NEMA: Automatic Integration of Large Network Management Databases

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2020.3036414, IEEE Transactions on Network and Service Management

NEMA: Automatic Integration of Large Network Management Databases

Fubao Wu, Han Hee Song, Jiangtao Yin, Lixin Gao, Mario Baldi, Narendra Anand

***Abstract*—Network management, whether for malfunction analysis, failure prediction, performance monitoring and improvement, generally involves large amounts of data from different sources. To effectively integrate and manage these sources, automatically finding semantic matches among their schemas or ontologies is crucial. Existing approaches on database matching mainly fall into two categories. One focuses on the schema-level matching based on schema properties such as field names, data types, constraints and schema structures. Network management databases contain massive tables (e.g., network products, incidents, security alert and logs) from different departments and groups with nonuniform field names and schema characteristics. It is not reliable to match them by those schema properties. The other category is based on the instance-level matching using general string similarity techniques, which are not applicable for the matching of large network management databases. In this paper, we develop a matching technique for large NEtwork MAnagement databases (NEMA) deploying instance-level matching for effective data integration and connection. We design matching metrics and scores for both numerical and non-numerical fields and propose algorithms for matching these fields. The effectiveness and efficiency of NEMA are evaluated by conducting experiments based on ground truth field pairs in large network management databases. Our measurement on large databases with 1,458 fields, each of which contains over 10 million records, reveals that NEMA can achieve accuracy of 95%. We further compare with several other existing algorithms, and show that NEMA outperforms them by $7\% - 15\%$ in numerical matching and achieves the best trade-off for non-numerical matching.**

***Index Terms*—Network management, Database Matching, Graph Database.**

I. Introduction

With the development of big data analytics and data mining techniques, scalable network measurement and analysis techniques have been used in finding hidden information or patterns to help with network management, network monitoring and network security [35], [15], [29]. In one scenario of important network management, big network operators/vendors serving various customers own a multitude of databases on network products, configurations, incidents, troubleshooting and diagnosis information, etc. Since the databases serve different departments, they are usually separated from each other and independently managed by different departments and groups. However, network data are inherently designed to host "connections" among different devices, groups that the devices are in, and functionality the devices serve together; they are often required to be shared and put together used in many important tasks such as network prediction, semantic query, network diagnosis and fault detection. With the database connection and integration, network administrators can easily query related product configurations and performances around devices or services. This could also help them automatically identify the correlated network trouble tickets/issues and pinpoint problems [25], [1]. As many studies [11], [8], [32] point out, the discovery of matching fields is the most crucial and foremost step for the integration of databases. For this reason, in this paper we aim to automatically construct such matchings that lead to efficient network management and analysis.

There is an abundance of research on field matching and integration approaches for different data formats in different contexts such as relational databases, XML and object-oriented data formats [13], [27], [33], [6]. Existing database matching approaches include two main categories of techniques. One is based on schema-level matching, which exploits metadata using schema characteristics such as field names, data types, structural properties and other schema information [4], [14]. However, network management databases from different sources have different design standards and naming conventions [16], [10]. Even similar fields can have different names (e.g. "product_family" can also be named as "product series"). The other category is instance-level matching, which uses the record values of two fields to obtain the similarity and determine matched fields [9], [26], [24], [21]. Most of the previously proposed schemes rely on syntactic similarities, sampling or machine learning techniques that are meant to extract common patterns from the matching data corpus. However, it is difficult to directly apply these techniques to network databases or challenging to reliably construct dictionaries, corpus with large datasets when the naming convention is not consistent and diverse.

The challenges for matching these network databases are: (1) The database design is not ideally uniform. The data tables are created in different groups and departments by different people. Therefore, it is not reliable to use schema information to match data directly and easily. (2) The data is noisy and irregular. Some table fields contain unexpected records such as null, invalid values and typos. Some table records are either partly missing, incomplete or incorrect. Some fields have a large amount of records, while some have very few records. (3) The table contents are complicated and heterogeneous with numerical and non-numerical data formats. (4) No thesauri or auxiliary information exist that we can rely on for matching.

Fubao Wu and Lixin Gao are with the Department of Electrical and Computer Engineering, University of Massachusetts, Amherst. E-mail: fubaowu@umass.edu

Han Hee Song, Jiangtao Yin, Mario Baldi and Narendra Anand are with Cisco.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2020.3036414, IEEE Transactions on Network and Service Management

The only set of observation available is the database itself.

To solve these challenges, we propose an automatic matching technique for large NEtwork MAnagement databases (NEMA) to construct a graph database for network management and analysis effectively and efficiently. We propose several algorithms for numerical and non-numerical field matching respectively based on instance matching. Our main contributions are as follows:

1) We propose effective range difference and bucket dot product similarity metric to match numerical fields, and top priority metrics to match non-numerical fields.
2) To make the algorithm more scalable for large network management databases, we utilize min-hash-locality sensitive hashing algorithm for faster processing with a little scarification of accuracy.
3) We further propose to use the proposed similarity metric scores as features for classification to improve the reliability of our matching technique.
4) We experimentally demonstrate the effectiveness and efficiency of our matching algorithms in the real Cisco network management databases.

The rest of this paper is organized as follows. We define the problem in Section II. Section III describes NEMA matching algorithms in detail including both numerical field matching and non-numerical field matching. Experimental evaluation is presented in Section IV and related work is shown in Section V. We conclude our paper and present future work in Section VI and VII respectively.

II. Problem Description

Given structured network management databases, our goal is to create a graph database of network management by finding the most accurate matched field pairs among different tables in these databases. The matching of two fields is determined by the matching score measured by their record pair similarities. We utilize the matched results to construct a graph database for semantic query, network analysis, network prediction, etc. [23], [19].

To illustrate our problem and algorithms clearly, we use three sample tables below as a toy example throughout the rest of this paper. In Table I, PRODUCT Table (T_P) contains 2 fields $product\_id$ (primary key), and $family$ with 7 records respectively. In Table II, INCIDENT Table (T_I) contains 2 fields $incident\_key$ (primary key), and $prod\_key$ with 7 records respectively. In Table III, ORDER Table (T_O) contains 3 fields $order\_key$ (primary key), $incident\_id$, and $product\_name$ with 7 records respectively. The problem is to find whether these 7 fields match among the Table I, II and III by evaluating the matching of their records, then to construct a graph database for network analysis and management.

Table I: PRODUCT (T_P)

product_id	*family*
107	AIR series
108	con series
109	con series
150	47-7000
151	cisco0500
152	80-7066C
153	con5100

Table II: INCIDENT (T_I)

incident_key	*prod_key*
201	107
202	107
203	108
204	109
207	150
208	151
209	152

Table III: ORDER (T_O)

order_key	*incident_id*	*product_name*
301	201	AIR1212AC
302	201	AIR1002
303	203	con5122
304	204	mem-4700m-64d=
305	207	47-7066C
306	208	cisco0510
307	208	cs6012

Record Matching: Given two instances e1 and e2, a record matching function is defined as a 4-tuple: $\langle e_1, e_2, v, r\rangle$ where e1 and e2 are two field records; v is a similarity score (typically in the [0, 1]) between e1 and e2; r is a relation (e.g., equivalence, part-of, etc.) between e1 and e2. The matching function $\langle e_1, e_2, v, r\rangle$ asserts that the relation r holds between the record $e1$ and $e2$ with score v. For numerical field matching, only if two records are equal, they are considered matched. For example, records {107, 108, 109, 150, 151, 152 } in $product\_id$ field in Table I are matched as the equal records in $prod\_key$ field in Table II, respectively. For a non-numerical pair, however, if the similarity score of two records are higher than a threshold based on a similarity metric, they are considered matched. Here we consider the part-of relation in the network management databases for meaningful relations including subgroup versus group, product versus product family, subseries versus series, etc. For example, the record "con5122" in $product\_name$ field and the record "con5100" in $family$ field can have high similarity score with part-of relation. A record pair which is matched is called a matched record pair, and it is called a non-matched record pair if the pair is not matched.

Field Matching: A field here means a database field indicating the names of a column and the single piece of data stored. Given two fields f_1, f_2 and a threshold T, we define $sim(f_1, f_2)$ as the matching/similarity score (e.g. Jaccard similarity [5]) between two fields f_1 and f_2. If $sim(f_1, f_2)$ value is above T, we call (f_1, f_2) a matched field pair, otherwise it is called a non-matched field pair. In the toy example, $sim(product\_id, prod\_key)$ has a high matching score with Jaccard similarity, so $(product\_id, prod\_key)$ can be correlated and matched. Moreover, the field pair $(family, product\_name)$ can also be matched in terms of many matched record pairs such as some pairs $(AIRseries, AIR1002)$, $(47-7000, 47-7066C)$ and $(con5100, con5122)$, etc.

Graph Database: One effective way to utilize matched results is to construct a graph database for semantic query, network analysis, network prediction, etc. [23], [19], which is also our goal. We define a graph database as a labeled, attributed and undirected graph $G = (V, E, L_v, L_e)$ where V is the node set containing all the records appearing in the fields which match, E is the edge set between node pairs for node set V. L_v is a set of label information of node set V, which are the index attributes for the columns of a table. L_e is a set of label information of edge sets E, which are the relations of two records from two tables' column attributes. Specifically, when we construct a graph database from matching of relational databases, a node v consists of a field and a record value in a

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2020.3036414, IEEE Transactions on Network and Service Management

row; the label L_v of v comprises of the other field information in the same row as node v; an edge e is the matching between two records; l_e indicates the field information when two records of the fields matches. Fig. 1 shows an example of a constructed graph database from parts of matched results in Tables I, II and III. For example, a specific node v "product_id: 107" in the Table I has a node label l_v "PRODUCT" and "Family:AIR series". Node v matches with "prod_key: 107" in Table II, so it is connected to the node "Incident_key:202" with a Product-incident relation, to the node "Incident_key: 201" with a Product-incident relation, and to the node "Family:AIR series" with a Product-family relation.

