0.57$.

2.2 Top-K Matches

Top-K matches are the best $K$ matches a matcher can generate in $\Sigma$ (according to $f(\sigma)$) [9], provided as a ranked list $\sigma^K = \langle \sigma_1, \sigma_2, \ldots, \sigma_K \rangle$. Continuing with Example 1, $\sigma_2 = \langle \text{orderDate, poDay}, \text{orderNumber, poCode}, \text{city, city} \rangle$, with $f(\sigma_2) = 1.31$ (the same as for best match computed above), $P(\sigma_2) = 1.00, R(\sigma_2) = 0.75$, and $F(\sigma_2) = 0.86.$ Note that here, the second-best match is better, in terms of $P, R$, and $F$, than the top match. Algorithms for computing top-$K$ matches can be found in [9][Ch. 5].

Historically, the ranked list of matches was evaluated by a human expert to identify the best match out of the list. Gal et al. [10] introduced the notion of a monotonic matcher, a matcher that ranks matches in an approximate ordering of their quality. The monotonicity principle, which was also introduced there, suggests that a top-$K$ list, if created by a monotonic matcher, contains a set of matches of high quality yet their ordering within the list may not truly represent their relative quality. Therefore, a match at the top of the list may be of lesser quality than another, further down the list, as the example above illustrates.

To put this understanding in formal terms, we assign a match $\sigma_i (1 \leq i \leq K)$ in $\Sigma^K = \langle \sigma_1, \sigma_2, \ldots, \sigma_K \rangle$ with a relevance value, $F(\sigma_i)$. We can transform relevance values into a ranked list so that the match with the highest $F(\sigma_i)$ is assigned a value of 1 and so forth, aided by some tie-breaking mechanism. We term this ranking an ideal ranking of $\sigma^K$ and denote it by $\sigma^K(1), \sigma^K(2), \ldots, \sigma^K(K)$). Then, for each $\sigma_i$ in the list, we denote its rank in the ideal ranking by $\text{Rank}(\sigma_i)$, $\sigma^K$ and $\sigma^K(1)$ can be jointly used to provide an overall quality measure of the ranking of the top-$K$ match list.

To capture correctly the monotonicity nature of a matcher, a Naive evaluation approach, assigning a value of 1 for recognizing the best match as top ranked ($\sigma_1 = \sigma_i$) and 0 otherwise, does not suffice. We introduce two measures, namely Kendall’s Tau ($\tau$ [18]) and Normalized Discounted Cumulative Gain (NDCG [17]), both take into account the list ordering in evaluation. The NDCG is a non-binary evaluation measure that penalizes relevant items that appear
low in a ranking while the $\tau$ measure assigns equal penalty to ranking errors, whether found at the top or the bottom of the list [40].

Let $C(\sigma^K, \sigma^{(K)})$ represent the number of match pairs in $\sigma^K$ that are ranked in the same order in $\sigma^{(K)}$ and let $D(\sigma^K, \sigma^{(K)})$ be the cardinality of the complementary set of pairs, those that are inversely ranked in $\sigma^K$ and in $\sigma^{(K)}$ ($D(\sigma^K, \sigma^{(K)}) = K(K-1)/2 - C(\sigma^K, \sigma^{(K)})$). Kendall’s $\tau$ computes the correlation between $\sigma^K$ and $\sigma^{(K)}$, as follows:

$$\tau(\sigma^K, \sigma^{(K)}) = \frac{C(\sigma^K, \sigma^{(K)}) - D(\sigma^K, \sigma^{(K)})}{K(K-1)/2} \quad (3)$$

NDCG is given as the division of a discounted cumulative gain (DCG) by the ideal DCG:

$$\text{NDCG}(\sigma^K, \sigma^{(K)}) = \left( \sum_{i=1}^{k} \frac{2^F(\sigma_i) - 1}{\ln_2(i+1)} \right) / \left( \sum_{i=1}^{\infty} \frac{2^F(\sigma_i) - 1}{\ln_2(i+1)} \right) \quad (4)$$

### 2.3 On Choosing an Optimal $K$

There are two conflicting requirements when determining the size of a match list. On the one hand, a larger $K$ provides a higher chance for the list to contain a good match, even for monotonic matchers, which are expected to assign high scores to good matches. On the other hand, a larger list may be challenging to a human judge when sifting through it; whereas for a machine learning algorithm it increases the risk of erroneous decision making and requires more time and computation resources [20].

We denote by $F^* = \max_{\sigma \in \Sigma} F(\sigma)$ the maximum value of $F_1$ over all possible matches; and further denote by $U(\sigma^K) = \max_{\sigma \in \sigma^K} F(\sigma)$ the highest $F_1$ in a given top-$K$ match list.

Given the inherent uncertainty of the matching problem, the best top-$K$ list for our purposes is a list that contains the best match possible, the one whose $F_1$ is maximum. Note that while we expect to find the reference match as the match with maximum $F_1$ ($F^* = 1$), the reference match may not be present in $\Sigma$; simply because it does not satisfy the constraints as set by the matching environment.

Given $0 < p < 1$, a user-defined confidence level, we seek a $K$ such that $Pr\{U(\sigma^K) = F^*\} \geq p$. That is, a top-$K$ match list that is large enough to contain, with a probability $p$, the match with the highest $F_1$. The monotonicity principle dictates a diminishing probability to find a match $\sigma$, such that $F(\sigma) = F^*$, low in a top-$K$ match list. Therefore, for monotonic matchers, ranking such a match lower is still possible but with low probability. In what follows, we define $K$ to be a random variable that represents the size of $\sigma^K$ that contains a match $\sigma$ such that $F(\sigma) = F^*$ while $\sigma \notin \sigma^{K-1}$. We assume $K$ is geometrically distributed with some parameter $\theta$. The $\theta$ parameter is the probability of having the best match as the one with the highest $F_1$, i.e., $F(\sigma_1) = F^*$.

Geometric distribution assumes an infinite number of possible matches. While this assumption does not hold, the residual probability mass for high number of matches is sufficiently small for matchers to approximate a geometric behavior quite closely, as shown empirically in Section 4.3.

**Theorem 1.** Let $\sigma^K$ be a top-$K$ match list. For $K = \left[ \frac{\ln(1-p)}{\ln(1-\theta)} \right]$, with a user-defined parameter $0 < p < 1$, $Pr\{U(\sigma^K) = F^*\} \geq p$.

**Proof 1.** Let $K \sim \text{Geo}(\theta)$ be a random variable, representing the size of $\sigma^K$ that contains a match $\sigma$ s.t. $F(\sigma) = F^*$ while $\sigma \notin \sigma^{K-1}$. Let $k_p \geq 1$ be the minimum value of $K$ at which the probability that the top-$K$ match list contains $\sigma^*$ (a match such that $F(\sigma^*) = F^*$) is $p$, $k_p$ probabilistically guarantees (with probability $p$) that the match list contains a match $\sigma^*$.

