NEMA: Automatic Integration of Network Management Databases

Fubao Wu University of Massachusetts, Amherst
fubaowu@umass.edu

Han Hee Song
Cisco Systems, Inc
hanhsong@cisco.com

Jiangtao Yin
Cisco Systems, Inc
jiangyin@cisco.com

Lixin Gao University of Massachusetts, Amherst
lgao@engin.umass.edu

Mario Baldi
Cisco Systems, Inc
mariobal@cisco.com

Narendra Anand
Cisco Systems, Inc
nareanan@cisco.com

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 large network management databases matching. 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 the accuracy of NEMA on both numerical and non-numerical instances are up to 95%.

1. INTRODUCTION

Big network vendors own different sources of databases for plenty of different customers about network products, configurations, sites and incidents, etc. These databases are usually separated from each other and managed by different department and groups. Network data management usually involves interoperability and sharing these different databases together for efficient network prediction, semantic query, network analysis and fault detection, etc. To automatic and efficient identify connections of these databases, finding field matching is a preliminary and crucial step[7]. We aim to construct matching for efficient network management and analysis. For example, with the data connections, we can model a graph databases to query related product configurations and performance issues around a fault or incident, which are matched with another product-related tables. Moreover, we can standardize the expression of converting the semantic description of same type of products (such as 9000, 9002, 9003 and so on) as the matched product family (such as 9k or 900k) for device and incident management.

Ontology matching plays a key role in completing almost all modern knowledge-based graph systems. There is an abundance of research on schema matching approaches for different use cases and data formats, such as relational databases, XML, and object-oriented data formats. Existing database matching approaches include two main types of algorithms. One type is based on schema-level matching, which exploits metadata using schema characteristics such as field names, data types, structural properties, and other schema information [9, 15, 4, 14]. When the data name and structure are not consistently constructed, the method for this is not quite reliable. The other type is instance-level matching, which uses the values of two fields to calculate the similarity and determine matched fields [13, 11, 6, 19, 2, 17]. Most of them rely on the dictionary, corpus and machine learning techniques to learn schema matching. However, characteristics of network databases show noisy and heterogeneous data that is difficult to match with the schema-level information. Also, it is not reliable to construct dictionaries, corpus and trained texts with random naming conventions from large number of product series, product families and abbreviated descriptions. We consider the record instances in the network databases based on the network characteristics to match databases and construct the knowledge graph.

In this paper we consider the characteristics of structured network databases and propose a matching algorithm NEMA to conduct field matching in an instance-level. The goal is to significantly reduce manual labor required in database

mapping to construct a knowledge graph to enable use case queries and data analytics.

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 field name similarity to match data directly and easily.

2. The data is noisy and irregular. Some table fields contain unexpected records such as null, invalid value and typos, etc. 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 format.

4. No thesauri or auxiliary information exists that we can rely on for matching. The only set of observation is the data itself.

To solve these challenges, we propose a systematic approach to conduct the field matching and transfer the matched records into a graph processing system (e.g. Neo4j graph database [23]). Specifically, according to the heterogeneous characteristics of structured data, we divide the data into numerical and non-numerical parts, then conduct matching respectively. For numerical data, we develop column-wise matching metrics including range difference similarity metric and bucket dot product similarity as a final matching score to measure the correlation of two fields. For non-numerical data, we develop record-based matching based on the proposed similarity metric and get the matching ratios for column pairs as final matching scores to correlate the possible columns.

The contributions of our work are as follows:

- We develop a systematic algorithm for structured network database matching by considering the network database characteristics without any prior information, which also hugely reduces human labor.

- We propose an effective pruning method for reduction of search space in large-scale non-numerical field matching.

- We implement and experimentally evaluate our algorithm NEMA on a very large network databases containing 1458 fields with each 10 million records on the average. We find that NEMA can achieve up to 95% matching accuracy for the network databases.

The rest of this paper is organized as follows. We define the problem in Section 2. Section 3 describes the algorithm in detail including both numerical matching and non-numerical matching algorithms. Experimental evaluation is analyzed in Section 4. Section 5 describes the related work. We conclude our paper in Section 6.

2. PROBLEM DESCRIPTION

Given the structured network databases, our goal is to construct a knowledge graph from these table data by identifying matched field pairs. Each table in a database source has a large number of fields and every field contains huge amounts of records. Our goal is to find more accurate and meaningful matched field pairs among different tables. The determination of matching of two fields is based on their similarity measured by record pairs matching. Then the knowledge graph can be created by the matched records as vertices and the correlations between record pairs as edges.

To illustrate our problem and algorithm clearly, we use three sample tables below as a toy example throughout the rest of this paper. In Table 1, PRODUCT Table (T_P) contains 2 fields *product_id*, and *family* with 7 records respectively. In Table 2, INCIDENT Table (T_I) contains 2 fields *incident_key*, *prod_key* with 7 records respectively. In Table 3, ORDER Table (T_O) contains 3 fields *order_key*, *incident_id*, *product_name* with 7 records respectively. The problem is to find table T_P, T_I and T_O matching by evaluating whether these 7 fields match based on the matching results of their record values, then to construct correlations from them for knowledge graphs.

Given network database sources, we aim to find table matching between any two tables by identifying the field matching between any two fields. The field matching is decided by the record matching between their field records. The three types of matchings are defined as follows.

Table 1: 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 2: INCIDENT (T_I)

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

Table 3: 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: A record here is a specific ontology instance used for comparing. Given two instances e1 and e2, a record matching function is defined as a 4-uple: $< e1, e2, v, r >$ where e1 and e2 are table record of the first and the second ontologies, respectively; v is a confidence measure value (i.e. similarity score, typically in the [0, 1] range) holding for the correspondence between e1 and e2; r is a relation (e.g., equivalence, part-of, subsumed, overlapping, etc.) holding between e1 and e2. The matching function $< e1, e2, v, r >$ asserts that the relation r holds between the ontology record e1 and e2 with confidence v. The higher the confidence v, the higher the likelihood that this relation holds. For example, the records in *product_id* field in table 1 and *prod_key* field in table 2 are ontology instances heres. These two fields have common records "107", "108", "109", "150", "151", "152" which have equivalence with similarity score as high as 1, respectively. The record "con5122" in *product_name* field and the record "con5100" in *family* field

have high similarity score with overlap relationship. If one record pair has similarity score above a certain threshold, it is correlated and matched. A matched pair is called the matched record pair, and it is called the non-matched record pair if the pair is not matched.