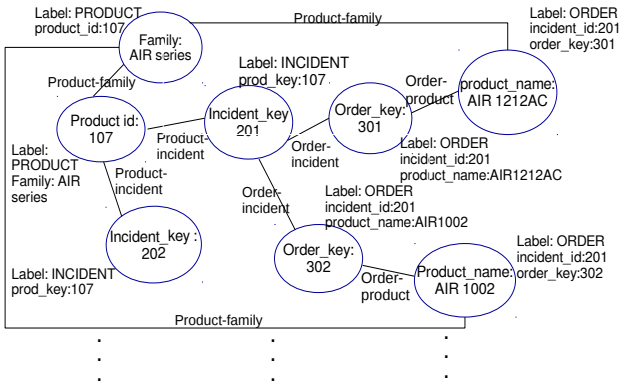

Fig. 1: Graph database created from Table I, II and III.

III. Matching Algorithm

To find whether two fields match, one simple way is to use field name matching. If the name of two fields are the same or similar, they are matched. However, this is not reliable for many database sources because they are noisy and irregular. Moreover, the network databases comprise of numerical and non-numerical fields with different attributes and matching requirements. For numerical field matching, we consider equivalence relation between record pairs. For non-numerical field matching, however, we do not directly consider the equivalence relation as the matching standard. non-numerical record values are possible to be semantically correlated with different names. For example, the non-numerical fields $family$ and $product\_name$ in the Table I and III have very few common characters on their names, but they are semantically correlated that $product\_name$ has a part-of relation with $family$. Moreover, in terms of field records, records "cisco0510" and "cisco0500" in these two fields can be considered belonging to the same family and being matched with a high similarity. However, the record pair "47-7066C" and "80-7066C" are considered to be non-matched with different families, even if the two strings have many common characters. (Details will be covered in III-C). Hence, we use the record matching to decide whether two fields match to improve matching accuracy and satisfy semantic matching. Overall, we match numerical and non-numerical fields separately and design different matching algorithms respectively.

A. System Overview

The system overview is shown in Fig. 2. We divide the structured data into numerical data with only numerical fields and non-numerical data with only non-numerical fields. In each part, we develop an independent matching algorithm for field matching. Matching algorithms for numerical and non-numerical data are quite different, which will be introduced in section III-B and III-C. The results of each part are combined together to load into a graph database.

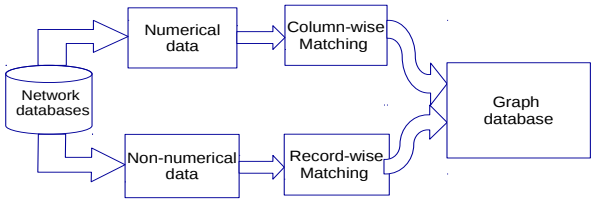

Fig. 2: System diagram of automatic integration for network management databases.

B. Numerical Field Matching

Numerical fields are table fields with records which are numerical values. For example, the $incident\_key$ and $prod\_key$ in Table II are numerical fields. Their record values serve as a basis for similarity metrics of fields. We define each numerical field record values as a set. This is transferred to a problem of set similarity.

1) Range Difference and Bucket Dot Product Similarity Metrics: There are some common methods for solving set similarity including Jaccard index, Dice index, Hamming distance, cosine similarity [7], etc. However, it is not practical to just use one method to get accurate decision bounds of matching because of the noisiness and complexity of the structured databases. We propose a synthetic column-wise numerical field matching algorithm to get the decision bounds to determine whether two fields are matched or not. The numerical field algorithm is shown in Fig. 3.

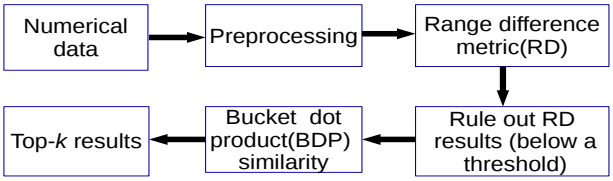

Fig. 3: Numerical field matching flow.

The process of this algorithm is shown as follows:

- We preprocess the numerical data, such as removing null values, negative values and some exceptional non-numerical values in every field.
- We apply range difference similarity metric to all the preprocessed numerical field pairs between every two tables.
- After we get the range difference similarity score for each field pair, a threshold T_r (which will be discussed later) will be decided to cut the filtered results.
- Finally, bucket dot product similarity metric will be applied to the range difference similarity metric's filtered

results. Then we sort the similarity scores of all the pairs to select the most correlated field pairs, that is, top-k result as matched pairs with scores above a certain threshold T_b will be selected to input to a graph database.

a) Preprocessing: To deal with the irregular and noisy data, we do some preprocessing of field record values before the formal similarity calculations. This includes removing null values, negative values, and considering unique values only. We do not consider negative values because they are useless and noisy data in the real databases. Almost all the table fields are about identifications, or numbers which can possibly be matched among them. In our databases, an average of 6% of field records are removed (excluding unique value reduction) which does not impact the instance-based algorithms.

b) Range Difference Similarity Metric: Considering the noisy and sparse characteristics of data, Jaccard similarity [5] as a similarity metric, which measures how many common values between two sets, is not ideal for differentiating some matched pairs and non-matched pairs. For example, for an non-matched field pair with limited number of records, the number of common values in these two fields might take a large portion and hence the Jaccard similarity is very high for them. Their distribution of range, however, can be quite different, which is probably not to be matched in most cases. Therefore, we propose range difference (RD) similarity metric to measure the distribution of these field pairs first. Using RD similarity metric first, we can effectively prune lots of unwanted computations, which can also hugely reduce time consumption for further matching. Given a field set A, we sort the record value and then get the different percentile (10th, 20th, 30th,..., 90th percentile). The percentile ith value in A is recorded as A_i. Therefore, given two field record sets, A and B, the RD value D_i for each percentile i is given as

$$D_i = \frac{|A_i - B_i|}{|A_i + B_i|} \tag{1}$$

We use 20th plus 30th percentile as the low range coverage, and 80th plus 90th percentile as the high range coverage, to cover the distribution of field record values. Hence the RD similarity score for a field pair (A, B) is defined as:

$$\mathrm{RDS}(A, B) = 1 - \frac{D_{20} + D_{30} + D_{80} + D_{90}}{4} \tag{2}$$

To keep consistent with the general similarity metric and be convenient for comparisons, we use 1 minus the averaged RD value as the RD similarity score RDS. The metric based on this similarity score is called RD similarity metric. Using this similarity metric, we can get the similar distribution for matched pairs. RDS value is in [0, 1]. The bigger the similarity score, the more correlated the pair. There are three cases about the similarity score here: (1) If there are no overlaps between two field ranges, RDS would be as low as the minimum value 0. (2) If two fields have similar distributions, RDS would be higher, up to 1. (3) If two field ranges overlap at the head, tail or in the middle, RDS can fall into a middle value.

In our toy example, the matched pairs $(incident\_key, incident\_id)$ and $(product\_id, prod\_key)$ have RDS values as high as 0.996 and 0.999 respectively. In contrast, the pairs $(incident\_key, product\_id)$ and $(prod\_key, incident\_id)$ have no overlaps with RDS value 0, which are not matched pairs. The more correlated the field pairs are, the higher RD similarity score they have. Therefore, using a threshold T_r to filter results, we can almost rule out case (1) and part of case (3), then mainly consider case (2) to differentiate them further. To minimize the error of RD similarity score in the first step, we can use a conservative threshold close to the boundary to only filter out definite non-matched pairs, which will be discussed in the section IV.

c) Bucket Dot Product Similarity Metric: After we consider the distribution of field pairs with range difference similarity metric, we propose bucket dot product (BDP) similarity metric to further refine the filtered results of RD similarity metric. BDP similarity metric is to divide the whole concatenated ranges of two fields into different bucket/bins and compress each bucket as one point to calculate dot product similarity. The intuition behind this is that matched pairs generally have more common values than non-matched pairs. If we increase the bucket size up to a certain value to calculate dot product, it can make the similarities of all the non-matched pairs decrease more, and meanwhile make the similarities of all the matched pairs drop less, therefore it effectively increases the similarity gaps between matched pairs and non-matched pairs. Therefore, a good design of BDP will help significantly differentiate between matched pairs and non-matched pairs.

The general dot product similarity of two vectors X and Y with n elements is DP, defined as follows: $DP(X, Y) = \sum_{i=1}^{n} X_i \cdot Y_i$. We use the bucket number (b_n) to determine the number of buckets for calculating the dot product. Given two field record sets A and B, we first derive the required vectors A_v and B_v for the input to the BDP similarity calculation. The vectors A_v and B_v derived from A and B are constructed in this way. Given two sets A and B, we concatenate A and B's value ranges as a combined set C, and then divide C into several buckets according to the b_n. If there is any one value in A or B falling in a bucket, the bucket point for A_v or B_v' is 1, otherwise it is 0. Then we apply the general dot product similarity to A_v and B_v. Therefore, the BDP similarity score (normalized) is defined as follows:

$$\mathrm{BDPS}(A_v, B_v) = \frac{\sum_{i=1}^{b_n} A_{vi} B_{vi}}{|A_v||B_v|} \tag{3}$$

where b_n decides the sparsity/density of range distributions. Since sets A and B usually have different sizes with different ranges, it would make sense for b_n the same for each set. For example, we calculate the BDPS for a field pair $(incident\_key, incident\_id)$ in Table II and III as A and B with $b_n = 3$. We first concatenate these two field ranges into a set $\{201, 202, 203, 204, 207, 208, 209\}$. Then we construct a set C =$\{\{201, 202, 203\}, \{204, 207, 208\}, \{209\}\}$ according to the bucket number b_n. After that, we get the vector A_v = $\{1, 1, 1\}$ and B_v = $\{1, 1, 0\}$. Finally, the BDPS is 0.816, which is high for matching. If we set b_n as 4, BDPS for this pair is 1, which is the highest for matching.

b_n is also an important factor to affect the quality of this metrics. According to our experimental observations, it is affected by the data range and distribution. Generally, matched

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2020.3036414, IEEE Transactions on Network and Service Management

pairs would have more similar ranges than non-matched pairs. A trade-off value of b_n would effectively improve matched pairs' similarities more and also not help grow non-matched pairs' similarities much, which can potentially increase more gaps between matched and non-matched pairs. The selection of b_n will be discussed later in the section IV.

d) Final Top-k Selection: To construct a high quality graph database, more true positive field pairs are preferable from higher similarity scores. Also, manual thresholds could lead to selection instability of field pairs with low similarity scores, so we seek top-k to further refine the quality of matching for the graph database. We sort all the candidate pairs by the similarity scores in a non-ascending order, then we verify this final field pair matching results to select top-k field pairs, which involves only a little human labor.