$p = Pr\{K = k_p\}$ is the probability that the top-$K$ match list contains $\sigma^*$ while $\sigma^*$ was not present in $\sigma^K \forall k : 1 \leq k < K$. We are, in fact, interested in the probability that $K$ is less than (or exactly) $k_p$, i.e., we have seen $\sigma^*$ in the top-$K$ match list size smaller than (or equals) $k_p$:

$$Pr\{K \leq k_p\} = p \quad (5)$$

From the definition of $K$ and Eq. 5,

$$Pr\{U(\sigma^{k_p}) = F^*\} = Pr\{K \leq k_p\} = p \quad (6)$$

$K$ is geometrically distributed, and therefore

$$p = Pr\{K \leq k_p\} = 1 - Pr\{K > k_p\} = 1 - \sum_{k=k_p+1}^{\infty} \theta(1-\theta)^{k-1} \quad (7)$$

By sum of a geometric series one has that

$$1 - \frac{\theta(1-\theta)^{k_p}}{1-(1-\theta)} = p \quad (8)$$

After some re-writing of Eq. 8 and taking $\ln(\cdot)$ on both sides, we obtain:

$$k_p = \frac{\ln(1-p)}{\ln(1-\theta)} \quad (9)$$

The top-match list size is an integer, hence define $K$ to be:

$$K = \left[ k_p \right] = \left[ \frac{\ln(1-p)}{\ln(1-\theta)} \right] \quad (10)$$

Combining Eqs. 6 and 10 one has that

$$Pr\{U(\sigma^K) = F^*\} = Pr\{K \leq k_p\} = Pr\{K \leq \left[ \frac{\ln(1-p)}{\ln(1-\theta)} \right] \} \geq Pr\left\{K \leq \frac{\ln(1-p)}{\ln(1-\theta)} \right\}$$

$$= Pr\{K \leq k_p\} = p$$

which concludes the proof. \hfill \square

Theorem 1 provides a theoretical bound on the size of a top-$K$ match list. For example, given $K \sim \text{Geo}(0.38)$, to ensure a top-$K$ match list that consists the best match, with probability of at least 0.99, one needs to obtain a match list of size: $K = \left\lceil \frac{\ln(1-p)}{\ln(1-\theta)} \right\rceil = \left\lceil \frac{\ln(1-0.99)}{\ln(1-0.38)} \right\rceil \approx [9.63] = 10$. 


2.4 Matching Predictors

Matching predictors (predictors for short, not to confuse with a prediction function notation) assess the quality of a match in the absence of a reference match [32]. Formally, a predictor is a function that maps a similarity matrix into a real prediction value.

Predictors predict different qualities, such as precision or recall. In general, there are two types of predictors, idealizers and internalizers [32]. Idealizers assume there is some principle that dictates high quality results and the prediction is based on the distance between the supplied result and an ideal result adhering to this principle. For example, the Binary Matrix Converter (BMC) predictor [32] is based on the assumption that non-binary similarity matrices are binary matrices masked by random noise. With that said, this predictor prefers matrices with entries close to 0 or 1. Internalizers are based on internal properties of the similarity matrix. For example, Max, STDEV, and Avg calculate the respective measure they are named after for each row in the matrix and average the values over the number of the rows in the matrix [32].

3 Learning To Rerank Schema Matches

Given a top-\( K \) match list that contains, with high probability, a close-to-reference match, we are now left with the problem of identifying which match in the list has the highest F1 value. We advocate an algorithmic solution for reranking a top-\( K \) match list, aiming at positioning at the top of the list the match with the highest F1. We propose a learn-to-rerank algorithm for schema matching, which utilizes novel matching predictors as its feature set for learning.

Learn-to-Rank (LTR) [19] algorithms are used in Information Retrieval to optimally combine features, extracted from query-document pairs through discriminative training, to rerank a retrieved set of documents. LTR is a framework with three basic elements, the choice of which is critical, namely features, a method, and an optimization function. We present an innovative use of feature vectors to represent successful matches in an LTR algorithm. Section 3.1 presents a new set of features using matching predictors (Section 2.4). In Section 3.2 we detail the last two elements in the context of schema matching and present a complete learn-to-rerank algorithm.

3.1 Matching Predictors as Features

To construct feature vectors for the learn-to-rerank task we propose to use predictors over similarity matrices. Matrix-based predictors predict the usefulness of an entire similarity matrix for a matching task. As a baseline we suggest the use of state-of-the-art predictors [32], [11], which mostly emphasize positive characteristics of a match. In addition, we next propose three novel types of predictors that capture complementary negative aspects.

Most of the matching predictors in the literature manipulate similarity matrices. Recall that a match is defined as a subset of a similarity matrix entries. We offer a simple transformation of a match into a similarity matrix, as follows. Let \( M \) be a similarity matrix over a schema pair \( (S, S') \) and let \( \sigma \in \Sigma \) be a match between \( S \) and \( S' \). A non-binary transformation of \( \sigma \) is a matrix \( M' \) such that

\[
M'_{i,j} = \begin{cases} 
M_{i,j} & \text{if } (i, j) \in \sigma \\
0 & \text{otherwise.} 
\end{cases}
\]  

(11)

The non-binary transformation of the best match of Example 1 is:

\[
M' = \begin{pmatrix}
0.22 & 0.00 & 0.00 & 0.00 \\
0.00 & 0.09 & 0.00 & 0.00 \\
0.00 & 0.00 & 0.00 & 1.00 \\
\end{pmatrix}
\]

3.1.1 PCA Predictors

Principal Component Analysis (PCA) is a dimension reduction method, aiming at preserving as much variance as possible [37]. PCA is well-suited for matching since matchers tend to focus on a specific semantic cue of the matching problem (e.g., attribute labels) while ignoring other aspects. Constructing a PCA model over such data provides a compact representation that ignores missing semantic cues while maintaining the main variability of the match. This could potentially differentiate correct correspondences from background noise [11].

PCA typically uses eigen-decomposition of a matrix, which is limited to square matrices. For non-square similarity matrices we propose to use singular value decomposition (SVD) [37].

We propose three new matching predictors, namely First Principal Component predictor (1PC), Second Principal Component predictor (2PC), and Principal Component Analysis predictor (PCA). The first two simply use the first and the second principal components as predictors. The third is a sum of informative components.

Informative components are chosen as the leading principal components that balance the dimension reduction with keeping as much variance as possible. We use Horn’s Parallel Analysis [15], a Monte Carlo simulation process, to generate a PCA predictor. Formally, the significant singular values that are associated with informative components are larger than the mean value of expected singular values. Such expected singular values are obtained by simulating normal random samples that capture the observed data in terms of sample size and number of variables. Therefore, the PCA predictor is given as follows:

\[
\sum_{i=1}^{l} \mathbb{I}(\mu_i > \overline{\sigma}_i^{perc}) \mu_i
\]  

(12)

where \( l \) is the number of singular values, \( \mathbb{I}\{\} \) is an indicator function, \( \mu_i \) is the \( i \)-th singular value, and \( \overline{\sigma}_i^{perc} \) are the mean values of expected singular values of the random data.

PCA predictors capture the overall variability of the matching via component analysis. The higher the variability the higher the indecisiveness and lack of consistency in the matcher’s behavior. Therefore, a prediction of low variability values for the top principal components is better.