Field Matching: Given two fields f_1 and f_2, we define field similarity as a function $s = sim(f_1, f_2)$ where sim is the similarity function and s is the similarity score. We define the correlation as the similarity score s above a certain threshold T.

$$match(f_1, f_2) = \begin{cases} True & sim(f_1, f_2) \geqslant T \\ False & otherwise \end{cases} \quad (1)$$

Here $sim(f_1, f_2)$ is the similarity score (e.g. Jaccard similarity score) function between two fields f_1 and f_2, which consider all their record similarities (details will be covered in the next section). If $match(f_1, f_2)$ is true, we call $< f_1, f_2 >$ a matched field pair, otherwise it is called a non-matched field pair. Matched field pairs are used finally for graph edge correlation. For example, in the data example, the $simm(product\_id, prod\_key)$ has high similarity score above a certain threshold, so $< product\_id, prod\_key >$ is correlated and matched. Also, the field pair $< incident\_key$, $incident\_id >$ is also matched. This could be extended to table matching which is the problem we need to solve for constructing knowledge graph.

Table Matching: Given any two tables T1 and T2, which include different fields f_i and f_j respectively, a collection of all the possible matched fields between these two tables is denoted as Tmatch(T1, T2).

$$Tmatch(T1, T2) = \{< fi, fj > | match(fi, fj) \text{ is true for } fi \in T1 \text{ and } fj \in T2\} \quad (2)$$

The $Tmatch(T1, T2)$ is the set of all the matched field pairs between these two tables T1 and T2. In our example, $< product\_id, prod\_key >$ is the set between tables T_P and T_I, $< incident\_key, incident\_id >$ is the set between tables T_I and T_O, and $< product\_name, family >$ are the set between tables T_P and T_O. If there are n different table sources, we need to do the maximum $\binom{n}{2}$ comparisons in the field matching level.

With the match results of table matching, we could construct a knowledge graph, which is the goal for the network database matching.

Graph Model: We construct our data graph using a common property graph model [16] which is supported by most graph processing systems. We define our knowledge graph as attributed, labeled and undirected multi-graph $G = (V, E, L\_v, L\_e)$ where V is the node sets, E is the edge set, L_v is the label and property of node sets V. L_e is the label and property of node sets E. Specifically, in our graph, V contains all the records appearing in the fields which match. E contains the associations between records which are equal in numerical record matching or similar in non-numerical record matching. L_v is the record attribute in the same column, L_e denote the relationship of two records from two table's column attributes. If some records do not correlate with any other records between two matched fields, the record vertices would be isolated. Figure 1 shows an example of the knowledge graph constructed from the three table matching results.

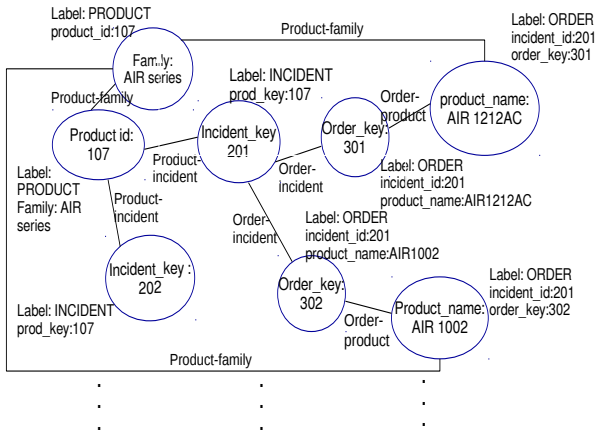

Figure 1: Knowledge graph from toy example matching results

3. 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. Also, the network databases comprise of numerical and non-numerical fields, and they have different attributes and matching requirements. For numerical matching, we consider equivalence relationship between record pairs. For non-numerical matching, however, we do not directly consider the equivalence relationship as matching standard. For example, the non-numerical fields $family$ and $product\_name$ in the example table T_P and T_O have very few common characters on their names, but they are actually correlated. Also, the field records "cisco0510" and "cisco0500" in these two fields should be considered to belong to the same family and be matched pair with high similarity. However, the record pair "47-7066C" and "80-7066C" are considered to be non-matched, even though the two strings have many common characters. Hence, considering the characteristic of network databases, we make use of the record matching to decide whether two fields match. Therefore, we match numerical and non-numerical field separately and design different matching algorithms for each of them.

3.1 System Overview

The system overview is shown in Figure 2. We divide structured data into numerical and non-numerical data parts. In each part, we develop an independent matching approach for table matching. Matching approaches for numerical and non-numerical data are quite different, which will be introduced in different parts. The results of each part are combined together to load into the graph processing system.

3.2 Numerical Matching

Numerical fields are table fields with records which are numerical values. For example, the $incident\_key$ and $prod\_key$ in Table T_I are numerical fields. The field record values is the basis of similarity metric of fields. We define each numerical field record values as a set. This is transfered to a set similarity problem.

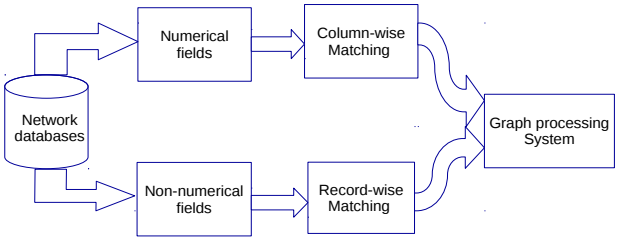

Figure 2: System overview of data matching for graph

There are some common methods on set similarity including Jaccard index, Dice index, Hamming distance, cosine similarity [5], etc. However, it is not practical to just use one simple method to get the decision bound for matching because of the noisiness and complexity of the structured data. We propose a synthetic approach to get the decision tree to determine whether two fields are matched. The numerical matching approach is shown in Figure 3.

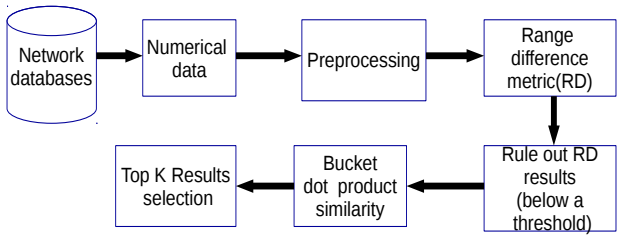

Figure 3: Numerical matching flow

The process of this approach is 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 have got 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, the bucket dot product 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 values above a certain threshold T_b will be selected for input to graph processing system.