2) SVM Classification-based Matching: Our previous proposed similarity metric-based algorithm for numerical field matching involves three manual thresholds to determine field matching. To overcome the problem of selecting thresholds manually, we propose a classification-based learning approach to decide matching or non-matching for numerical fields here. The target label is whether a field pair matched or not matched. Hence, the matching could be modeled as a binary classification problem. Using a classification model and previously proposed similarity scores as features, the model can learn the internal thresholds and decide a given field pair is matched or not. We select support vector machine (SVM) as our binary classification for the reason that it works well on unstructured data and scales well with high dimensional data.

Features for Classification: In the numerical field matching, we have previously generated RD similarity score (RDS) and BDP similarity score (BDPS). We propose to use these similarity scores as features. To generalize our model, we generate 19 different BDPS based on different bucket numbers with 19 percentiles [5th, 10th, 15th,..., 85th, 90th, 95th] of combined records from each ground truth field pair. As a consequence, we have 20 features in total for classification. With these features and data, the thresholds in previous similarity metric-based numerical field matching algorithm involved in deciding the boundary decision of RD or BDP, and bucket number b_n could be internally learned through our SVM model.

C. Non-numerical Field Matching

Here we proposed algorithms for non-numerical field matching, including top priority match metric, minHash-locality sensitive matching, and SVM-based classification.

1) Top Priority Match Metric: We propose an algorithm for non-numerical field matching–top priority match metric (TPM) for fast filtering, and match ratio score for final similarity computations. The diagram is shown in Fig. 4. The main process of this algorithm is as follows:

- Preprocess non-numerical data: this is an important step that decides the quality of our non-numerical field matching algorithm. After splitting non-numerical data from the original databases, we use our designed natural language processing methods of segmentation, stemming and prefixing for every field record.

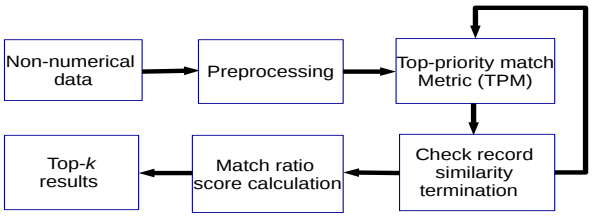

Fig. 4: Non-numerical field matching flow based on TPM.

- Calculate the record-wise similarities iteratively: It is very time-consuming to apply cosine similarity to the combination of every record pair in a field pair. Considering the scalability of large-scale data matching, we propose top priority match metric for record-wise similarity calculation. In the process of iterative computations, we check the termination condition to terminate the iterations earlier, which significantly reduces time complexity.
- Calculate match ratio score: after record-wise similarity computations for every field pair are finished, we calculate the defined matching ratio score for each field pair.
- Select top-k results: we sort the field pairs by the matching ratio score in a non-ascending order, and top-k field pairs are selected as the final results for a graph database.

Here, we discuss each proposed steps in detail.

a) Preprocessing: Non-numerical field matching considers partial match between two strings. For example, product "cisco0510" and product "cisco0500" are in the same series, which is considered as a partial match. Hence, we propose the following preprocessing method. (1) Parse every record string A, remove null value, separate alphabetic and numerical characters into different new substrings, and tokenize the string words. (2) Stem the alphabetic strings of the original record and new substrings. (3) Reserve the prefixes with a certain length of the original numerical strings and the new substrings if they are digital substrings. The prefix length is 2 here according to our experiments.

Each record string is preprocessed in those three steps above. For example, we have an original field record R {'mem-4700m-64d='} in Table III. We can obtain a string collection X {'4700', '64', '4700m', 'd', 'm', 'mem 4700m 64d', '64d', 'mem', '47xx'} after preprocessing R.

b) Top Priority Match Metric for Record-wise Similarity: One intuitive way is to preprocess all the combination of record pair comparisons and calculate the similarity of each record-wise pairs. That would be very time-consuming or even unfeasible when the data are large. Specially, if two fields are not correlated as a matched pair, it would be costly for useless computations. Therefore, we propose a fast record-wise matching algorithm called top priority match (TPM) metric for record-wise similarity to fulfill this. Intuitively, if two fields A and B are correlated, there will be a high percentage of record-wise pairs that have higher similarities. The probability of a matched record pair encountered is higher than non-matched record pairs. Therefore, we first sort all the preprocessed records in each two fields A and B, then we compute how many of records in A are matched with records in B from top to bottom, and vice-versa. The comparisons

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2020.3036414, IEEE Transactions on Network and Service Management

can hence be terminated as long as the current record pair similarity achieves below the similarity threshold T_{rn} we set for deciding the matching of a record pair, which greatly reduce the times of comparisons with combinations.

c) Record Pair Similarity: In this fast record-wise comparisons, the record pair similarity used is cosine similarity between two record collections after preprocessing two records. It decides how and when to reduce the comparisons of matching in a fast and effective way. A threshold T_{rn} is to decide how similar a record pair is as a matched record pair, which can also be adjusted by users.

Given a preprocessed string collection X and another preprocessed string collection Y, we remove duplicated elements and transfer them into a set $XY(X$ union $Y)$. Next we generate a binary vector V_x and a binary vector V_y according to the value distribution of X and Y in XY, then we calculate the cosine similarity $sim(V_x, V_y)$ between V_x and V_y by getting their dot product divided by their magnitude multiplication.

$$sim(V_x, V_y) = \frac{V_x \cdot V_y}{|V_x|\,|V_y|} \tag{4}$$

For example, we have a preprocessed string collection X {cisco, 0510, cisco0510, 05xx}, and Y {cisco, 05xx, 0500, cisco0500}, we transfer them into a set of X union Y, XY {cisco, 0510, cisco0510, 05xx, 0500, cisco0500}. Then the binary vectors generated according to X, Y and XY are V_x {1, 1, 1, 1, 0, 0} and V_y {1, 0, 0, 1, 1, 1}. Finally, we calculate the cosine similarity of V_x and V_y as the similarity of X and Y, that is, $sim(X, Y) = (1 + 1)/(2 * 2) = 0.5$.

d) Matching Ratio Score: Matching ratio score is proposed to calculate the final similarity for a field pair. After we have gone through the reducing comparisons for record-wise similarities, we select the number of record pairs that have similarity scores above T_{rn}. A matching ratio score as a final field pair similarity is the average of ratios of top matched record pairs calculated as follows:

Given two non-numerical sets A and B, there are m items $\{a_1, a_2, ..., a_m\}$ in A and n items $\{b_1, b_2, ..., b_n\}$ in B. The matching ratio score (MRS) between A and B is defined as follows.

$$\text{MRS}(A, B) = \frac{1}{2} * (\frac{\sum_{i=1}^{m} A_i}{m} + \frac{\sum_{j=1}^{n} B_j}{n}) \tag{5}$$

where

$$A_i = \begin{cases} 1 & \text{if } \exists\ b_j \in B,\ sim(a_i, b_j) >= T_{rn} \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

and

$$B_j = \begin{cases} 1 & \text{if } \exists\ a_i \in A,\ sim(b_j, a_i) >= T_{rn} \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

where $sim(a_i, b_j)$ and $sim(b_j, a_i)$ are the cosine similarities of the record pairs (a_i, b_j) and (b_j, a_i), respectively. MR value is in [0, 1] and it is the final similarity score to decide the correlation of each field pair.

e) Final Top-k Results: Similar to numerical field matching, matched non-numerical field pairs in the result list are more meaningful and important than non-matched field pairs, so we select top-k results of non-numerical field pairs sorted with MRS in non-ascending order for a graph database. K value can be selected by users for deciding most effective field pairs in a graph database and limiting the size of the graph database.

2) MinHash-Locality Sensitive Matching: The proposed TPM metric can be effective to distinguish between matched and non-matched non-numerical fields. However, it possibly involves all the pairwise record combinations in the worst time complexity, which is time-consuming for large databases with millions of records. Therefore, we propose applying more scalable minHash-locality sensitive hashing algorithm (MH-LSH) [18] to estimate the matching score of non-numerical fields in the databases. It can greatly reduce the comparison size and time for non-numerical record-wise pairs with little cost of matching accuracy down.

The proposed diagram for non-numerical field matching based on MH-LSH is shown in Fig. 5. The main process of the algorithm is as follows:

- Preprocess non-numerical data: this step is the same as the preprocessing step of TPM algorithm.
- Select matched field pairs fast: we apply the locality sensitive hashing technique in the database field matching for fast selecting field pairs that are correlated.
- Field pair similarity calculation: We use minHash technique to estimate the matching score of field pairs.
- Select top-k results: Same as the operation for non-numerical field matching based on TPM, we sort the field pairs by the estimated matching score in a non-ascending order, the top-k field pairs are selected as the final results of field pairs for a graph database.

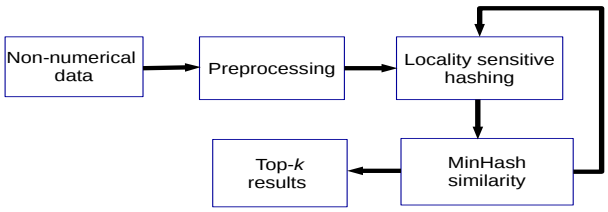

Fig. 5: Non-numerical field matching flow based on MH-LSH.