For example, consider \( M' \), the non-binary transformation of the best match (Eq. 11) of Example 1. SVD uses the singular values \( \mu_1 = 1.00, \mu_2 = 0.22, \) and \( \mu_3 = 0.09 \). The principal components are calculated using the orthogonal matrices, a result of the SVD. Full calculations will be given in an extended version.
For the third predictor PCA, let $\overline{p}^{mc} = 0.28$, $\overline{p}^{mc} = 0.21$, $\overline{p}^{mc} = 0.13$ be a simulated normal random samples given $M'$. The predictor value would be $\overline{p} = 1 \cdot 1.00 + 1 \cdot 0.22 + 0 \cdot 0.09 = 1.22$. For this example we have received a relatively low values of the predictors (1.00, 0.22, and 1.22), which indicates that $M'$ represents a good match. For comparison, recall the similarity matrix from Table 1 (before the transformation). This matrix would produce the following predictors values (1.05, 0.28, and 1.31) indicating, by comparison, that $M'$ is better.

### 3.1.2 Entropy Predictors
Match diversity was shown to be useful in regulating a tradeoff between precision and recall [11]. We now propose two new diversity measures based on entropy, an information theory measure that estimates data diversity according to its expected informativeness.

The Mean Pure attribute-wise Entropy predictor (MPE) averages entropy over matrix rows, each for a specific attribute:

$$\frac{1}{n} \sum_{i=1}^{n} \left( \sum_{j=1}^{m} -M_{i,j} \ln M_{i,j} \right)$$

The Von Neumann Entropy predictor (VNE) (Eq. 14) (borrowed from the field of quantum statistical mechanics [36]) quantifies the diversity of a system, described by a density matrix. A similarity matrix in a pure state has zero entropy; whereas large entropy indicates a more uncertain match, and hence, more chance for lower quality. VNE is computed to be the entropy of the singular values of the similarity matrix (recall that we use singular value decomposition instead of the common eigen-decomposition):

$$- \sum_{i=1}^{n} M_{i,i} \ln M_{i,i}$$

Entropy predictors measure the amount of disorder in the matching system, quantifying the unpredictability of a match, thus capturing unwanted behavior. For $M'$ (Example 1), VNE $\approx 0.55$, a relatively low value, indicating a good match.

### 3.1.3 Matrix Norm Predictors
A matrix norm $\|M\|$ of a matrix $M$ is the equivalent of a vector norm in a vector space, which can be used to quantify errors in a matrix [16], assigning small norm value with a better matrix quality. In similarity matrices, errors can be attributed to the uncertainty involved in the matching process. To understand better this phenomenon, recall that the similarity matrix of Example 1 contains many non-zero entries. Each such entry, which does not relate to a correspondence in the reference match, adds to the cumulative error of the matrix. Also, three of the four correspondences in the reference match are assigned with low similarity scores (less than 0.25), which adds to the error introduced by the similarity matrix. We introduce four matrix norm predictors:

- **Norm1**, the maximum absolute column sum norm:

$$\|M\|_1 = \max_{j} \sum_{i=1}^{n} |M_{i,j}|$$

- **Norm2**, the square root of the maximum eigenvalue of the square matrix $M^HM$, where $M^H$ is the conjugate transpose. In our case, when there are no complex numbers, $M^H$ is equivalent to the original matrix transpose:

$$\|M\|_2 = \sqrt{\max_{i} (M^HM)}$$

- **NormInf**, a maximum absolute row sum norm:

$$\|M\|_\infty = \max_{i} \sum_{j=1}^{m} |M_{i,j}|$$

- **NormF**, the square root of the sum of the absolute squares of all the elements in the matrix:

$$\|M\|_F = \sqrt{\sum_{i=1}^{n} \sum_{j=1}^{m} |M_{i,j}|^2} = \sqrt{\text{trace}(M^HM)}$$

### 3.2 LRSM Algorithm
Equipped with matching features, we now detail the proposed Learn-to-Rerank Schema Matches (LRSM for short) algorithm, which has two phases, namely training and testing. The training phase’s input is a set of top-$K$ ideal lists $\sigma^{(K)}$ and its output is a learned model, used to rerank a new top-$K$ list at the testing phase. Each input list $\sigma^{(K)}$ is first transformed into a set of feature vectors $FV(\sigma)$, one per each match $\sigma_i \in \sigma^{(K)}$ (upper left corner of Figure 1). Each feature vector contains the values of various matching predictors over a single match. The algorithm then learns a model that, given a top-$K$ match list obtained for a newly (unseen) schema pair similarity matrix, represented by the same feature vectors (bottom left), is used for reranking that list (bottom right).

![Figure 1: Learn-to-Rerank Schema Matches Algorithm illustrated](image-url)
perfect relevance score. The features of this match serve as a desiderata for the best match and hence assist in the training of the model. As a lower bound we provide the original similarity matrix, before applying transformation (Eq. 11). Without transformation, any non-zero value in it is considered an attribute correspondence, regardless of how low the similarity value may be. Such a matrix, with its expected high recall (and low precision) serves as a regulator by letting the algorithm explore matrix entries that the rest of the list avoids, allowing the algorithm to generalize better.

To learn the optimal \( \hat{F}(\sigma) \), we use LambdaMART LTR algorithm, which combines LambdaRank and MART (Multiple Additive Regression Trees) and was proven to be successful for solving real-world ranking problems [2]. LambdaMART defines gradients with respect to each object (in our case \( \sigma \)) by using its feature vector representation \( (FV(\sigma)) \) and evaluation measures (e.g., NDCG) directly.

The overall scoring function is as follows:

\[
\hat{F}(\sigma) = \sum_{t=1}^{T} \omega_t \psi_t(FV(\sigma)),
\]

where \( \psi_t(FV(\sigma)) \) and \( \omega_t \) denote a given single tree’s score and its relative weight at iteration \( t \), respectively. Gradient decent is used, at each iteration, to improve tree scores until convergence [2].

**Algorithm 1 The LRSM Algorithm**

1. **Input**: Schema pair \((S, S')\), \(K\)
2. **Output**: A ranked list of matches \( \hat{\sigma}^{(K)} \)
3. \( \sigma \leftarrow \text{Match}(S, S') \) \( \triangleright \) Match \((S, S')\) using a matcher
4. \( \sigma^K = \text{Top-}K(\sigma) \) \( \triangleright \) Get top-\( K \) matches
5. \( f_{\text{max}} = \max_{\sigma \in \sigma^K}(f(\sigma)) \); \( f_{\text{min}} = \min_{\sigma \in \sigma^K}(f(\sigma)) \)
6. for \( i = 1 \) to \( K \) do
7. \( M' \leftarrow \sigma_i \) \( \triangleright \) apply Eq. 11
8. for \( j = 1 \) to \( |FV| \) do
9. \( FV[j] = \text{pred}_j(M') \)
10. end for
11. \( \hat{F}(\sigma_i) = \sum_{t=1}^{T} \omega_t \psi_t(FV(\sigma_i)) \)
12. end for
13. \( \hat{F}_{\text{max}} = \max_{\sigma \in \sigma^K}(\hat{F}(\sigma)) \); \( \hat{F}_{\text{min}} = \min_{\sigma \in \sigma^K}(\hat{F}(\sigma)) \)
14. \( \hat{F}_{\text{smoothed}}(\sigma) = \alpha \hat{F}(\sigma) - f_{\text{min}} + (1 - \alpha) \frac{f(\sigma) - f_{\text{min}}}{f_{\text{max}} - f_{\text{min}}} \)
15. \( \hat{\sigma}^{(K)} \leftarrow \text{sort } \sigma^K \text{ by } \hat{F}_{\text{smoothed}}(\sigma) \)
16. return \( \hat{\sigma}^{(K)} \)

Using the learned model \( (i.e., \hat{F}(\sigma)) \), we rerank a top-\( K \) match list (see Algorithm 1 for the pseudocode of the testing phase). The algorithm generates a similarity matrix (line 3) and a top-\( K \) list, where \( K \) is determined by Theorem 1 and given as input to the algorithm (line 4). \( f_{\text{max}}(\sigma) \) and \( f_{\text{min}}(\sigma) \) are computed (line 5) to be used in normalizing similarity values with the predicted F1 (line 14). A feature vector is created for each of the matches in the list, using the non-binary transformation method (Eq. 11) (lines 6-12). Each predictor \( \text{pred}_j \) is computed and added to the feature vector \( FV \). \( \hat{F} \) is computed for each match using Eq. 19 (line 11). Reranking minimizes the chance of model concept-drift [41] by interpolating the model score \( \hat{F}(\sigma) \) of each match \( \sigma \in \sigma^K \) with its similarity score \( f(\sigma) \) (line 14) using the normalizing values that were computed in lines 5 and 13. Finally, the reranked list \( \hat{\sigma}^{(K)} \) is returned (line 16).