3.2.1 Preprocessing

To deal with the irregular and noisy data, we do some preprocessing of them before the formal similarity calculations. This includes removing null values, negative values and any non-numerical values and considering unique values only. We do not consider negative values because they are useless and noisy data in real dataset. Almost all the table fields are about id, key and numbers which should potentially be matched.

3.2.2 Range Difference Similarity Metric

Considering the noisy and sparse characteristics of data, Jaccard similarity coefficient 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. Because for some field pairs, their numbers of common values might be close and the Jaccard similarity values are very close between each other, but the their distribution of range is quite different, which is possible to be not matched in most cases. Therefore, we propose the range difference similarity metric to measure the distribution of these field pairs first. In addition, using range difference similarity metric first, we can prune lots of unwanted computations which would also hugely reduce time consumptions for further matching.

Given two set x and set y, we calculate their different percentiles x_i, y_i (i is the 10, 20, 30...90 percentile). The range difference score D_i for each percentile i is

$$D_i = \frac{|x_i - y_i|}{|x_i + y_i|} \tag{3}$$

We use 20, 30, 80, 90 percentile generally to cover the distribution of field values. Hence the range difference similarity D we use here is

$$D = 1 - \frac{D_2 + D_3 + D_8 + D_9}{4} \tag{4}$$

To keep consistent with the general similarity metric and convenient for comparison, we use the 1 minus the averaged range differences as the similarity metric called range difference similarity metric. Using this metric, we can get the similar distribution for matched pairs. The range difference similarity score's range is [0, 1]. The bigger the score value, the more correlated the pair. There are several cases to be considered here:

(1) If there is no overlaps between two field ranges, D would be as low as the minimum value 0.

(2) If two fields have similar distributions, D would be higher, up to 1.

(3) If two field ranges overlap at the head, tail or in the middle, the range difference similarity score could fall into middle.

In our data example, the matched pairs $< incident\_key, incident\_id >$ and $< product\_id, prod\_key >$ have D values as high as 0.996 and 0.999 respectively. In contrast, both the non-matched pair $< incident\_key, product\_id >$ and $< prod\_key, incident\_id >$ have no overlaps with D value 0, which are potentially not matched pairs. The more correlated the field pairs are, the higher range difference 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 use bucket dot product to differentiate them further. To minimize the error of range difference 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 experiments.

3.2.3 Bucket Dot Product Metric

After we consider the distribution of range difference score similarity metric, we propose the bucket dot product metric to cover the filtered results of range difference similarity metric. Bucket dot product metric is to divide the whole

concatenated range 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, so if we increase the bucket size before a certain value to calculate dot product, it could make non-matched pair similarities drop more while matched pair similarities drop less, thus effectively increasing the gaps between them. Generally, the dot product similarity is defined:

$$dot(A, B) = \sum_{i=1}^{n} A_i B_i \tag{5}$$

We use the bucket number (BN) to determine the inner range for calculating dot product. Considering two sets A and B's value ranges, we concatenate A and B's ranges as combined nested set AB, and then divide AB into the several buckets according to the BN. If there is any one value in A or B falling in a bucket, the bucket point is 1 or else it is 0 for them. The vector A_v and B_v's values are hence constructed from A, B and AB. Then we apply the general dot product to A_v and B_v. Therefore, the bucket dot product similarity (normalized) is defined as follows:

$$btDot(A_v, B_v) = \frac{\sum_{i=1}^{BN} A_{vi} B_{vi}}{|A_v||B_v|} \tag{6}$$

where BN decide the sparsity/density of range distribution. Because A and B sets usually have different size with different range, it would make sense for the BN the same for each set. We use normalized bucket dot product metric in our computation. In our data example, we calculate the bucket dot product $< incident\_key, incident\_id >$ as A and B with $BN = 3$. We first concatenate these two field ranges into a set {201, 202, 203, 204, 207, 208, 209}. Then we construct AB ={{201, 202, 203},{204, 207, 208}, {209}}. After that, we get the vector A_v = {1, 1, 1} and B_v = {1, 1, 0}. Finally, the bucket dot product similarity is 0.816, which is effectively high. If we set BN as 4, the bucket dot product similarity for this pair is 1, which is effectively highest.

Therefore, the BN is also an important factor to affect the quality of this metrics. According to our experimental observation, it is affected by the data range and distribution. Generally, matched pairs would have more similar range than non-matched pairs. A trade-off value of the BN would effectively improve matched pairs' similarities more and also do not help improve non-matched pairs' similarities much, which would potentially increase the similarity gaps between matched and non-matched pairs. The selection of BN will be discussed later in the experimental section.

3.2.4 Primary Key Constraint Matching

If we use all numerical fields and match between each other, it would be time-consuming or even unfeasible when we have a large number of fields. If there are n tables from database sources, the maximum $\binom{n}{2}$ comparisons are required. If we consider only table A and table B matching in a semantic way, the primary key constraint [3] could be utilized to reduce the time consumptions for pair comparisons. Moreover, relational databases should have primary keys and closely-related foreign keys, which generally describe the semantic meanings of the table. It is important to match two tables in this semantic way to construct an effective knowledge graph. Assuming every table has at least a primary key, we can identify the primary key and use the primary key's records to compare with all other keys records in other tables. We identify the primary key as a field that has unique values of its records before we apply range difference similarity score and bucket dot product. For example, in the data example, $product\_id$ and $incident\_key$ are identified as primary keys in these two tables T_P and table T_I, and thus we only need to compare $< product\_id, incident\_key >$, $< product\_id, prod\_key$, $< incident\_key, incident\_id >$ and $< product\_id, incident\_id >$ in total, reducing 33% times of comparisons.

3.2.5 Final Top K Selection

After we calculate all the pairs bucket dot product similarity, it is necessary to select top K results as the final results into the graph processing system because the field pairs with higher similarity scores are potentially more correlated. To ensure the effectiveness of constructed knowledge graph, we sort the similarity score of all the candidate pairs, then we verify this final field pairs list to select the top K results, which involves only very small amount of human labor.

3.3 Non-numerical Matching

Non-numerical fields in the data sources are different from numerical fields in data types and properties, we can not use the same approach as numerical matching. Moreover, most of non-numerical field records in matching pairs do not have equivalence relationship but possible have similar relationship. For example, $family$ field in table T_P and $product\_name$ field in table T_O could be correlated in the way that $product\_name$ belongs to $family$. The record pairs $< con5122, con5100 >$, $< cisco0510, cisco0500 >$, $< AIR1002, AIRseries >$ and $< 47-7066CC, 47-7000 >$ overlap and they are very similar, even though the records in each pair are not the same. However, the $< 47-7066c, 80-7066c >$ is not correlated, even though they have a large proportion of common characters. Because of the characteristics of non-numerical fields in network databases, we design a record-wise algorithm to find the most possible matches. The system flow is shown in Figure 4.