Here, we discuss important steps in detail.

a) Matching Score Estimation with MinHash: The similarity score of matched field pairs can be estimated fast with minHash signatures. Given two fields A and B, we can evaluate the similarity between them as follows. We choose n hash functions $h_1, h_2, ..., h_n$. For each hash function h_j, we let a signature of set A be $sgn(A) = \min_{i:a_i \in A} h_j(a_i)$ for $j \in n$. Let a signature of set B be $sgn(B) = \min_{i:b_i \in B} h_j(b_i)$ for $j \in n$. Then, the probability that the two sets have the same minHash signatures is used to estimate their similarity.

$$\text{MHSim}(A, B) = P\{sgn(A) = sgn(B)\} \tag{8}$$

Here each record in a field A or B is also preprocessed with the same preprocessing method of TPM algorithm. The

preprocessed record strings are combined into a new set like a word set in a document. We also use k shingle (a substring of length k) to create a set of k-shingles strings and apply n different hash functions on the set of strings.

b) Field Pair Selection with Locality Sensitive Hashing: Performing pairwise similarity measurement can be time consuming with large amounts of field pairs available. In order to identify which field pairs are similar quickly, we propose using locality sensitive hashing (LSH) to select the candidate field pairs.

The values of minHash signature $sgn(A)$ for one field A are grouped into b-tuples (referred to as sketches) with r rows. Similar field pairs have similar minHash signatures and hence have a high probability of having the same sketches. Moreover, dissimilar pairs have low chance of falling into the same sketch. The probability that two fields of A and B have at least one sketch (of size b) in common out of r is

$$P_C(A, B) = 1 - (1 - \text{MHSim}(A, B)^b)^r \tag{9}$$

Therefore, we can find the candidate pairs with the designed number b and r. The selection of b and r is generally decided by a threshold $t = (1/b)^{1/r}$ shown in [18], which indicates how similar the two fields is to be considered as a candidate pair, and can also be set by users. In this way, if pairs with similarity above $P_C(A, B)$, they will be selected as candidate pairs to be further estimated, and the matching score between them with minHash will be calculated.

3) SVM Classification-based Matching: Similarly as numerical field matching, we propose a SVM classification-based matching method to avoid the manual thresholds involved in the top-priority match (TPM) or minHash-locality sensitive (MH-LSH) metrics for non-numerical matching.

Features for Classification: During the calculation of TPM matching ratio score (MRS) and MH-LSH score (MHSim), we have one important record-pair threshold T_{rn} while calculating TPM matching ratio score. To avoid the record similarity threshold T_{rn}, we use a broad range of 7 different T_{rn} values in [0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8] to calculate different matching ratio scores as features. Combined with MHSim, 8 features of scores are obtained for each non-numerical field pair. With these features and data, the thresholds T_{rn} and the final threshold for deciding the matching boundary are internally learned through our SVM classifier.

IV. Experimental Evaluation

We evaluate our technique NEMA for structured network management database matching. Specifically, we measure the effectiveness of NEMA using ground truth for numerical and non-numerical data that are annotated by humans. Meanwhile, experiments on a large dataset are also conducted, and we show the top-k effective results of matching field pairs. Moreover, comparisons of NEMA with other existing algorithms are also shown.

A. Dataset

The structured network management databases available in the form of database tables are provided by Cisco Systems, Inc. The dataset includes heterogeneous and diversely distributed "install_base" and "service_request" databases, which are generated by various departments of the corporation.

In these databases, there are 21 tables which contain 1,458 columns. Each column has 10 million records on the average. Out of them, there are 679 numerical fields and 779 non-numerical fields. Therefore, a complete match involves the maximum 1,067,882 field matching decisions. With primary key constrains in numerical field matching, there are 5 "primary keys" on average in each table, which would reduce to 374,326 field pairs matching.

We have ground truth field pairs that are annotated by humans to be matched or non-matched field pairs for a subset of the data. There are 60 balanced ground truth field pairs in numerical dataset and 40 balanced ground truth field pairs in non-numerical dataset. For future reference, table names in service_request database start with "T_", and start with "X_" in install_base database, respectively.

B. Experimental Setup

We implement NEMA system in Python. To evaluate the effectiveness of NEMA , we evaluate the numerical and non-numerical algorithm parts, respectively. For each part, we first evaluate its algorithms based on the ground truth data. Then all the column pairs in the large dataset are evaluated in the following experiments shown in sections IV-C2 and IV-D2, which shows the effectiveness of NEMA. Then we evaluate the SVM classification-based algorithms. Finally, we compare with the common matching system COMA [3] on the ground truth for both schema-level and instance-level matching. Our quality evaluation is based on the balanced ground truth of positive and negative field pairs. We use common metrics-precision, recall, "accuracy" (ACC) and "F1" score (F1) to evaluate our field matching algorithms.

C. Evaluation based on Numerical Data

We evaluate our technique NEMA on the numerical data in two parts. Because of the limited ground truth field pairs, other than the common 80/20 splitting ratio, we randomly select 60% of matched field pairs and 60% of non-matched field pairs to make balanced ground truth from the whole ground truth field pairs to determine the thresholds of NEMA numerical algorithms. The rest 40% of them will be tested to show the quality in Section IV-F. The more proportion for test data also helps reduce the randomness and improve the generalization of our algorithms. The matching results of all of other numerical field pairs are also described in Section IV-C2.

1) Evaluating of Ground Truth: We show the evaluation result of NEMA numerical algorithms and the compared baseline method-Jaccard similarity using numerical ground truth (We choose Jaccard similarity since it is an exemplar method considering common values of two sets for similarity calculation). In the dataset, there are 30 matched ground truth field pairs which are originally from fields pairs annotated by humans or from join operations in the databases and proved to be matched field pairs. Referring to the undersampling technique, we randomly sample 300 non-matched field pairs

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2020.3036414, IEEE Transactions on Network and Service Management

confirmed by humans and select 30 non-matched field pairs from them as non-matched ground truth to construct balanced data instances of positive and negative field pairs.

Table IV: Example of numerical ground truth field pairs

No.	Table.fieldA	Table.fieldB	Matched class
1	T_INCI.prod_hw_key	T_HW_PROD.bl_prod_key	1
2	T_INCI.cur_ct_key	T_CT.bl_ct_key	1
3	T_INCI.up_tech_key	T_TECH.bl_tech_key	1
4	T_INCI.inci_id	T_INCI_I2.inci_id	1
5	T_INCI.bl_cot_key	T_COT.bl_cot_key	1
6	T_INCI.inci_id	T_OR_HD.inci_id	1
7	T_INCI.item_id	T_PROD.item_id	1
8	T_INCI.ins_site_key	T_SITE.bl_site_key	1
9	T_OR_LN.prod_key	T_PROD.bl_prod_key	1
10	T_OR_HD.header_id	T_OR_LN.header_id	1
1	T_OR_HD.order_dur	T_OR_LN.loc_key	0
2	T_CAL.bl_cal_key	T_PROD.item_id	0
3	T_INCI_I2	T_DEFT.deft_id	0
4	T_INCI_I2.res_time	T_TECH.sub_tech_id	0
5	X_PRO.list_price	T_TECH.sub_tech_id	0
6	T_INCI_I2.res_time	T_TECH.bl_tech_key	0
7	T_OR_HD.deliv_dur	T_DEFT.deft_key	0
8	T_OR_LN.hold_dur	T_TECH.sub_tech_id	0
9	T_IN.serlevel_key	T_INCI.last_dur	0
10	T_IN.closect_key	T_INCI.resp_tz	0

Table IV shows 10 matched and 10 non-matched field pair examples of numerical ground truth. Columns "Table.field A" or "Table.field B" shows a table name and its field pair name to be compared. The matched class column indicates that the field pair is matched with value "1" or non-matched with value "0". For example, in the first row, "T_INCI.prod_hw_key" indicates a field "prod_hw_key" in the table "T_INCI", and "T_HW_PROD.bl_prod_key" indicates a field "bl_prod_key" in the table "T_HW_PROD". This field pair about products' key is matched, indicated with "1" in the matched class value. The remaining rows share the same characteristics too.

The problem of similarity of a numerical field pair is modeled as similarity problem of a set pair. The baseline method for the similarity is the well-known Jaccard similarity which measures the similarity of two given sets.

Fig. 6 shows the Jaccard similarity scores on these ground truth field pairs. Red circle represents matched pairs and blue star represents non-matched pairs. X axis denotes the index of these 30 matched and 30 non-matched field pairs, and Y axis indicates Jaccard similarity score. From this figure we can see that there are about half of positive and negative pairs mixed together from which are difficult to differentiate.

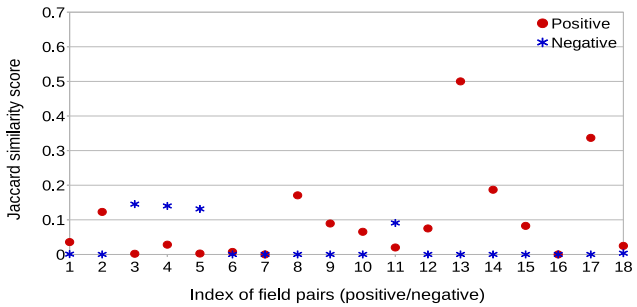

Fig. 6: Matching with Jaccard similarity metric.

[1]BDPS values are obtained from the rest of 24 field pairs after RD similarity metrics is applied with a threshold $Tr = 0.1$ and normalized in $[0.1, 1]$.

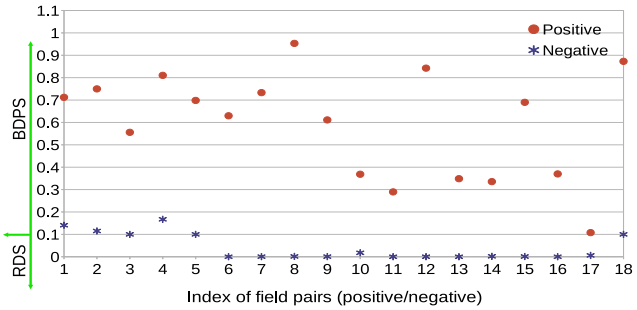

Fig. 7: Matching with combined RDS and BDPS [1].