## 4 Empirical Evaluation

We conducted a thorough empirical evaluation and report on its results herein. We validate Theorem 1 assumptions and analyze the impact of \( K \). We then evaluate the performance of Algorithm 1 in identifying the best match and analyze predictors as features in the top-\( K \) learning task.

### 4.1 Datasets

For each dataset used in the experiments, Table 2 details number of schemata, size range (in attributes), number of schema pairs, and reference match type (either 1 : 1 or 1 : \( n \)).

<table>
<thead>
<tr>
<th>Dataset</th>
<th>#Schemata</th>
<th>#Attr</th>
<th>#Pairs</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Web forms</td>
<td>247</td>
<td>10-30</td>
<td>147</td>
<td>1 : 1</td>
</tr>
<tr>
<td>Purchase Order</td>
<td>110</td>
<td>80-100</td>
<td>44</td>
<td>1 : ( n )</td>
</tr>
<tr>
<td>University Applications</td>
<td>16</td>
<td>50-150</td>
<td>182</td>
<td>1 : ( n )</td>
</tr>
<tr>
<td>OAEI</td>
<td>101</td>
<td>80-100</td>
<td>100</td>
<td>1 : ( n )</td>
</tr>
<tr>
<td>Beta</td>
<td>247</td>
<td>10-30</td>
<td>147</td>
<td>1 : 1</td>
</tr>
</tbody>
</table>

1. [http://www.nisb-project.eu/](http://www.nisb-project.eu/)

The Web-forms [10] dataset contains schemata that were automatically extracted from Web forms using the Onto-Builder extractor. Reference matches were manually constructed by human judges. The Purchase Order dataset [4] contains XML documents describing purchase orders extracted from various systems. The University Applications dataset [32] contains university application forms from various US universities, collected as part of the NisB project and transferred into XML Schema Definition (XSD) format. The OAEI dataset contains ontologies from the 2011 competition in the comparison track, using the bibliographic references domain. 100 ontologies are compared with a reference ontology.

The four benchmark datasets were designed to present schema pairs with differing levels of difficulties, introducing alongside easy matches also complex relationships and attributes, which may yield low precision and recall levels, even when using the strongest of matchers.

We also created a synthetic dataset, following a model of Marie et al. [23], to test the proposed features and the LRSM algorithm independently of the quality of matchers. We manipulated 147 reference match matrices from the Web-Forms dataset using two Beta distributions, one for correspondences in the reference match and one for non-corresponding attributes. The two distributions differ in their parameters, where attribute correspondences are assigned values that are closer to 1 and the others are closer to 0. Each of the attribute pair correspondences in the reference match was assigned a value that is Beta distributed for corresponding attributes. Out of the remaining values, we randomly chose entries and assigned them with values that were randomly generated using a Beta distribution for non-corresponding attributes. The number of non-corresponding attribute pairs is double the number of reference match pairs. Remaining entries are assigned with a value of 0.
All datasets, with the exception of Purchase Order, contain a sufficiently large number of schema pairs for the training phase of LRSM. For the Purchase Order dataset we use bootstrapping [7], a re-sampling method based on a small sample data.

### 4.2 Experimental Setup

Algorithms were implemented in Java and evaluation was performed using a Dell Inc. PowerEdge R720 server with a 20 true (40 virtual) cores Intel(R) and a CentOS 6.4 operating system. The various matchers, predictors, and the LRSM algorithm are all available online.3

Three schema matching tools were used, namely ORE, AMC, and COMA. OntoBuilder Research Environment? (ORE) is a research prototype for large scale matching experiments. The Auto-Mapping Core (AMC) [27], developed by SAP Research, provides an infrastructure and a set of algorithms to match business schemata. We used ORE’s embedded implementation of AMC’s Token Path algorithm. We also used two matchers from COMA 3.0,4 a state-of-the-art schema matching research tool, namely Threshold(\(\nu\)) and Max-Delta(\(\delta\)), whose details will be given shortly.

We used three matchers to generate similarity matrices (termed 1LM – First Line Matchers) and four matchers that create matches out of similarity matrices (termed 2LM – Second Line Matchers), see Table 3. The 1LMs are Term, WordNet, and Token Path. Term [9] compares attribute names to identify syntactically similar attributes (e.g., using edit distance and soundex). WordNet [30] uses abbreviation expansion and tokenization methods to generate a set of related words for matching attribute names. Finally, Token Path [27] integrates node-wise similarity with structural information by comparing the syntactic similarity of full paths from root to a node. Maximum weighted bipartite graph match (MWBG) [13] and stable marriage (SM) [23] are 2LMs that use well-known algorithms for solving the problem of matching two sets of elements given a score or ordering of preferences for each element. Threshold(\(\nu\)) and Max-Delta(\(\delta\)) are 2LM selection rules, prevalent in many matching systems [4]. Threshold(\(\nu\)) selects those entries \((i, j)\) having \(M_{i,j} \geq \nu\). Max-Delta(\(\delta\)) selects those entries that satisfy: \(M_{i,j} + \delta \geq \max\{\max_{i,j} M_{i,j}, \max_{i} M_{i,j}\}\). A top-\(K\) list, \(\sigma^K\), for each (dataset, 1LM) pair was created using the TopK algorithm [9] with the similarity function of Eq. 1 and reranked using Algorithm 1.

![Figure 2: Left: an example of \(U(\sigma^K)\) as a function of \(K\). Right: the curve illustrates expected values by the geometric distribution and the bars show the observed values. A vertical dashed line is drawn on both sides in the calculated \(K\) value (10).](image)

Table 3: List of 1LMs and 2LMs used in the evaluation

<table>
<thead>
<tr>
<th>Matcher</th>
<th>System</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Term (Synthetic)</td>
<td>Ontobuilder [25]</td>
<td>1LM</td>
</tr>
<tr>
<td>Token Path (Synthetic)</td>
<td>AMC [27]</td>
<td>1LM</td>
</tr>
<tr>
<td>WordNet (Semantic)</td>
<td>ORE [30]</td>
<td>1LM</td>
</tr>
<tr>
<td>Maximum weighted bipartite graph match (MWBG)</td>
<td>Ontobuilder</td>
<td>2LM</td>
</tr>
<tr>
<td>Stable marriage (SM)</td>
<td>Ontobuilder</td>
<td>2LM</td>
</tr>
<tr>
<td>Threshold</td>
<td>Coma</td>
<td>2LM</td>
</tr>
<tr>
<td>Max-Delta</td>
<td>Coma</td>
<td>2LM</td>
</tr>
</tbody>
</table>

For learning the LRSM model we used the RankLib tool,5 which is commonly used in the Information Retrieval community for LTR tasks. We used LambdaMART’s recommended parameter configuration [39] for training. The smoothing parameter \(\alpha\) (see Algorithm 1) was chosen (during the training phase) as \(\alpha \in \{0.1, 0.2, \ldots, 1\}\).