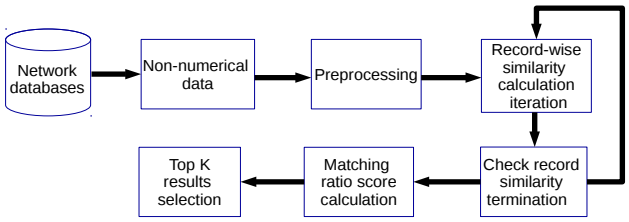

Figure 4: Non-numerical matching flow

As shown in the figure, the main idea of our non-numerical matching algorithm is as follows:

- Preprocess non-numerical data: this is an important step that decides the quality of our non-numerical matching algorithm. After splitting non-numerical data from the original data sources, we use our modified natural language processing methods of segmentation, stemming, prefixing for every field record value.
- Calculate the record-wise similarities iteratively: we also apply the cosine similarity between every record

pair in all field pairs. Considering the scalability of large-scale data matching, we sort the records in every field, and calculate the similarity of record-wise combination. While in the process of iterations for combination computation, we check the termination of every iteration, which effectively reducing time complexity.

- Calculate the matching ratios: after the record-wise combination for every field pair is finished, We calculate the defined matching ratio score for the field pair.
- Select Top K results: we sort the field pairs by the matching ratio non-increasingly, the field pairs of top K are selected as the final result of field pairs for input to graph processing system.

3.3.1 Preprocessing

Non-numerical matching considers partial match between two strings. For example, product "cisco0510" and product "cisco0500" could be in the same series. We propose the following preprocessing method.

(1) Parse every record string, 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 (certain length) of the original numerical strings and the new substrings if they are digital substrings. The prefix length is 2 here according to the general characteristics of network databases. For example, we have the original record X {'mem-4700m-64d='} in Table T_O, after preprocessing, we get a string collection {'4700', '64', '4700m', 'd', 'm', 'mem 4700m 64d', '64d', 'mem', '47xx'}.

3.3.2 Fast record-wise Similarity Calculation

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 our data are big. Especially if two fields is potentially a matched pair, it would have huge costs for useless computations. Therefore we propose an effective and fast way of record-wise comparisons to fulfill this. Intuitively, if two fields are correlated, there will be a high percentage of record-wise pairs that have higher similarities. The probability of two matched record pairs encountered is higher than non-matched record pairs. Therefore, we first sort all the records in each two fields A and B, then we get how many of records in A are matched with records in B, and vice-versa. That is applied to our proposed matching ratio score, which will be discussed in the next section.

In the fast record-wise comparison, each record pair similarity is based on the cosine similarity metric after preprocessing the record string. The similarity metric we use here is the cosine similarity between two records. 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), then we convert them into binary vector Vx, Vy according to XY, and then calculate the cosine similarity between them.

$$sim(V_x, V_y) = \frac{V_x \cdot V_y}{|V_x|\,|V_y|} \tag{7}$$

For example, we have preprocessed string collection X: {cisco, 0510, cisco0510, 05xx} and Y: {cisco, 05xx, 0500, cisco0500}, we transfer them into sets of X union Y, XY: {cisco, 0510, cisco0510, 05xx, 0500, cisco0500}, Then the binary vector for X and Y are V_x: {1, 1, 1, 1, 0, 0} and V_y: {1, 0, 0, 1, 1, 1}. Finally, we get the dot product and magnitude for these two binary vectors to calculate the cosine similarity $sim(X, Y) = (1 + 1)/(2 * 2) = 0.5$.

3.3.3 Matching Ratio Score

The matching ratio score is used to decide how to reduce the comparisons of matching in a fast and effective way. The threshold value T_{rn} is to decide how similar a record pair as a matched record pair for the network database and can be adjusted by users. Then we calculate our matching ratio scores to select non-numerical candidate field pairs.

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 fast matching ratio score(MR) is defined below.

$$MR(A, B) = \frac{1}{2} * (\frac{\sum_{i=1}^{m} A_i}{m} + \frac{\sum_{j=1}^{n} B_j}{n}) \tag{8}$$

where

$$A_i = \begin{cases} 1 & \text{if } \exists\ b_j \in B,\ sim(a_i, b_j) >= T_{rn} \\ 0 & \text{otherwise} \end{cases} \tag{9}$$

and

$$B_j = \begin{cases} 1 & \text{if } \exists\ a_i \in A,\ sim(b_j, a_i) >= T_{rn} \\ 0 & \text{otherwise} \end{cases} \tag{10}$$

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

3.4 More Efficient Hashing Algorithm for Non-numerical 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 records combination in the worst time complexity, which is time-consuming for large amounts of dataset with millions of records. Therefore, we propose to apply more scalable minHash-locality sensitive hashing algorithm (MH-LSH) [12] to estimate the matching score of non-numerical fields in the databases. It could greatly decrease comparison size and time for the record-wise non-numerical pairs with little lost of matching accuracy.

3.4.1 Matching Score Estimation with MinHash

Potential field pairs similarity score could be fast estimated with minHash signatures. Given two sets A and B, we could evaluate the similarity between the two sets as follows. We choose n hash functions. For each hash function, we let signature of set A be $h(A) = \min_{i:a_i \in A} h(a_i)$. Then, the probability that the two sets have the same minHash signature can be used to estimate the similarity between them.

$$\text{sim}(A, B) = P\{h(A) = h(B)\} \tag{11}$$

Here each record in a field A or B is also preprocessed with the same preprocessing method shown in section **??**.

The preprocessed record strings are combined into a new set. We also use k shingle (a substring of length k) to create a set of k-shingles string treated as one record string and apply n different hash function on the set.

3.4.2 Field Pairs 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 to use locality sensitive hashing (LSH) to select the candidate field pairs.

The values of minHash signature for one field h_i 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. Also, dissimilar pairs have low chance of falling into the same sketch. The probability that two fields have at least one sketch (of size b) in common out of r is

$$P_C(A,B) = 1 - (1 - \text{sim}(A,B)^b)^r \tag{12}$$

Therefore, we could find the candidate pairs with the designed number b and r. The selection of r and b is generally decided by the threshold $t = (1/b)^{1/r}$ shown in [12], which indicates how similar the two sets have to be considered as candidate pair, and could be set by users.