Fig. 7 shows our numerical field matching algorithm. It shows the applied result on 18 positive and 18 negative field pairs with combined RD similarity and BDP similarity metrics. For RD similarity metric, we use a threshold T_r and rule out the result pairs below T_r as non-matched field pairs. Then we keep the rest of field pairs to BDP similarity metric for further matching. We show this combined RDS and BDPS together for better visualizing the decision boundary of matching. The decision threshold for RDS is $T_r = 0.1$. When $0 < T_r < 0.1$, y-axis shows the RDS. When $T_r \geq 0.1$, it shows the normalized BDPS for the rest of field pairs. There are 12 field pairs which are considered as non-matched pairs and removed with $0 < RDS < 0.1$. The rest 24 field pairs (18 matched pairs and 6 non-matched pairs) with $RDS \geq 0.1$ are easily differentiated with BDP similarity metric. We can see that the BDP result could provide an excellent decision boundary among matched pairs and non-matched pairs in which the final threshold T_b is chosen around 0.2. With this combined RD and BDP similarity metrics, we have greatly improved the result over the Jaccard similarity result.

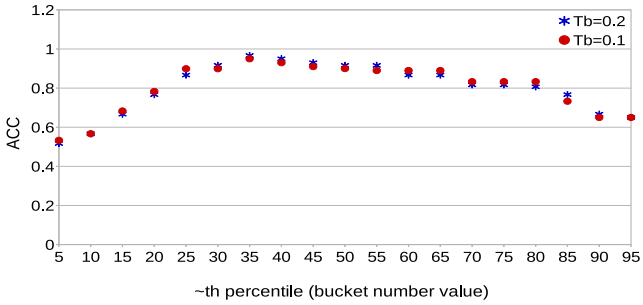

Fig. 8: Selection of the best T_b based on the matching accuracy.

The bucket number b_n is the main factor affecting the BDP similarity and the final results. To determine the optimal b_n value, we use a number of different b_n values to apply on that 60% of ground truth field pairs to evaluate on the accuracies. The b_n values are obtained from different percentiles (5th, 10th,..., 90th and 95th percentile) of record values of each combined field pair when computing BDPS. Fig. 8 shows the accuracy of BDP similarity metric with different percentiles (b_n values) when the threshold T_b is set at 0.1 and 0.2. It shows a similar summit that the accuracy is at an optimal value when the b_n is around from 35th percentile. BDP accuracy goes down when the b_n becomes smaller or bigger. Thus the b_n is selected as 35th percentile for matching other part of our

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2020.3036414, IEEE Transactions on Network and Service Management

dataset in the later experiments.

2) Top-20 Similarity Results: The matching experiment based on all the 679 numerical fields is shown here. Table V shows top-20 matching results. We use RDS threshold $T_r = 0.1$ and bucket number $b_n = 50,000$ for BDP similarity computation. All the rows are accurate matches, which are also confirmed by humans. The accuracy can be up to 100% for the top-20 results, which shows a great potential for our numerical field matching algorithm applied on the large dataset, also reducing lots of human labor for matching. Moreover, with our system-aided matching findings, we can find some pairs matching which are difficult to be found with human annotations such as "changewg_key" and "subregion_key", indicating which regions that the workgroup mainly serves.

To ensure a graph database more meaningful and complete, the selection of top-k results is finally decided by users. Users can observe the top-k results and rule out the unwanted matches as the final input pairs to a graph database so as to introduce less "noisy" connections of the graph database.

Table V: Top-20 similarity result of numerical field pairs

Table.field A	Table.field B	BDP similarity
T_DEFT.deft_key	T_INCI_DE.bl_def_key	0.844
T_OR_LN.item_id	X_PRO.item_id	0.698
T_DEFT.deft_id	T_INCI_DE.defect_id	0.682
T_INCI.item_id	X_PRO.item_id	0.673
T_DEFT.deft_key	T_PROD.item_id	0.64
X_INS.item_id	X_PRO.item_id	0.597
T_PROD.bl_prod_key	**T_SUR.task_key**	**0.567**
T_INCI.changewg_key	T_WK.subregion_key	0.551
T_INCI.changewg_key	T_WK.wkgrp_key	0.551
T_INCI.changewg_key	T_WK.theater_key	0.551
T_INCI.currentwg_key	T_WK.theater_key	0.545
T_INCI.currentwg_key	T_WK.subregion_key	0.545
T_INCI.currentwg_key	T_WK.wkgrp_key	0.545
T_INCI.hwversion_id	T_PROD.bl_prod_key	0.507
T_INCI.createwkgrp_key	T_WK.theater_key	0.507
T_INCI.createwkgrp_key	T_WK.subregion_key	0.507
T_INCI.createwkgrp_key	T_WK.wkgrp_key	0.507
T_PROD.item_id	X_PRO.item_id	0.468
T_INCI.prod_hw_key	T_HW_PROD.bl_prod_key	0.463
T_SUR.evalwkgrp_key	T_WK.wkgrp_key	0.433

D. Evaluation based on Non-numerical Data

We evaluate our NEMA non-numerical algorithms based on TPM and Hashing on the non-numerical data in two parts as well. We use non-numerical ground truth data to evaluate the effectiveness of our algorithms. The matching results of all the other non-numerical field pairs are then described.

1) Evaluating of Ground Truth: There are 20 positive ground truth field pairs which are annotated by humans. Also, 20 negative ground truth field pairs are randomly selected from the dataset and verified to make balanced data instances of positive and negative field pairs. Similarly to numerical evaluation, we randomly select 60% matched field pairs and 60% non-matched field pairs to make balanced ground truth from the whole ground truth field pairs to determine the thresholds of NEMA non-numerical algorithms. The rest 40% will be tested to show the quality in Section IV-F. Part of field pair examples are shown in table VI.

We first analyze the ground truth record-pairs and show the viability for the record pair similarity threshold T_{rn}.

Table VI: Examples of non-numerical ground truth

No.	Table.field A	Table.field B	Matching class
1	T_PROD.item_name	X_INS.item_name	1
2	T_COT.cpr_country	T_SITE.country	1
3	T_CT.temp_desc	T_PROD.item_desc	1
4	T_CT.ctserv_line	X_SAH.servline_name	1
5	T_INCI.curr_wg_name	T_WK.wkgp_name	1
6	T_CT.temp_desc	X_SAH.temp_name	1
7	T_SITE.cust_state	X_SAH.billto_state	1
8	T_CT.temp_name	X_SAH.temp_desc	1
9	T_PROD.prod_family	T_HW_PROD.family	1
10	T_PROD.prod_family	T_HW_PROD.erp_family	1
1	T_INCI.init_gp_name	T_SITE.address	0
2	T_DEFT.deft_submitter	T_SITE.email_addr	0
3	T_COT.cpr_country	T_INCI.summary	0
4	T_SITE.address1	X_PRO.prod_family	0
5	T_PROD.prod_family	X_SAH.hdrcust_name	0
6	T_INCI.tacpica_ct	T_HW_PROD.family	0
7	T_WK.wkgp_desc	X_PRO.physisn_loc	0
8	T_SITE.county	T_SUR.batchcot_name	0
9	T_INCI.customersw_ver	T_SITE.state	0
10	T_OR_LN.partsloc_code	X_INS.item_name	0

Table VII shows the record pair similarity scores of 9 different record pairs in a field pair ("T_PROD.prod_subgrp", "T_HW_PROD.platform"). The first 7 rows of pairs with high similarity scores are matched record pairs. The last 2 rows are not matched record pairs with lower score of 0.333. They have a decision boundary of score around 0.4. Also, based on the database matching standards of prefixing and our experimental observations on ground truth field records, we set $T_{rn} = 0.4$ as the record similarity threshold.

Table VII: Sample of non-numerical record pairs

T_PROD.prod_subgrp	T_HW_PROD.platform	Record similarity
c900 series	c900 series	1
c2950 series	c2916 series	0.8
1601r series	1601 series	0.775
css2950	css2916	0.667
C2960	C2960CX	0.577
C3560CX	C3560X	0.5
AIR35CE	AIR35SE	0.4
ts900	**cs900**	**0.333**
c800	**s800**	**0.333**

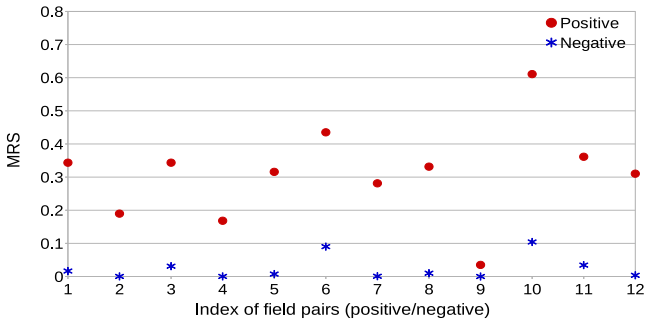

Fig. 9: TPM-based matching ratio score.

We demonstrate the effectiveness of non-numerical algorithms of TPM and MH-LSH by calculating matching ratio score based on TPM and estimated matching score based on MH-LSH on non-numerical ground truth. Fig. 9 shows matching ratio scores of this ground truth data matching in a non-ascending order on TPM. The matching ratio scores of almost all the matched pairs are above the non-matched pairs's. If we use the threshold 0.1 or select top-20 results from this, the accuracy can achieve about 95%, which shows

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2020.3036414, IEEE Transactions on Network and Service Management

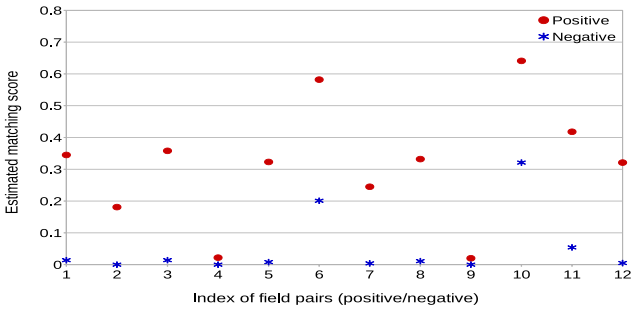

Fig. 10: MH-LSH-based matching score.

the effectiveness of NEMA based on TPM. Fig. 10 shows the matching scores of these ground truth in a non-ascending order based on MH-LSH. Although the decision boundary is not as good as the TPM-based result, the accuracy can achieve 90% when the threshold is 0.1 or top 19 results are selected.