We compared LRSM against three baselines: 1) The original best match (\(\sigma_1\)), 2) Cross Entropy Matcher [11] (CEM), a state-of-the-art 2LM that utilizes match diversity as its optimization target and was previously shown to give best schema matching performance overall, and 3) ideal ranking (defined in Section 2.2).

### 4.3 Geometric Distribution Validation

We start with an empirical validation to the Geometric distribution assumption of Theorem 1. We then show an empirical evidence for the choice of \(K\).

#### 4.3.1 Geometric Distribution Assumption

For a given dataset, for each of its schema pairs, we first generated the corresponding match matrix by applying a given 1LM. Given each such matrix, a sample of top-30 matches was generated using the Top-K algorithm [9]. Let \(N\) be the number of schema pairs in a given dataset (e.g., \(N = 147\) for Web-forms dataset) and let \(O_1\) be the number of times \(\sigma_1\) (i.e., the original best match) was the one with the highest F1 (\(F(\sigma_1) = F^+\)). Using the moments parameter estimation method, for a given (dataset, 1LM) pair, we estimated the Geometric distribution parameter as follows: \(\hat{\theta} = \frac{O_1}{N}\). Next, we validated that \(K \sim \text{Geo}(\hat{\theta})\) using the Chi-Square goodness of fit test [34].

Detailed \(\theta\) estimation and p-values for every (dataset, 1LM) and the Beta dataset will be given in an extended version. For all (dataset, 1LM) pairs, \(K\) was successfully validated to be Geometrically distributed and results are statistically significant (i.e., all \(p\)-values < .05). Following this validation, using a desired confidence level \(p\), we can now safely derive \(K\) as suggested in Theorem 1.

3. https://bitbucket.org/tomers77/ontobuilder-research-environment
5. https://sourceforge.net/p/lemur/wiki/RankLib/
4.3.2 Empirical Evidence for the Choice of \( K \)

We illustrate the impact of the choice of \( K \) on LRSM performance using a (Web-Forms,Term) example. \( K \sim \text{Geo}(\theta \approx 0.42) \), and using \( p = 0.995 \), the confidence level according to Theorem 1 is \( K = 10 \). Figure 3 provides the result of reranking with LRSM, using precision \((P)\), recall \((R)\) and F1 of the top reranked match for varying \( K \in \{5, 10, 15, 20, 30, 40, 50, 75, 100\} \). \( P \) starts high, increases further until about \( K = 20 \) and remains stable and high from that point onwards. Both \( R \) and \( F1 \), however, obtain their maximum at \( K = 10 \), followed by a sharp drop, serving as an empirical validation of \( F1 \)’s theoretical bound. Such performance sensitivity to \( K \) makes it evident that a proper \( K \) selection is highly important for guaranteeing the best LRSM performance.

![Figure 3: P, R, and F for various (increasing) K values](image)

4.4 LRSM Evaluation

We now evaluate the impact of applying the LRSM model for reranking top-\( K \) match lists, utilizing both previously suggested state-of-the-art and the newly proposed predictors as feature vectors. We first analyze the quality of our newly suggested predictors vis-à-vis state-of-the-art predictors. We then show that the \( NDCG \) is preferable for configuring LambdaMART. Finally, we evaluate the LRSM algorithm and present feature analysis over the algorithm and the models.

4.4.1 Prediction Quality

We start by assessing the relative prediction quality of our proposed predictors (Section 3.1) compared to previously suggested state-of-the-art predictors. We wish to validate that the proposed predictors provide competitive (or even better) representation of the similarity matrix. Therefore, we show that our proposed predictors, as additional learning features for LRSM, improve the overall reranking performance. Following previous works [11], [32], we measure prediction quality using the Pearson’s \( r \) correlation between the predicted values and the actual match quality. We assessed prediction quality over 3300 correlation between matrix entries and mean entry values, \( MCD \) (Match Competitor Deviation) measures the average difference between matrix entries and mean entry values, \( 1 \)-valued entry per matrix row, while the former dominates \( \nu \)-valued entry per matrix row, while the former.

![Table 4: Pearson's r correlation to F1 (F), Recall (R) and Precision (P) of baseline (left) and new (right) matrix predictors.](image)

4.4.2 LRSM Optimization Function

LambdaMART uses evaluation measures directly in its optimization function and gradient. The main goal of LRSM is to demonstrating the diminishing growth of \( U(a^k) \). Figure 2(right) demonstrates, using the University Applications dataset, the resemblance between the expected values by the geometric distribution (red curve) and the observed values (blue bars). Both examples use the Term matcher.
better position the best match and accordingly we evaluate its performance in terms of the top match in the ranked list. Therefore, one could argue that optimizing LambdaMART in a binary manner (the best match is the top ranked match) is sufficient and there is no need to optimize the whole list to be ranked according to the evaluation measure. To compare these two alternatives, we validate the use of NDCG compared to Kendall Tau (τ), both ranking-based measures, and a binary evaluation (Bin).

<table>
<thead>
<tr>
<th>Model</th>
<th>Term</th>
<th>Token Path</th>
<th>WordNet</th>
</tr>
</thead>
<tbody>
<tr>
<td>LRSM\textsubscript{NDCG}</td>
<td>55.25</td>
<td>.24 .54</td>
<td>.44 .54 .44</td>
</tr>
<tr>
<td>LRSM\textsubscript{F}</td>
<td>57.32</td>
<td>.55 .35</td>
<td>.59 .09 .15</td>
</tr>
<tr>
<td>LRSM\textsubscript{Bis}</td>
<td>69.24</td>
<td>.35 .40</td>
<td>.59 .09 .15</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Model</th>
<th>Term</th>
<th>Token Path</th>
<th>WordNet</th>
</tr>
</thead>
<tbody>
<tr>
<td>LRSM\textsubscript{NDCG}</td>
<td>55.25</td>
<td>.24 .54</td>
<td>.44 .54 .44</td>
</tr>
<tr>
<td>LRSM\textsubscript{F}</td>
<td>57.32</td>
<td>.55 .35</td>
<td>.59 .09 .15</td>
</tr>
<tr>
<td>LRSM\textsubscript{Bis}</td>
<td>69.24</td>
<td>.35 .40</td>
<td>.59 .09 .15</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Model</th>
<th>Term</th>
<th>Token Path</th>
<th>WordNet</th>
</tr>
</thead>
<tbody>
<tr>
<td>LRSM\textsubscript{NDCG}</td>
<td>55.25</td>
<td>.24 .54</td>
<td>.44 .54 .44</td>
</tr>
<tr>
<td>LRSM\textsubscript{F}</td>
<td>57.32</td>
<td>.55 .35</td>
<td>.59 .09 .15</td>
</tr>
<tr>
<td>LRSM\textsubscript{Bis}</td>
<td>69.24</td>
<td>.35 .40</td>
<td>.59 .09 .15</td>
</tr>
</tbody>
</table>