3.4.3 Final Top K Results

After we calculate all the matching ratio of field pairs, we obtain a list of pairs sorted by the matching ratio score values. Similar to numerical field matching, matched field pairs in the result list are more meaningful and important than non-matched field pairs, and thus we select top K results of non-numerical field pairs with high matching ratio scores to input the graph processing system. K value could be selected by users for deciding most effective field pairs in a graph and the size of the knowledge graph.

4. EXPERIMENTAL EVALUATION

In the section, we evaluate our algorithm NEMA for structured network database matching. In detail, we measure the effectiveness of NEMA using ground truth for numerical and non-numerical data that are annotated by humans. Meanwhile, experiments on the large datasets are also conducted, and we show the top-k result of matching field pairs. Moreover, the comparisons of NEMA with other based line methods are also shown.

4.1 Data Set

The structured network data available in the form of database tables are provided by Cisco Systems, Inc. The data sources include "install_base" and "service_request" databases. They are generated by different departments and groups of the company, which are heterogeneous and diversely distributed.

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 could involve the maximum 1,067,882 field matching decisions. With primary key constrains in numerical 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 pairs. Matched pairs are defined as field pairs that are observed to be correlated while non-matched pairs are field pairs which are not correlated. There are 60 field pairs of ground truth in numerical data and 40 pairs in non-numerical data.

For future reference, the table names in service_request database start with "T_", and start with "X_" in install_base database.

4.2 Experimental Setup

We implemented NEMA and prototype 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 our algorithm based on the ground truth data, Then, all the column pairs in large datasets are evaluated in the final experiment, which shows the effectiveness of NEMA. Finally, we compare with the common matching system COMA [1] on the ground truth in both schema-level and instance-level matching.

To construct the effective knowledge graph, identifying accurate matched pairs is much more important than finding the non-matched pairs. Therefore, we use Accuracy (ACC) to evaluate the accuracy. ACC is calculated with true positive(TP), true negative(TN), total positive pairs(P) and total negative pairs(N) (Here positive means matched and negative means non-matched).

$$ACC = \frac{TP + TN}{P + N} \tag{13}$$

4.3 Evaluation based on numerical data

We evaluate our algorithm NEMA on the numerical data in two parts. We use numerical ground truth data to evaluate the effectiveness of NEMA. Then the matching results of whole numerical field pairs are shown.

4.3.1 Evaluating of ground truth

We first show the evaluation result of NEMA and the compared baseline method-Jaccard similarity using numerical ground truth, then the evaluation of non-numerical ground truth. In the dataset, there are 30 matched ground truth field pairs which are from originally fields pairs with join operations in databases or annotated by humans. They are proved to be matched field pairs. We randomly sample some field pairs and select 30 non-matched pairs checked by humans.

Table 4 shows 10 matched and 10 non-matched field pairs of the ground truth examples. In every row "Table.field A" and "Table.field B" show the field pairs to be matched. The matched class indicates the pair is matched or non-matched. 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". The matched class value for this pair is 1, which means it is a matched field pair. The rest of rows share similar characteristics as well.

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

Figure 5 shows the Jaccard similarity scores on these ground truth pairs. Red circle represents matched pairs and blue star represents non-matched pairs. From this figure, we can

No.	Table.fieldA	Table.fieldB	Matched class
1	T_INCI.prod_hw_key	T_HW_PROD.bl_prod_key	1
2	T_INCIćur_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 4: Example of numerical ground truth field pairs

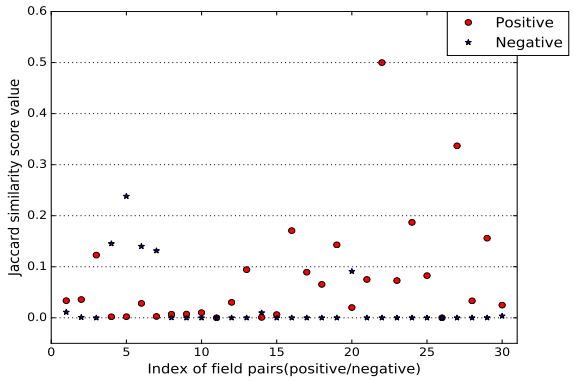

Figure 5: Jaccard similarity metric on the ground truth

see that there are about half of matched and non-matched pairs mixed together from which are difficult to differentiate.

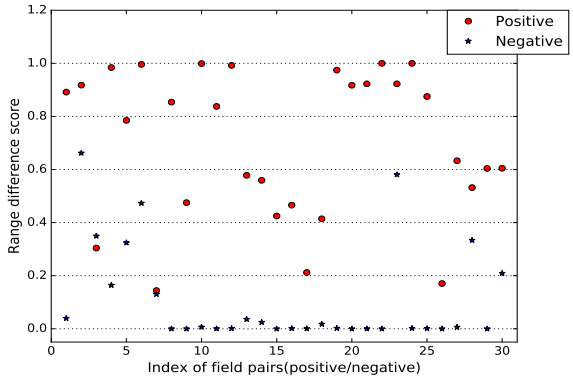

Figure 6: Range difference similarity metric on the ground truth

Figure 6 shows us range difference similarity metric on ground truth pairs. We know that the pair is more likely to match if the range difference similarity score is larger. The metric has better result over the Jaccard similarity result but there are some cases that we can not effectively differentiate the matched from non-matched pairs. Therefore, we need to continue refining our method using bucket dot product similarity metric.

From the previous range difference similarity metric result, we could decide a threshold and then rule out the result pairs below that threshold for non-matched field pairs. To minimize the error of initial matching, we can select a conservative threshold close to 0.1 (all of the matched pair are above 0.1) to eliminate some generally precise non-matched pairs.

After we select the filtered result pairs from range difference similarity metric, there are total 39 field pairs left, that is, about 35% of all ground truth pairs (all of them are non-matched pairs) are pruned. In these 39 pairs, there are 30 matched pairs and 9 non-matched pairs to be applied bucket dot product to them.

Figure 7 shows the bucket dot product result. X axis is the index of these 30 matched and 9 non-matched pairs, and Y is the normalized bucket dot product value in non-ascending order. We can see from that it has very good decision line between matched pairs and non-matched pairs in which the final threshold T_b is chosen around 0.1. If we select T_b to be 0.2, the final accuracy is about 93%. The final accuracy could be up to 95% when T_b is 0.1.