2) Top-20 Similarity Results: There are 779 non-numerical fields in the large dataset. Considering that almost all the primary keys in a table are numerical fields, we do not consider primary key constraint matching method for non-numerical field matching. The record similarity threshold T_{rn} is set to be 0.4 here based on the analysis of Table VII. The final top list of matching ratio scores are obtained based on TPM algorithm from all the non-numerical field pairs.

Table VIII shows the top-20 results of field pair matching based on TPM. We can see that all the field pairs are matched pairs, and they are also confirmed by humans later, which shows the effectiveness of NEMA based on TPM algorithm.

Table VIII: Top-20 matching results of non-numerical field pairs

Table.field1	Table.field2	Matching ratio
T_INCI_I2.currentwg_key	T_WK.wkgp_name	0.637
T_INCI_I2.currentwg_key	T_WK.wkgp_desc	0.632
T_INCI.initwg_name	T_WK.wkgp_name	0.63
T_INCI.initwg_name	T_WK.wkgp_desc	0.628
T_WK.wkgmgr_email	T_SUR.eval_email	0.626
T_INCI.creatorwg_name	T_WK.wkgp_name	0.624
T_INCI.creatorwg_name	T_WK.wkgp_desc	0.622
T_INCI.curr_wg_name	T_WK.wkgp_name	0.611
T_INCI.curr_wg_name	T_WK.wkgp_desc	0.607
X_SAH.billto_state	T_SITE.state	0.504
X_SAH.billto_state	T_SITE.cust_state	0.499
T_INCI.initwg_name	T_INCI_I2.curr_wg_name	0.462
T_INCI.initwg_name	T_INCI_I2.wkgrp_name	0.457
T_COT.cpr_country	T_SITE.country	0.453
T_SITE.cust_country	T_COT.cpr_country	0.453
T_INCI_I2.wkgrp_name	T_INCI.curr_wg_name	0.451
T_COT.cpr_country	T_SITE.cust_country	0.447
T_INCI.curr_wg_name	T_INCI_I2.curr_wg_name	0.442
T_SITE.country	T_COT.cpr_country	0.44
T_HW_PROD.erpplatform	X_SCDC.productsub_grp	0.431

E. Evaluation of SVM Classification-based Matching

1) Extending Ground Truth Data: To address the problem of limited availability of the positive ground truth in our dataset, we refer to a sampling method by H. Kohler [17] to synthesize more positive ground truth field pairs as "synthetically positive field pairs". It preserves the "synchronization property" (to preserve the Jaccard similarity of original sets). Given an original ground truth field pair (A, B), we sample a new field pair based on this original pair. The sampling process is to make sure if a particular record sample e is sampled in A and e is also in B, then it is also sampled from B. If the original ground truth pair is positive, the synthetic field pair is considered as positive as well. We synthesize x (e.g. 100) more field pairs out of each ground truth field pair. Here 20% of records values from each field are sampled. In the original dataset there are 30 positive ground truth field pairs in numerical matching. For each one of the ground truth field pairs, we synthesize 100 field pairs out of it, thus 3,000 synthetically positive field pairs are created. Also, we randomly select 3,000 negative ground truth field pairs from the original dataset. Therefore, we create balanced data instances of 6,000 field pairs where 3,000 positive field pairs and 3,000 negative field pairs for classification. Similarly, for non-numerical fields, we create data instances of 6,000 field pairs where 3,000 synthetically positive field pairs from the 20 positive field pairs and randomly select 3,000 negative ground truth field pairs as the classification dataset.

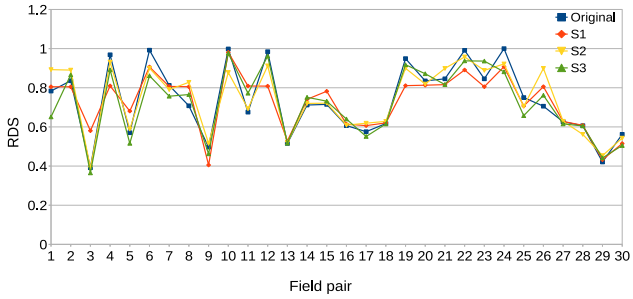

Fig. 11: RDS values for original and synthetic field pairs in numerical fields.

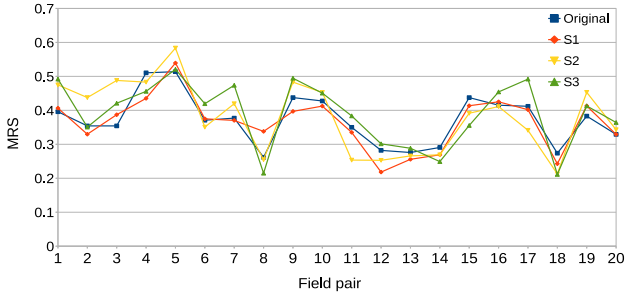

Fig. 12: MRS values for original and synthetic field pairs in non-numerical fields.

We show some comparisons of original ground truth dataset and synthetic dataset here. There are 30 positive ground truth field pairs from original dataset for numerical fields and 20 positive ground truth field pairs for non-numerical fields. For each field pair we compare with 3 randomly selected synthetic field pairs labeled as S1, S2, and S3 respectively. Fig. 11 shows the comparison of range difference score (RDS) for these synthetic dataset with original dataset in numerical field matching. Fig. 12 shows the comparison of these matching ratio score (MRS) for these synthetic dataset with original dataset in non-numerical field matching. Both results show that synthetic field pairs have similar scores with original field pairs. It further implies that synthetic dataset has similar range and distribution with the original dataset.

2) SVM Classification Results: Here we show the experimental results of numerical and non-numerical field matching with cross-validation and testing. For each classification, we randomly split our large synthetic dataset into 80% as training dataset, and 20% as test dataset according to the common splitting ratio based on the Pareto principle [34]. In the training stage, 5-fold cross-validation is used for validation. We run the same procedure of the whole experiment 20 times and obtain the average measures of precision, recall, ACC and F1 for validation and testing shown in the Table IX. It shows our SVM classification-based matching has high and similar performance compared to previous similarity metric-based algorithms.

Table IX: SVM validation and testing results on synthetic numerical and non-numerical ground truth

Fields	Data	Precision	Recall	ACC	F1
Numerical	Validation	0.997	0.994	0.995	0.995
	Testing (20%)	0.973	0.972	0.972	0.972
Non -numerical	Validation	0.949	0.958	0.953	0.953
	Testing (20%)	0.938	0.955	0.946	0.947

F. Comparisons with Other Existing Algorithms

We compare our technique NEMA with other existing algorithms-COMA system [12] and rule-based Regex [21] here. COMA is a state-of-the-art and popular hybrid matching tool and system supporting both schema-level and instance-level matching. Regex is an instance-level rule-based matching method based on regular expressions. We test and compare their matching results on the rest 40% of numerical and non-numerical ground truth field matching, respectively.

1) Comparison of quality: We measure the quality of precision, recall, ACC, and F1 and compare the COMA in the schema level and instance level and Regex in the instance level for numerical and non-numerical data matching. On the schema level matching, COMA uses the best field matching similarity "0" (which has no corresponding line in the COMA system) as a threshold in the schema-level matching. On the instance level matching, COMA has one similar instance-level matching that uses aggregated maximum record-wise similarities to obtain the final field pair similarities. The record-wise similarity is based on common similarity metrics such as edit distance [28] and trigram [2]. Edit distance is to measure how dissimilar two strings are to one another by counting the minimum number of operations required to transform one string into the other. Trigram is to split a string into triples of characters and comparing those to the trigrams of another string. The field matching similarity between two fields A and B in COMA is defined as follows:

$$sim(A,B) = \frac{1}{m+n} \cdot (\sum_{i=1}^{m} max_{j=1\ldots,n}(sim(a_i,b_j)) + \sum_{j=1}^{n} max_{i=1\ldots,m}(sim(b_j,a_i))) \quad (10)$$

Regex is a matching method based on regular expression by creating patterns from sampling instances of one field and then match against instances of another field to decide matching.

Table. X shows the quality comparisons among COMA, NEMA and Regex. COMA-SCH is COMA with schema-level matching algorithm. COMA-ED and COMA-TRG means that COMA uses edit distance and trigram to measure the record similarity in instance-level matching, respectively. Regex is the matching based on regular expression. NEMA-(RD+BDP) is NEMA using combined RD and BDP in numerical field matching, while NEMA-TPM is NEMA using TPM in non-numerical field matching. NEMA-MH_LSH indicates that NEMA uses minHash-locality sensitive hashing in non-numerical field matching. The accuracies and F1 scores of NEMA-(RD+BDP) and NEMA-TPM in numerical and non-numerical data matching can be up to 95%, as high as COMA's non-numerical data matching, but having better performance than COMA-SCH and Regex matching. For numerical field matching, the accuracies of COMA-ED, COMA-TRG and Regex are 7%-15% lower than NEMA-(RD+BDP) because of the ineffectiveness to identify non-matched pairs of numerical ground truth. For non-numerical field matching, the highest accuracies and F1-scores of COMA-ED and COMA-TRG are only 1-2% higher than NEMA-TPM. However, the field matching score of COMA is measured based on its general string similarity matching, which is not well applied to the network management database matching for record pair similarity requirements. A large number of pairs with high record similarities in COMA are not thought of as matches in the network management databases, which shows the usefulness of the NEMA non-numerical algorithm.