Table 5: Comparing precision (P), recall (R) and F1 (F) over σ\textsubscript{1} (Orig), and training LRSM model of \(\hat{\sigma}(1)\) by NDCG (LRSM\textsubscript{NDCG}), Kendall Tau (LRSM\textsubscript{F}), and by binary trained LRSM model

Table 5 reports results in terms of precision (P), recall (R) and F1 (F) over σ\textsubscript{1} (Orig), and training LRSM model of \(\hat{\sigma}(1)\) by NDCG (LRSM\textsubscript{NDCG}), Kendall Tau (LRSM\textsubscript{F}), and by binary trained LRSM model

4.4.3 LRSM Algorithm Evaluation

Evaluation was performed using 5-fold cross validation. We randomly split each dataset into 5 folds and repeat the experiment 5 times, using 4 folds for training and the remaining fold for testing. We report on the average performance (P, R and F1) over all folds. For each fold we estimated \(\theta\) using only the training folds (see Section 4.3.1), fixed the appropriate \(K\) by Theorem 1 and optimized by NDCG (see Section 4.4.2). The user defined confidence was fixed to \(p = 0.95\) for Purchase Order and University Applications (datasets with lower \(\theta\) value) and to \(p = 0.99\) for the other datasets.

We trained three LRSM models to be compared with the baselines introduced in Section 4.2. LRSM\textsubscript{Base} and LRSM\textsubscript{Nnew} were trained using the previously suggested baseline and the newly proposed predictors separately (a full predictor list is given in Table 4). LRSM\textsubscript{Full} was trained using the full predictor (feature) set.

<table>
<thead>
<tr>
<th>Model</th>
<th>Term</th>
<th>Token Path</th>
<th>WordNet</th>
</tr>
</thead>
<tbody>
<tr>
<td>LRSM\textsubscript{Full}</td>
<td>.55 .56</td>
<td>.54 .55 .44</td>
<td>.44 .54 .44</td>
</tr>
<tr>
<td>LRSM\textsubscript{Base}</td>
<td>.55 .56</td>
<td>.54 .55 .44</td>
<td>.44 .54 .44</td>
</tr>
<tr>
<td>LRSM\textsubscript{Nnew}</td>
<td>.55 .56</td>
<td>.54 .55 .44</td>
<td>.44 .54 .44</td>
</tr>
<tr>
<td>CEM</td>
<td>.53 .54</td>
<td>.55 .55 .44</td>
<td>.44 .54 .44</td>
</tr>
<tr>
<td>Ideal</td>
<td>.57 .58</td>
<td>.57 .58 .44</td>
<td>.44 .54 .44</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Model</th>
<th>Term</th>
<th>Token Path</th>
<th>WordNet</th>
</tr>
</thead>
<tbody>
<tr>
<td>LRSM\textsubscript{Full}</td>
<td>.55 .56</td>
<td>.54 .55 .44</td>
<td>.44 .54 .44</td>
</tr>
<tr>
<td>LRSM\textsubscript{Base}</td>
<td>.55 .56</td>
<td>.54 .55 .44</td>
<td>.44 .54 .44</td>
</tr>
<tr>
<td>LRSM\textsubscript{Nnew}</td>
<td>.55 .56</td>
<td>.54 .55 .44</td>
<td>.44 .54 .44</td>
</tr>
<tr>
<td>CEM</td>
<td>.53 .54</td>
<td>.55 .55 .44</td>
<td>.44 .54 .44</td>
</tr>
<tr>
<td>Ideal</td>
<td>.57 .58</td>
<td>.57 .58 .44</td>
<td>.44 .54 .44</td>
</tr>
</tbody>
</table>

Table 6: Comparing precision (P), recall (R) and F1 (F) over σ\textsubscript{1} (Orig), \(\hat{\sigma}(1)\) by full LRSM model (LRSM\textsubscript{Full}), \(\hat{\sigma}(1)\) by basic LRSM model (LRSM\textsubscript{Base}), \(\hat{\sigma}(1)\) by LRSM model using just the new predictors (LRSM\textsubscript{Nnew}), \(\hat{\sigma}(1)\) by CEM objective function, and \(\hat{\sigma}(1)\) that have been obtained in case of ideal ranking (Ideal).

Results in Table 6 use precision (P), recall (R) and F1 (F) to compare the top match of each ranked list \(\hat{\sigma}(1)\) with the reference match \(\hat{\sigma}(1)\). Statistical significant differences in performance are marked with an asterisk using a paired two-tailed t-test with a Bonferroni correction [6] for \(p-value < .05\).
ing, $LRSM_{Full}$ achieves 75%-100% of the former’s F1 (on average). This shows that $LRSM_{Full}$ maximizes the top-K list’s potential. $LRSM_{Full}$ performs significantly better than $LRSM_{Base}$ and $LRSM_{New}$, with +2%-48% boost in F1 over the two, for the real-world datasets and +12% for the synthetic dataset, respectively. None of $LRSM_{Base}$ and $LRSM_{New}$ is dominant over all datasets. In some cases, $LRSM_{Base}$ and $LRSM_{New}$ performance actually degrades compared to the original top-K ranking. Evidently, combining the newly proposed and state-of-the-art predictors is more informative than using them separately.

### 4.4.4 LRSM Robustness

We further evaluate LRSM using a robustness measure, which quantifies the performance of a model by measuring its improvement over majority of datasets rather than an average improvement. Ideally, a robust model would never create a re-ranked list that is worse then the original ranking.

To this end, we use the robustness index (RI) measure [33] (also referred to as reliability of improvement index) that computes portion of times a referred model outperformed a baseline model.

Let $\Sigma_K$ denote a dataset with $N$ top-K list instances to re-rank, and let $\Sigma^K$ denote the resulting re-ranked dataset. For a given quality measure (i.e., P, R, or F1), the robustness index (RI) of a given re-ranking model is computed as follows:

$$ RI(\Sigma^K, \Sigma^K) = \frac{n_+ (\Sigma^K, \Sigma^K) - n_- (\Sigma^K, \Sigma^K)}{N}, \quad (20) $$

where $n_+ (\Sigma^K, \Sigma^K)$ and $n_- (\Sigma^K, \Sigma^K)$ denote the number of instances in which the re-ranking model performed better or worse than the original top-K ranking, respectively. It is worth noting that $n_+ (\Sigma^K, \Sigma^K) + n_- (\Sigma^K, \Sigma^K) \leq N$ and $RI(\Sigma^K, \Sigma^K) \in [-1, 1]$.

### Table 7: Comparing NDCG (See Eq. 4) over $\sigma^K$ (Orig), $\hat{\sigma}^{K}$ (by full LRSM model ($LRSM_{Full}$)), $\hat{\sigma}^{K}$ (by basic LRSM model ($LRSM_{Base}$)), $\hat{\sigma}^{K}$ by LRSM model using just the new predictors ($LRSM_{New}$), and $\hat{\sigma}^{K}$ by CEM objective function.

Table 7 analyzes the resemblance between the ordering of the top-K list obtained by the algorithm and the ideal ranking. Given the ideal top-K list we can use the ranking quality measure suggested in Section 2.2 (i.e., NDCG), to quantify the success of re-ranking the top-K list. Again, statistical significant differences in performance are marked with an asterisk using a paired two-tailed t-test with a Bonferroni correction for $p$-value < .05.