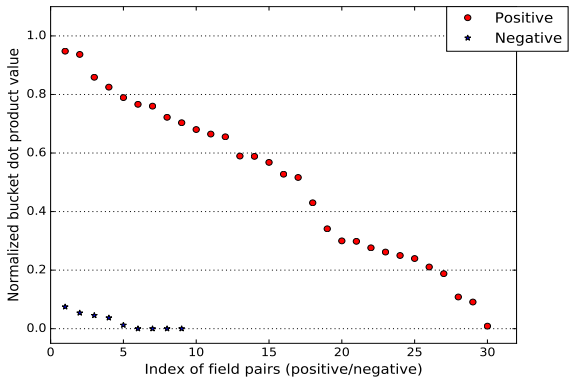

Figure 7: Bucket dot product metric on the ground truth after range difference similarity metric filtering

The bucket number is the main factor affecting the bucket dot product similarity and final results. To evaluate the effects of different bucket number, we use ground truth to test the accuracy with different bucket number values from 100, 200, 300, 400, 1,000 to 100,000,000 where there exists some numerical records that have values up to 10 billion for all the ground truth pairs. Figure 8 shows the ACC of bucket dot product with different bucket number when the threshold T_b is set at 0.1 and 0.2. It shows a similar summit that when bucket number is around 10,000 to 200,000 range, the accuracy is at an optimal value. The bucket dot product accuracy goes down when bucket number becomes smaller or bigger. We use a trade-off bucket number value 50,000 in our experiments to avoid overfitting.

4.3.2 *Top 20 similarity result*

The matching experiment based on all the 679 numerical fields is shown here. Table 5 shows top 20 matching results. We use range difference similarity score threshold and bucket number 50,000 in bucket dot product similarity

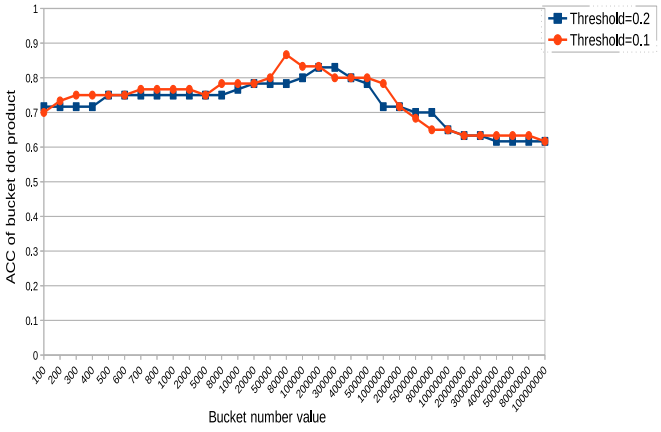

Figure 8: Bucket dot product accuracy comparison with different bucket number on the ground truth

computation. Almost all the rows are correct matches except that 2 field pairs which are possibly "noisy" matches with bold fonts. The accuracy could be up to 90%, which shows almost all of matched pairs could be captured by our numerical matching algorithm on large datasets and reducing human labor for matching. Also, with our system-aided matching findings, we could find some pair matches which are difficult to be found with human annotation such as "changewg_key" and "subregion_key".

To ensure the knowledge graph more meaningful and complete, top K results selection is finally decided by users. We can observe the top K results and rule out the wrong matches (if they exist) as the final input to the graph processing system so as to introduce less "noisy" connection of the graph.

Table.field A	Table.field B	Bucket dot product score
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

Table 5: Top 20 similarity result of numerical field pairs

4.4 Evaluation based on non-numerical data

We evaluate our algorithm NEMA on the non-numerical data in two parts as well. We use non-numerical ground truth data to evaluate the effectiveness of NEMA. The matching results of whole non-numerical field pairs are also shown.

4.4.1 *Evaluating of ground truth*

The experiment in non-numerical ground truth is shown here, which achieves high accuracy of matching results. There are 20 matched ground truth pairs which are annotated by humans. Additionally, 20 non-matched ground truth pairs are randomly selected and verified, part of which are shown in table 6.

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 6: Examples of non-numerical ground truth

We demonstrate the effectiveness of our record-wise similarity algorithm by analyzing the ground truth record-pairs. Table 7 shows the cosine similarity scores of our different record pairs in one field pair "T_PROD.prod_subgrp" and "T_HW_PROD.platform". The first 7 rows of pairs are matched record pairs that have higher similarity scores. The last 2 rows are not matched record pairs with lower score of 0.3333, which is lower than 0.4. It is conservative to capture the matching among different records with similarity value 0.4 assigned to the threshold T_{rn}.

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**

Table 7: Sample of non-numerical record pairs

We conduct the experiment on ground truth data and calculate the matching ratios of these field pairs. Figure 10 (f) shows matching ratios of these ground truth data matching in non-ascending order. The matching ratio values of almost all the matched pairs are above the non-matched pairs's. If we use the threshold 0.1 or select top 20 result from this, the accuracy could achieve about 95%, which shows the effectiveness of NEMA.

4.4.2 Top 20 similarity result

We have 779 non-numerical fields in the large datasets. Considering that almost all the primary keys in a table are not numerical fields, we do not consider primary key constraint matching method for non-numerical matching. The record similarity threshold is set $T_{rn} = 0.4$. The final top list of matching ratios of all the field pairs with whole column records above T_{rn} are obtained.

Table 8 shows the top 20 field pair matching results. We can see from that all the field pairs are potential matched pairs, which shows the effectiveness of NEMA.

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

Table 8: Top 20 matching result of non-numerical field pairs

4.5 Comparisons with COMA

We compare our algorithm NEMA with the COMA here. Because COMA is a popular hybrid matching tool and system for generic database and XML, etc. It has several versions and the latest version 3.0 supports both schema-level and instance-level matching. Also, it could be evaluated on our network database compared to other matching tools. We test the numerical and non-numerical ground truth matching in NEMA and COMA in schema-level and instance-level. We compare and analyze their accuracies and differences in matching results quantitatively.

4.5.1 Schema-level matching comparison

Figure 9 shows the accuracy comparison between COMA in schema-level and NEMA algorithm. NEMA uses the optimal threshold for deciding the boundary and COMA here uses the best field matching similarity "0" (which has no corresponding line in the COMA system) as threshold in schema-level matching. The accuracies of NEMA in numerical and non-numerical matching could be up to 95%. Moreover, the accuracy of NEMA is 8% higher than COMA in numerical matching and even 15% higher than non-numerical matching. It shows the effectiveness and high superiority of NEMA over COMA in schema-level. This is because COMA in schema level considers more about the data name, data type and name path, which could not effectively capture the irregular naming and abbreviations, etc.