Table X: Quality comparisons with other existing algorithms

Field	Algorithms	Precision	Recall	ACC	F1
Numerical	COMA-SCH	0.867	0.838	0.85	0.852
	COMA-ED	0.848	0.933	0.883	0.889
	COMA-TRG	0.765	0.867	0.8	0.813
	Regex	0.833	0.833	0.833	0..833
	NEMA-(RD+BDP)	0.966	**0.933**	**0.95**	**0.949**
Non -numerical	COMA-SCH	0.781	0.833	0.8	0.806
	COMA-ED	1.0	0.933	**0.967**	0.966
	COMA-TRG	**0.967**	**0.967**	**0.967**	**0.967**
	Regex	0.867	0.897	0.883	0.881
	NEMA-TPM	0.936	**0.967**	0.95	0.951
	NEMA-MH_LSH	0.933	0.848	0.883	0.889

Table XI: Efficiency comparison with other existing algorithms

Non-numerical Algorithms	Time Mean (s)	Time SD (s)
COMA-SCH	**251**	21
COMA-ED	13,322	1,289
COMA-TRG	12,890	1,329
Regex	425	89
NEMA-TPM	**2,832**	159
NEMA-MH_LSH	**939**	62

2) Comparisons of Mismatched Examples: We further analyze the differences of COMA and NEMA in matching the ground truth field pairs. Table XII shows the field pairs in every row and its similarity scores by COMA and NEMA. These field pairs are found to be matched pairs by NEMA with relatively high similarity scores, but COMA shows no similarities with score 0. For COMA, the field names have very few common characters in spelling, even though the semantic commonality exists. NEMA does not rely on the inaccurate

schema-level properties, but it uses the record instance for the decisions of field matching, which indirectly considers the semantic correspondences. If the record instances for some matched pairs are incomplete or missing, however, the similarity scores for these field pairs are also low. Table XIII shows the two field pairs that have low similarities in NEMA. Although the field names in each pair express the same thing semantically, the record instances in the fields are actually incomplete and have very few in common between each other. However, in our databases, the missing or incomplete field pairs are very few compared to the large number of field pairs, which does not affect the overall performance for the numerical field matching with NEMA.

Table XII: Example of field pairs matched by NEMA, but not by COMA

Table.fieldA	Table.fieldB	COMA-ED	NEMA-TPM
T_INCI.ins_site_key	T_SITE.partysite_id	0	0.208
T_OR_HD.creator_id	T_INCI.lastup_by	0	0.643
T_CT.temp_desc	T_PROD.item_desc	0	0.05

Table XIII: Example of field pairs matched by COMA, but not by NEMA

Table.fieldA	Table.fieldB	COMA-ED	NEMA-TPM
T_INCI.bl_cot_key	T_CT.bl_cot_key	0.750	0.091
T_SUR.bl_surv_key	T_SUR_ANS.bl_surv_key	0.76	0.009

Here we analyze the specific record pair examples of non-numerical instance-level matching. COMA uses standard edit distance and trigram to calculate the similarities of records, which is not quite suitable for the matching requirement of network management databases. Table XIV below shows 9 examples of records pairs and three different kinds of similarities (NEMA-TPM, COMA-ED, COMA-TRG). The first 7 rows as one group are thought of as matched record pairs, the last two rows in the other group are non-matched record pairs. We can see from that the similarity of matched pairs based on COMA are quite similar around 0.7 for these two groups, from which is not easy to differentiate. While the similarities by NEMA have good differences (0.333 for non-matched pairs, 0.4 above for matched pairs). This further demonstrates that NEMA is more suitable for the network database matching.

Table XIV: Examples of record similarity comparisons

Record 1	Record 2	NEMA-TPM	COMA-ED	COMA-TRG
c900 series	c900 series	1	1	1
c2950 series	c2916 series	0.8	0.833	0.6
1601r series	1601 series	0.775	0.909	0.738
css2950	css2916	0.667	0.714	0.6
C2960	C2960CX	0.577	0.6	0.775
C3560CX	C3560X	0.5	0.833	0.671
AIR35CE	AIR35SE	0.4	0.857	0.6
c800	**s800**	**0.333**	**0.75**	**0.5**
ts 900	**cs900**	**0.333**	**0.8**	**0.667**

3) Comparison of Efficiency: Considering the expensive time consumption for non-numerical field pair matching, we test the efficiency based on the whole non-numerical ground truth. We run the experiment 20 times on the same machine with the same data to calculate the average computation time and standard deviation (SD) without data loading time. Table XI shows the total computation time spent for COMA-SCH, COMA-ED, COMA-TRG, Regex, NEMA-TPM and NEMA-MH_LSH. Among them, COMA-SCH and Regex are two fastest among all the algorithms, but the accuracies are the lowest. COMA-ED is the slowest, taking about 13,322 seconds. NEMA-TPM takes 2,832 seconds, about 5x speedup over COMA-ED. NEMA-MH_LSH takes 939 seconds, which is about 14 times faster than COMA-ED in the cost of 8% accuracy lost. NEMA-TPM is slower than COMA-SCH and Regex algorithms, but with about 8% higher accuracy. Therefore, NEMA-TPM outperforms most of other existing algorithms and reaches the best trade-off of quality and efficiency among them.

V. Related Work

The structured data matching is an old and important research topic but unsolved and ever-growing problem, which has a wide range of applications in database integration, migration, semantic query, etc. [6]. In the survey paper [30], the authors propose a solution taxonomy differentiating between element and structure level, schema and instance level, language and constraint-based matching techniques. Furthermore, P. Shivaiko et al. [31] review the state-of-the-art matching systems which were based on strings, structure, data instance and semantics matching techniques using different schema formats such as database, XML, OWL, RDFS, etc. In database schema matching, previous common matching systems in schema-level are introduced in several prototypes such as Similarity Flooding (SF) [22], Coma [3], etc.

SF [22] is a matching algorithm that models two structured columns to be compared as two directed labeled graphs. It makes use of field data, key properties and the string-based alignment (prefix and suffix test) to obtain the alignments between two nodes of the graph. The similarity is calculated from similar nodes to adjacent neighbors through propagation. Our NEMA only relies on the data instance values to infer the matching of fields, which does not utilize the structured properties and data types. However, SF uses a metric for matching quality based on the intended matching results, which is similar to our accuracy metric based on top-k results.

Coma [3] is a composite matching system providing extensible library and framework for combining obtained results. It contains mainly 6 elementary matchers using string-based techniques, 5 hybrid matchers using names and structural paths, and one reuse-oriented matcher based on previous matching results . The composite matcher effectively improves the match quality over a single matcher using the default combination strategy. Compared to SF, the overall average matching quality are the best among them [3]. The extended version Coma++ [12] utilizes the shared taxonomy and pivot schema to further improve the overall matching quality. In our evaluation, we compare with the Coma++ method using the default combination strategy and find our technique NEMA overall outperforms than COMA in schema-level matching.

Except from the previous matching approaches using field and structural information matching, data instanced-based

approaches [24], [20], [9], [36] use the similarity metric or machine learning or rule-based methods to determine the similarity of fields. In [20], the authors utilize a corpus that contains schema and mappings between some schema pairs, and learn the constraints from schema statistics to help generate more matching pairs. In [9], the authors use the mutual information of statistics to measures the similarity of schema instances between two columns to decide the matching, which shows an effective method based on instances. Also, the authors in [26] propose a new sample-driven approach which enables the end-users to easily construct their own data to match the source and target schema. [21] proposes a rule-based method by creating regular expression pattern to match against columns. [36] uses matching-learning technique of training neural networks for getting candidate pairs and then filters the pairs with a rule-based algorithm. Our NEMA uses proposed similarity metrics as features to train a SVM to classify for matching effectively and efficiently. COMA [12] proposes two instance-level matching methods based on the constraint of instance data and the content-based matching to measure field matching. The constraint-based method relies on the general, numerical and pattern constraint which has specific limitation to the specific data which is not suitable for the network databases. The content-based matching depends on the aggregation of similarity scores of instance contents and it is kind of similar to our NEMA technique on content-based similarity measurement.

To sum up, most of currently popular matching approaches and systems focus on schema-level information matching. The data instances level matching approaches using field record values are mostly based on some statistical models and machine learning from corpus. We further explore the database instance matching by comparing field records using different metrics and propose effective and overall matching algorithms considering the characteristic of network database matching for a graph database construction for efficient data query, analysis and management.

VI. Conclusion

In this paper we propose a systematic technique NEMA to match databases for network management. Different from previous database matching approaches, we design a technique to match numerical and non-numerical fields in instance-level respectively, which can effectively be integrated into a graph database for network management and analysis. For numerical field matching, we propose range difference similarity and bucket dot product similarity metrics. For non-numerical field matching, we design top priority match metric and also propose applying minHash-locality sensitive hashing algorithm, which reduces the matching time for large databases. To address the drawback of manual thresholds, an effective classification-based method is also proposed based on the proposed similarity metrics. NEMA are experimentally demonstrated with best trade-off of qualify and efficiency among other existing algorithms.

With the explosion of big data and popularity of distributed graph processing systems, this work has the potential to significantly reduce the human work involving identifying the matching fields for a large graph database construction and also be applied for large-scale data matching. A majority of partial matching pairs can be found by our matching algorithms which are not easily detected by humans.

VII. Discussion and Future Work

This work primarily discusses the big database integration and lays the foundation for network management on a graph database. It has direct implications for network management to help network operators/administrators with network query, network diagnosis, fault detection, network performance monitoring, etc. One specific example for network query is that it is more efficient to use graph traversal algorithms to find out which network routers communicated with CiscoASR9010 have the most frequent incidents in the last year. Another example is that if a network ticket/incident occurs, with the help of graph clustering or propagation models, the administrators could easily locate the network failure and other affected networks, and analyze its root cause. The system can also automatically suggest a potential solution to the network failure based on previous histories of tickets/incidents. In the future, we would like to explore two areas (1) investigate existing graph databases such as Neo4j [23] or develop a graph database, and integrate the matching results into it to build a network analytics and management system; (2) research and explore real network field problems in the area of network query, analysis, prediction, security, etc. on the system.