In terms of ranking performance, $LRSM_{Full}$ improves in NDCG over the original ranking by 17%-37%, on average, over all 1LMs. A significant improvement is demonstrated using the Beta dataset, (82% in NDCG), attributed to the learning abilities of LRSM (and CEM to almost the same extent). This substantiates our claim that it is possible to learn to rerank, given a good enough top-K list to start with.

### Table 8: RI, $n_+$, and $n_-$ of the $LRSM_{Full}$ model for the various (dataset, 1LM) pairs.

Table 8 summarizes our robustness analysis. Overall, on most datasets, LRSM achieves high precision RI values. As for recall RI, in five out of the eight use cases, LRSM’s re-ranking results in a lower recall. Yet, for all datasets we observe that LTM achieves a relatively high F1 RI values, providing a good tradeoff between precision and recall. The Beta synthetic dataset provides an interesting observation, where re-ranking was able to significantly improve on precision while maintaining the original recall, which was high to begin with.

### 4.4.5 LRSM Feature Importance

As part of model generation, the LambdaMART method produces trees that are used to calculate the final ranking of each similarity matrix (Section 3.2). These trees perform
feature selection and may choose to use only a subset of the feature set. Using RankLib we can analyze these trees, allowing an interesting reflection on the way each of the learned trees uses features in terms of weights, frequency, etc. The importance of each feature can be determined by computing the number of times a tree used a specific feature combined with the weight of this tree in computing the overall $F(\sigma)$.

Top-$K$ methods were proposed to effectively manage schema matching uncertainty [28], mostly ranking correspondences of individual attributes, e.g., in the context of holistic schema matching [14]. We rank whole matches, taking into account constraints such as $1 : 1$ matching, and propose a reranking method that increases the chance of finding better matches at the top of the list.

In this work we used representatives of three classes of list evaluation measures. The first ignores the list and looks only at $\sigma^1$. The second looks at mismatches, regardless of where they appear in the list (Kendall $\tau$). The third puts more emphasis on mismatches at the top of the list (NDCG). Other representatives of these classes also exist. For example, the Expected Reciprocal Rank (ERR) was shown to perform well in training machine learning models [2]. Our preliminary empirical evaluation showed NDCG to perform better.

Schema matching and ontology alignment [8] are closely related research areas, both aiming at finding matches between concepts. The two vary in their matching objects (schemata vs. ontologies), matching refinement (equivalence vs. richer semantics such as inclusion), and the underlying mathematical tools (e.g., similarity matrix analysis vs. logic). They do share many matching techniques (e.g., [26]) and their benchmarks can sometimes be used interchangeably (we have evaluated our model on OAEI, an ontology dataset benchmark, as well). To date, only a little work on predictors was done in the context of ontology alignment [11], yet we believe the research proposed in this work can be readily applied to ontology alignment.

6 Conclusion and Future Work

We propose LRSM, a learn-to-rerank algorithm, which reranks match lists, aiming at putting the best match at the top. A theoretical bound is set on the size of an initial list, probabilistically guaranteeing the presence of the best match possible in the list. We also offer new features, matching predictors, and show empirically their impact on reranking.

This line of work fits well with the quest of combining machine learning into classical data management tasks, in this case, that of schema matching. In future work, we intend to investigate the way humans and machine learning algorithms can interact to boost performance in matching tasks.

<table>
<thead>
<tr>
<th>Setting</th>
<th>Features ordered by importance from left to right</th>
</tr>
</thead>
<tbody>
<tr>
<td>Web</td>
<td>Term PCA VTDEV MPE MAX Dominant</td>
</tr>
<tr>
<td>Forms</td>
<td>Token Fall PCA VTDEV MPE MAX Dominant</td>
</tr>
<tr>
<td>OAEI</td>
<td>Term VTDEV PCA MPE MAX Dominant</td>
</tr>
<tr>
<td>Uni. App</td>
<td>Term VTDEV PCA MPE MAX Dominant</td>
</tr>
<tr>
<td>Token Fall</td>
<td>VTDEV PCA MPE MAX Dominant</td>
</tr>
<tr>
<td>WordNet</td>
<td>Term VTDEV PCA MPE MAX Dominant</td>
</tr>
<tr>
<td>VNE</td>
<td>Term VTDEV PCA MPE MAX Dominant</td>
</tr>
<tr>
<td>BMC</td>
<td>Term VTDEV PCA MPE MAX Dominant</td>
</tr>
</tbody>
</table>

TABLE 9: Top 5 informative features for each use case

Table 9 reports on the model’s top-5 most informative features for each combination of a dataset and 1LM. To start with, in the majority of cases, the most important feature was one of the newly proposed predictors (features), with at least two such features selected in the top-5 for each use-case. Among those features, the most notable ones are the PCA features (i.e., 1PC, 2PC, and PCA). To recall, such features reduce the complexity of the problem (introducing less noise), which in turn helps LRSM to make better decisions.

Further notable in their presence are the entropy-based features (i.e., MPE, VNE, and MCD, with higher presence of the first two). In most use-cases, at least one of those features appears in the top feature list. This demonstrates that entropy in general, which counts for either diversity (i.e., MPE or MCD) or reduced uncertainty (i.e., VNE), plays an important role in automatic schema matching decision making.

The newly proposed matrix norm features also have a respectful presence. Recall that such features encode the amount of uncertainty for schema matching decision making. Hence, such features bias the model towards matches that are less prone to errors. In all, combining the effect of all these newly proposed features can explain why $LRSM_{Full}$ is better suited for learning than $LRSM_{Base}$ or $LRSM_{New}$ separately (see again Table 6).

5 Related Work

To the best of our knowledge, ours is the first algorithm for re-ranking a top-$K$ schema match list. Being a feature-based algorithm, our main contribution is in the definition of a new set of features, specifically tailored to schema matching, and the definition of an appropriate optimization goal for the learning algorithm.

Matching predictors [32] were used in process matching [35], Data analysis recommendations [24], and Web table matching [29]. We show the benefit of internalizers, predictors that use matrix properties, in the task of match list reranking and extend the state-of-the-art in matching predictors by focusing our predictors on the negative aspects of matching.

References

International Conference on Data Mining (ICDM) 2018, Singapore, 2018. IEEE.


Avidgor Gal is a Professor of Data Science at the Technion — Israel Institute of Technology, where he received a DSc degree in 1995 in the area of temporal active databases. He has published more than 150 papers in leading journals and conferences on the topics of data integration, complex event processing, and process mining and he authored a book on Uncertain Schema Matching. He is a recipient of DEBS test-of-time award, and CoopIS and BPM best paper awards. He served as a program and general co-chair of multiple conferences. He is a member of the ACM and a senior member of the IEEE.

Haggai Roitman is a lead researcher and master inventor, a member of the Language & Retrieval team, IBM Research AI. His main research focus is on information retrieval (IR) and NLP, specifically, core IR problems such as query performance prediction, focused retrieval, fusion, neural IR and document summarization. He is an active member in the SIGIR research community, including numerous academic publications in leading conferences and journals, PC memberships, workshops and tutorial organizations. For his work on query performance prediction he has won the ACM CHI 2018 best paper award. He is also an adjunct lecturer in the computer science department in the University of Haifa.