We further analyze the difference of COMA and NEMA in matching the ground truth field pairs. Table 9 shows the

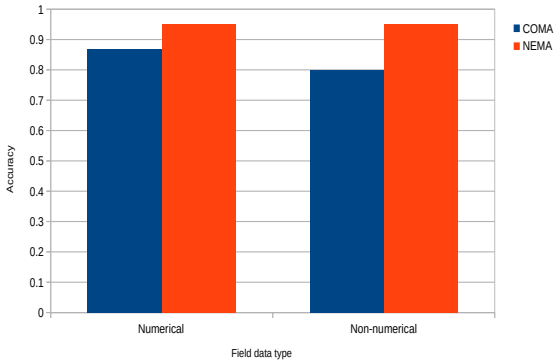

Figure 9: Accuracy comparison between NEMA and COMA in schema-level on numerical and non-numerical ground truth

field pairs in every row and its similarity score of COMA and NEMA. These field pairs are found to be matched pairs by NEMA with relative high similarity scores, but the 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 data instance for the decisions of field matching, which indirectly consider the semantic correspondences. However, if the data instances for some matched pairs are incomplete or missing, the similarity scores for these field pairs are also low. Table 10 shows the two field pairs that have low similarities in NEMA. Although the field names in each pair express the same thing semantically, the data instances in the fields are actually incomplete and have very few common between each other.

Table.fieldA	Table.fieldB	COMA similarity	NEMA similarity
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 9: Example of field pair matched by NEMA, but not by COMA

Table.fieldA	Table.fieldB	COMA similarity	NEMA similarity
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

Table 10: Example of field pair matched by COMA, but not by NEMA

4.5.2 Instance-level matching comparison

We also conducted experiments in instance-level matching comparison. COMA has one similar instance-level matching that uses the aggregated maximum record-wise similarities to obtain the final field pair similarity. The record similarity are based on the common similarity metrics such as edit

distance, trigram. The field matching similarity in COMA is defined as follows:

$$sim(A,B) = \frac{\sum_{i=1}^{m} max_{j=1\ldots,n}(sim(a_i,b_j)) + \sum_{j=1}^{n} max_{i=1\ldots,m}(sim(b_j,a_i))}{m+n} \quad (14)$$

We use our numerical and non-numerical ground-truth field pairs to evaluate the field pair similarity of COMA and compare with NEMA. Figure 10 show the field similarity score distributions in non-ascending order on numerical and non-numerical ground truth for COMA and NEMA in instance-level matching. Figure 10 (a) and (b) show the field similarity distribution of COMA on numerical ground truth matching. It can not effectively differentiate the matched and non-matched pairs for COMA because it could improve the non-matched field pair similarity scores dramatically for continuous numerical values with edit distance or trigram techniques. Figure 10 (c) and (d) show field similarity distribution of COMA on non-numerical ground truth matching. They shows a good boundary between matched and non-matched pairs according to the semantic of record pair matching of NEMA. Figure 10 (e) and (f) show field similarity distribution of NEMA on numerical and non-numerical ground truth matching. It can effectively differentiate the matched and non-matched pairs.

If we use the optimal threshold of field pair similarity for deciding the boundary in NEMA and COMA based on the similarity distributions, we can get the accuracy comparisons of them shown in Figure 11. NEMA uses 0.1 here as the final threshold and COMA here uses the best threshold 0.6 in numerical matching and 0.3 in non-numerical matching. The accuracies of NEMA in numerical and non-numerical matching are up to 95%. For numerical matching, The accuracy of COMA using edit distance and trigram are 10%-15% lower than NEMA because the inefficiency to identify the non-matched pairs of numerical ground truth. For the non-numerical matching, the accuracy of COMA is 2% higher than the accuracy of NEMA when we use the optimal threshold 0.2 for NEMA, 0.4 for COMA(edit distance) and 0.5 for COMA(trigram). However, the field similarity of COMA is measured based on its semantics of record pair matching, which is not quite applied to the network database matching. A large number of pairs with have high similarity records in COMA are not thought of as matching in the network databases scenario.

We further analyze the specific record pair examples of non-numerical instance-level matching. COMA uses standard edit distance and trigram to calculate the similarity of records, which is not quite suitable for our network matching requirements. Table 11 below shows 9 examples of records pairs and three different kinds of similarities(NEMA record similarity, COMA similarity using edit distance, COMA similarity using trigram). The first 7 rows are thought of as matched record pair, the last two rows are non-matched record pair. We can see from that the similarity of matched pairs based on COMA are quite similar around 0.7, from which is not easy to differentiate. While the similarities in 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.

Record 1	Record 2	NEMA record similarity	COMA (edit distance)	COMA (trigram)
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**

Table 11: Examples of record similarity comparisons

5. RELATED WORK

The structured data matching is an old and important research topic but unsolved and ever-growing problem, which has wide range of applications in database integration, migration, semantic query, etc. Survey papers [20, 21] proposed a solution taxonomy differentiating between element- and structure level, schema- and instance-level and language- and constraint-based matching techniques and approaches. Furthermore, [21, 22] reviewed the state-of-the-art matching systems which based on strings, structure, data instance and semantics matching technique using different schema format such as database, XML, OWL and RDFS, etc. In database schema application, previous common matching systems based on schema-level are introduced in several prototypes such as Similarity Flooding (SF) [15], Cupid [14], Artemis [4], Coma [8] and so on.

SF [15] proposed a matching algorithm that modeled 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, suffix test) to obtain the alignments between two nodes of the graph. The similarity is calculated from similar nodes to the adjacent neighbors through propagation the efficients. Our approach only relies on the data instance values to infer the matching of fields which does not utilize the structured properties and data type. However, SF used a metric for matching quality based on the intended matching results, which is similar to our accuracy metric based on top K results.

Cupid [14] was also a generic schema matching that discovers mappings between schema elements based on their names, data types, constraints, and schema structure with the assistance of a domain specific thesauri. It also explored the tree structure of schema to infer the similarity from upper levels. Similarly, Artemis [4] and DIKE [18] computed the structural affinity for all pairs of classes based on their name affinity and their respective class attributes. Cupid performs somewhat better overall than DIKE and Artemis.