References

[1] C. C. Aggarwal and H. Wang. Graph data management and mining: A survey of algorithms and applications. In *Managing and mining graph data*, pages 13–68. Springer, 2010.

[2] R. C. Angell, G. E. Freund, and P. Willett. Automatic spelling correction using a trigram similarity measure. *Information Processing & Management*, 19(4):255–261, 1983.

[3] D. Aumueller, H.-H. Do, S. Massmann, and E. Rahm. Schema and ontology matching with coma++. In *Proceedings of the 2005 ACM SIGMOD international conference on Management of data*, pages 906–908. ACM, 2005.

[4] A. Bhattacharjee and H. Jamil. Ontomatch: A monotonically improving schema matching system for autonomous data integration. In *Information Reuse & Integration, 2009. IRI'09. IEEE International Conference on*, pages 318–323. IEEE, 2009.

[5] M. Bilenko, R. Mooney, W. Cohen, P. Ravikumar, and S. Fienberg. Adaptive name matching in information integration. *IEEE Intelligent Systems*, 18(5):16–23, 2003.

[6] S. Castano, A. Ferrara, and S. Montanelli. Matching techniques for data integration and exploration: From databases to big data. In *A Comprehensive Guide Through the Italian Database Research Over the Last 25 Years*, pages 61–76. Springer, 2018.

[7] M. Cheatham and P. Hitzler. String similarity metrics for ontology alignment. In *The Semantic Web–ISWC 2013*, pages 294–309. Springer, 2013.

[8] C. Chen, B. Golshan, A. Y. Halevy, W.-C. Tan, and A. Doan. Biggorilla: An open-source ecosystem for data preparation and integration. *IEEE Data Eng. Bull.*, 41(2):10–22, 2018.

[9] W. Chen, H. Guo, F. Zhang, X. Pu, and X. Liu. Mining schema matching between heterogeneous databases. In *Consumer Electronics, Communications and Networks (CECNet), 2012 2nd International Conference on*, pages 1128–1131. IEEE, 2012.

[10] C. Coronel and S. Morris. *Database systems: design, implementation, & management*. Cengage Learning, 2016.

[11] C. Daraio, M. Lenzerini, C. Leporelli, P. Naggar, A. Bonaccorsi, and A. Bartolucci. The advantages of an ontology-based data management approach: openness, interoperability and data quality. *Scientometrics*, pages 1–15, 2016.

[12] D. Engmann and S. Massmann. Instance matching with coma+. 2007.

[13] D. Gomez-Cabrero, I. Abugessaisa, D. Maier, A. Teschendorff, M. Merkenschlager, A. Gisel, E. Ballestar, E. Bongcam-Rudloff, A. Conesa, and J. Tegnér. Data integration in the era of omics: current and future challenges, 2014.

[14] B. Gu, Z. Li, X. Zhang, A. Liu, G. Liu, K. Zheng, L. Zhao, and X. Zhou. The interaction between schema matching and record matching in data integration. *IEEE Transactions on Knowledge and Data Engineering*, 29(1):186–199, 2016.

[15] D. S. Himmelstein and S. E. Baranzini. Heterogeneous network edge prediction: a data integration approach to prioritize disease-associated genes. *PLoS computational biology*, 11(7):e1004259, 2015.

[16] D. Jang, T. Kim, and H. Kim. History management for network information of iot devices. In *International Conference on Security with Intelligent Computing and Big-data Services*, pages 138–145. Springer, 2017.

[17] H. Köhler, X. Zhou, S. Sadiq, Y. Shu, and K. Taylor. Sampling dirty data for matching attributes. In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of data*, pages 63–74, 2010.

[18] J. Leskovec, A. Rajaraman, and J. D. Ullman. *Mining of massive datasets*. Cambridge University Press, 2014.

[19] T. Lose, P. van Heusden, and A. Christoffels. Combat-tb-neodb: fostering tuberculosis research through integrative analysis using graph database technologies. *Bioinformatics*, 2019.

[20] J. Madhavan, P. A. Bernstein, A. Doan, and A. Halevy. Corpus-based schema matching. In *21st International Conference on Data Engineering (ICDE'05)*, pages 57–68. IEEE, 2005.

[21] O. A. Mehdi, H. Ibrahim, and L. S. Affendey. Instance based matching using regular expression. *Procedia Computer Science*, 10:688–695, 2012.

[22] S. Melnik, H. Garcia-Molina, and E. Rahm. Similarity flooding: A versatile graph matching algorithm and its application to schema matching. In *Data Engineering, 2002. Proceedings. 18th International Conference on*, pages 117–128. IEEE, 2002.

[23] J. J. Miller. Graph database applications and concepts with neo4j. In *Proceedings of the Southern Association for Information Systems Conference, Atlanta, GA, USA*, volume 2324, 2013.

[24] H. Nottelmann and U. Straccia. Information retrieval and machine learning for probabilistic schema matching. *Information processing & management*, 43(3):552–576, 2007.

[25] G. Pantuza, F. Sampaio, L. F. Vieira, D. Guedes, and M. A. Vieira. Network management through graphs in software defined networks. In *10th International Conference on Network and Service Management (CNSM) and Workshop*, pages 400–405. IEEE, 2014.

[26] L. Qian, M. J. Cafarella, and H. Jagadish. Sample-driven schema mapping. In *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*, pages 73–84. ACM, 2012.

[27] A. D. Raut and M. Atique. A survey of indexing techniques for xml database. *Compusoft*, 3(1):461, 2014.

[28] E. S. Ristad and P. N. Yianilos. Learning string-edit distance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(5):522–532, 1998.

[29] S. Sahu, A. Mhedhbi, S. Salihoglu, J. Lin, and M. T. Özsu. The ubiquity of large graphs and surprising challenges of graph processing. *Proceedings of the VLDB Endowment*, 11(4):420–431, 2017.

[30] P. Shvaiko and J. Euzenat. A survey of schema-based matching approaches. In *Journal on data semantics IV*, pages 146–171. Springer, 2005.

[31] P. Shvaiko and J. Euzenat. Ontology matching: state of the art and future challenges. *IEEE Transactions on knowledge and data engineering*, 25(1):158–176, 2013.

[32] M. Stonebraker and I. F. Ilyas. Data integration: The current status and the way forward. *IEEE Data Eng. Bull.*, 41(2):3–9, 2018.

[33] N. Sünderhauf, T. T. Pham, Y. Latif, M. Milford, and I. Reid. Meaningful maps with object-oriented semantic mapping. In *Intelligent Robots and Systems (IROS), 2017 IEEE/RSJ International Conference on*, pages 5079–5085. IEEE, 2017.

[34] A. Ultsch. Pareto density estimation: a density estimation for knowledge discovery. In *Innovations in classification, data science, and information systems*, pages 91–100. Springer, 2005.

[35] J. Wang, Y. Wu, N. Yen, S. Guo, and Z. Cheng. Big data analytics for emergency communication networks: A survey. *IEEE Communications Surveys & Tutorials*, 18(3):1758–1778, 2016.

[36] Y. Yang, M. Chen, and B. Gao. An effective content-based schema matching algorithm. In *2008 International Seminar on Future Information Technology and Management Engineering*, pages 7–11. IEEE, 2008.

Fubao Wu received his Bachelor degree in Electronic Information Engineering from Northeastern University, China and Master of Engineering in Electronic Science and Technology from University of Science and Technology of China in 2011. He is currently pursuing his Ph.D. degree in Electrical and Computer Engineering at University of Massachusetts Amherst. His research interests are data analytics, graph analytics and video analytics.

Han Hee Song is a Research Scientist at Apple and leads development of deep-learned training and online serving pipelines on distributed platforms. He was a Lead Data Scientist at Cisco Systems and Senior Member of Technical Staff at CTO office of Narus, Inc. His research focuses on big data mining on mobile users and on-device machine learning for privacy-preserving personalization. He received his Ph.D. degree in Computer Science from the University of Texas at Austin in 2011.

Jiangtao Yin received the Ph.D. degree in Electrical and Computer Engineering from the University of Massachusetts at Amherst, in 2016. He is currently working at Palo Alto Networks. His current research interests include big data, cloud computing, distributed systems, large-scale data processing, scalable data storage, and stream processing.

Lixin Gao received the Ph.D. degree in computer science from the University of Massachusetts Amherst in 1996. Now she is a professor of electrical and computer engineering with the University of Massachusetts at Amherst. Her research interests include social networks, Internet routing, network virtualization and cloud computing. Between May 1999 and January 2000, she was a visiting researcher in AT&T Research Labs and DIMACS. She was an Alfred P. Sloan fellow between 2003-2005 and received an NSF CAREER Award in 1999. She won the best paper award from IEEE INFOCOM 2010 and ACM SoCC 2011, and the test-of-time award in ACM SIGMETRICS 2010. She received the Chancellors Award for Outstanding Accomplishment in Research and Creative Activity in 2010. She is a fellow of the IEEE and ACM.

Mario Baldi is a Distinguished Technologist at Pensando Systems, Inc. and Associate Professor at Politecnico di Torino. He was Director of Technology at Cisco Systems, Data Scientist Director at Symantec Corp., Inc., Principal Member of Technical Staff at Narus, Inc., Principal Architect at Embrane, Inc.; Vice Dean of the PoliTong Sino-Italian Campus at Tongji University, Shanghai; Vice President for Protocol Architecture at Synchrodyne Networks, Inc., New York. His research, teaching and professional activities have covered data plane programmability, big data analytics, next generation network data analysis, internetworking, high performance switching, optical networking, quality of service, multimedia networking, trust in distributed software execution.

Narendra Anand is R&D Technology Associate Principal at Accenture. He worked as a Research Scientist at Cisco Systems. He received his Ph.D. degree in May of 2015 at Rice University in the Department of Electrical and Computer Engineering. His research focuses on the design and implementation of novel, cross-layer, Multi-user Multiple-Input-Multiple-Output (MU-MIMO) communication protocols.