Roei Shraga is a PhD student at Technion – Israel Institute of Technology. His research interests include data integration, machine learning, and their combination. More specifically, the role of humans in the data integration loop. He received his BSc in Industrial Engineering & Management from the Technion – Israel Institute of Technology.
APPENDIX A  
DETALIED EXAMPLE

Example 1. Table 1 provides an example of a similarity matrix over two purchase order schemata [21]. Schema $S$ has four attributes, namely poDay and poTime, representing a timestamp of the order. poCode represents a purchase order number and city represents the city of shipment. The other schema, $S'$, consists of three attributes, namely orderDate for the date of issuing the order, orderNumber for the order number, and city for shipment city, as before.

$$
\begin{array}{c|cccc}
  \downarrow S' & \text{poDay} & \text{poTime} & \text{poCode} & \text{city} \\
  \hline
  \text{orderDate} & \frac{4}{5} = 0.22 & \frac{4}{5} = 0.11 & \frac{4}{5} = 0.11 & \frac{4}{5} = 0.11 \\
  \text{orderNumber} & \frac{4}{5} = 0.09 & \frac{4}{5} = 0.09 & \frac{4}{5} = 0.09 & \frac{4}{5} = 0.00 \\
  \text{city} & \frac{4}{5} = 0.20 & \frac{4}{5} = 0.17 & \frac{4}{5} = 0.00 & \frac{4}{5} = 1.00 \\
\end{array}
$$

The similarity matrix of Example 1 is the outcome of the Term [9], a string-based matcher. Term matching is based on string comparison between attributes, e.g., the maximum common substring of the attributes poDay and orderDate is Da, and the similarity of the two terms is

$$
\frac{\text{length}(Da)}{\text{length}(\text{orderDate})} = \frac{2}{9} = 22\%. 
$$

A Maximum Weight Bipartite Graph algorithm was applied to generate a match, $\sigma = \{(\text{orderDate}, \text{poDay}), (\text{orderNumber}, \text{poTime}), (\text{city}, \text{city})\}$, marked by the boldface matrix entries, with $f(\sigma) = 0.22 + 0.09 + 1.00 = 1.31$, using Eq. 1.

Let the exact match be $\{(\text{orderDate}, \text{poDay}), (\text{orderDate}, \text{poTime}), (\text{orderNumber}, \text{poCode}), (\text{city}, \text{city})\}$. Then, we have that $P(\sigma) = \frac{2}{3} = 0.67$, $R(\sigma) = \frac{2}{4} = 0.5$, and $F(\sigma) = 2 \cdot 0.67 \cdot 0.5 = 0.57$.

$$
\sigma_2 = \{(\text{orderDate}, \text{poDay}), (\text{orderNumber}, \text{poCode}), (\text{city}, \text{city})\}, 
$$

with $f(\sigma_2) = 0.22 + 0.09 + 1.00 = 1.31$ (the same as for best match computed above), $P(\sigma_2) = \frac{3}{3} = 1.00$, $R(\sigma_2) = \frac{3}{4} = 0.75$, and $F(\sigma_2) = 0.86$. It is worth noting that in this case, the second-best match is in fact better, in terms of $P$, $R$, and $F$, then the top match. A description of several algorithms for computing top-K matches can be found in [9](Ch. 5).

$M'$ is a non-binary transformation of the best match (Example 1):

$$
M' = \begin{pmatrix}
0.22 & 0.00 & 0.00 & 0.00 \\
0.00 & 0.09 & 0.00 & 0.00 \\
0.00 & 0.00 & 0.00 & 1.00
\end{pmatrix}
$$

As an example of computing the three predictors consider $M'$, the non-binary transformation of the best match (Eq. 11) of Example 1. Using SVD, the following singular values are obtained: $\mu_1 = 1.00, \mu_2 = 0.22, \mu_3 = 0.09$, where the first two values are the first and second principal components. The principal components are calculated using the orthogonal matrices, which is a result of the SVD. $\min(m, n) = \min(3, 4) = 3$

We compute the singular values $\mu_i$ by finding the eigenvalues of $M' \cdot M'^T = \begin{pmatrix}
0.22 & 0.00 & 0.00 & 0.00 \\
0.00 & 0.09 & 0.00 & 0.00 \\
0.00 & 0.00 & 0.00 & 1.00 \\
0.0484 & 0.0000 & 0.0000 & 0.0000 \\
0.0000 & 0.0081 & 0.0000 & 0.0000 \\
0.0000 & 0.0000 & 1.0000 & 0.0000
\end{pmatrix}$

The characteristic polynomial is $\det(M' \cdot M'^T - \lambda I) = \begin{pmatrix}
0.0484 - \lambda & 0.0000 & 0.0000 & 0.0000 \\
0.0000 & 0.0081 - \lambda & 0.0000 & 0.0000 \\
0.0000 & 0.0000 & 1.0000 - \lambda & 0.0000 \\
0.0000 & 0.0000 & 0.0000 & 1.0000 - \lambda
\end{pmatrix}$

$\lambda_1 = 1.00 \rightarrow \mu_1 = \sqrt{1.00} = 1.00$

$\lambda_2 = 0.0484 \rightarrow \mu_2 = \sqrt{0.0484} = 0.22$

$\lambda_3 = 0.0081 \rightarrow \mu_3 = \sqrt{0.0081} = 0.09$

For the third predictor PCA, let $\mu_1^{mec} = 0.28, \mu_2^{mec} = 0.21, \mu_3^{mec} = 0.13$ be a simulated normal random samples given $M'$ obtained using Python 3.5.2. The predictor value would be $\text{PCA} = 1 \cdot 1.00 + 1 \cdot 0.22 + 0 \cdot 0.09 = 1.22$. For this example we have received a relatively low values of the predictors (1.00, 0.22, and 1.22) which indicates that $M'$ represents a good match. For comparison, recall the similarity matrix for Example 1 (A), before the transformation. This matrix would produce the following predictors values (1.05, 0.28, and 1.31) indicating, by comparison, that $M'$ is better. $A \cdot A^T = \begin{pmatrix}
0.22 & 0.11 & 0.11 & 0.00 \\
0.09 & 0.09 & 0.09 & 0.00 \\
0.20 & 0.17 & 0.00 & 1.00 \\
0.0847 & 0.0396 & 0.1727 & 0.0333 \\
0.0396 & 0.0243 & 0.0333 & 0.1727 \\
0.1727 & 0.0333 & 1.0689 & 0.1727
\end{pmatrix}$

The characteristic polynomial is $\det(A \cdot A^T - \lambda I) = \begin{pmatrix}
0.0847 - \lambda & 0.0396 & 0.1727 & 0.0333 \\
0.0396 & 0.0243 - \lambda & 0.0333 & 0.1727 \\
0.1727 & 0.0333 & 1.0689 - \lambda & 0.0333 \\
0.1727 & 0.0333 & 1.0689 - \lambda & 0.0333
\end{pmatrix}$

$\lambda_1 = 1.099 \rightarrow \mu_1 = \sqrt{1.00} = 1.05$

$\lambda_2 = 0.076 \rightarrow \mu_2 = \sqrt{0.0484} = 0.28$

$\lambda_3 = 0.0019 \rightarrow \mu_3 = \sqrt{0.0081} = 0.04$

For $M'$ (Example 1), one has that $\text{VNE} \approx 0.55$, a relatively low value, indicating a good match.