Coma [8] is a composite matching tool providing extensible library and framework for combining obtained results. It contains mainly 6 elementary matchers using string-based techniques, 5 hybrid matchers using name and structural path, and one reuse-oriented matcher based on previous matching results . The composite matcher effectively improves the match quality over single matchers using the default combination strategy. Compared to SF, Cupid and DIKE, the overall average matching quality are the best of them [8]. The extended version Coma++ [1] utilized the shared taxonomy and pivot schema to further improve the overall matching quality. In our evaluation, we compared with the Coma++ method using the default combination

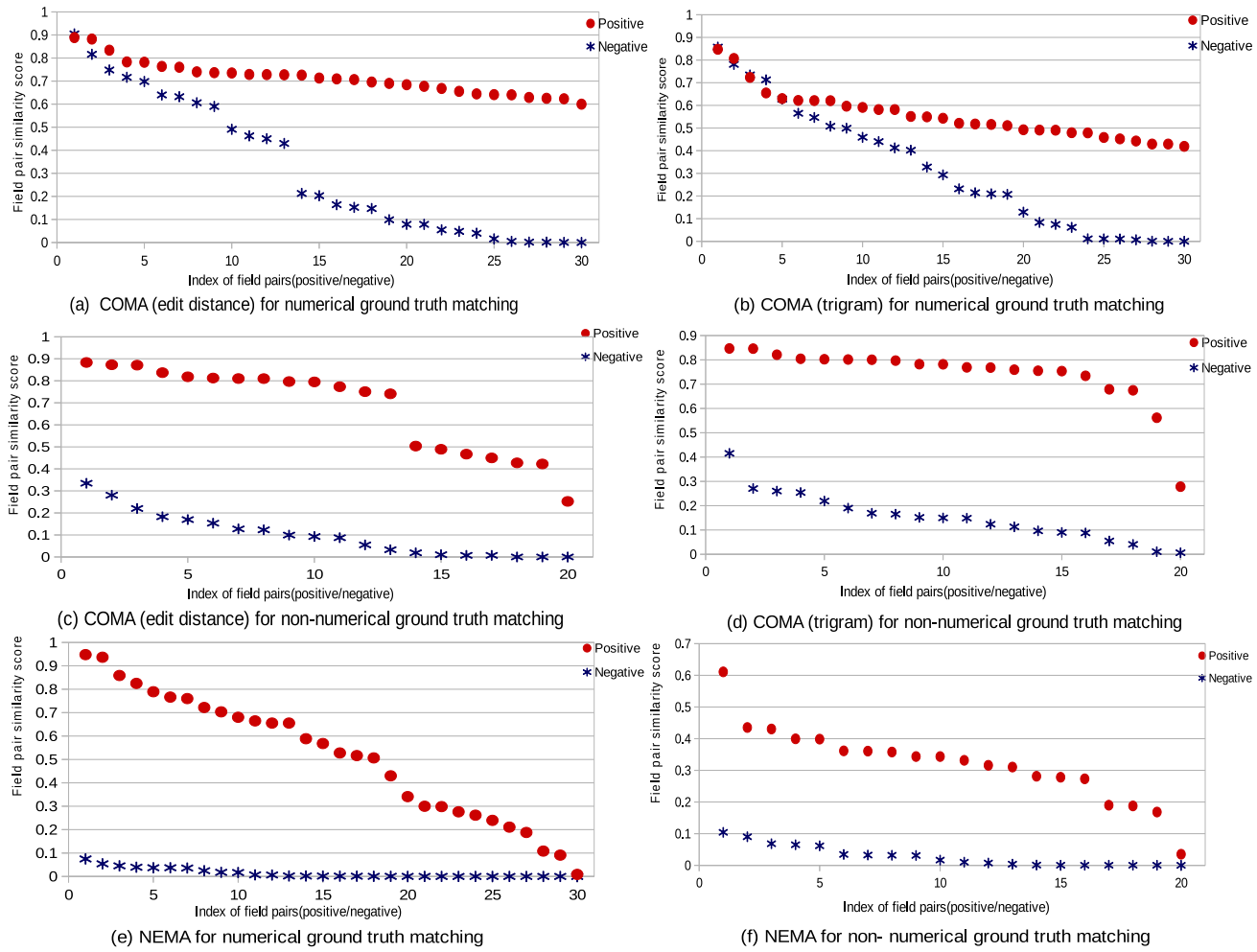

Figure 10: COMA and NEMA field similarity scores on numerical and non-numerical ground truths in instance-level matching

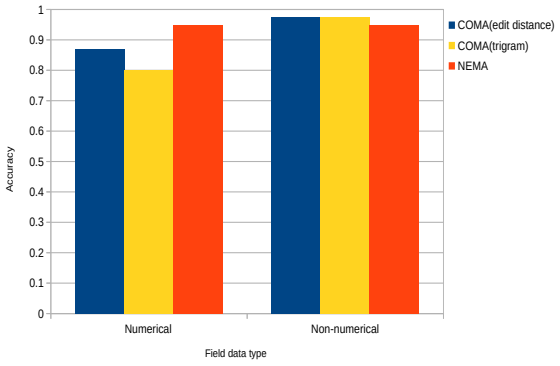

Figure 11: Accuracy comparison between NEMA and COMA in instance-level on numerical and non-numerical ground truth

strategy and found our algorithm NEMA has higher overall performance than COMA in schema-level.

Except from the previous matching approaches using field and structural information matching, data instanced-based approaches use the similarity of characteristics of their field instances to determine the similarity of fields. [13] utilized a corpus that contain schema and mappings between some schema pairs, and learn the constraints from schema statistics to help generating more matching pairs. But it has to have the labor of creating matching knowledge corpus and learning from text, in which our approach do not rely on any corpus and dictionary. [11, 6] used the mutual information of statistics to measures the similarity of schema instances between two columns to decide the matching, which shows a novel and effective method based on instances. Also, [19] proposed a new sample-driven approach which enables the end-users to easily construct their own data to match the source and target schema. COMA's extended version [10] also proposed 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 method on

content-based similarity measurement, which are compared with NEMA in the experimental evaluation.

To sum up, most of current popular matching approaches and systems focus on schema-level information matching. The data instances level matching approaches using field value are mostly based on some statistical models and machine learning from corpus. We further explore the database instance matching by comparing the field value using different metrics and proposed an effective and overall matching algorithm considering the characteristic for network database matching specifically for graph construction.

6. CONCLUSION

In this paper we propose a systematic algorithm to match structured data for graph constructing. Different from previous database matching method, we design different algorithms for numerical and non-numerical matching for effective knowledge graph construction. For numerical matching, we propose the range difference similarity and bucket dot product metrics in instance-level matching. For non-numerical matching, we design the record-wise similarity matching considering the characteristics of the network databases in instance-level matching, and reduce the time of matching for large-scale data. The results of our matching are demonstrated to be promising in which the accuracy could achieve up to 95%, which is much higher and more effective than the baseline methods.

With the explosion of big data and popularity of graph query, this work has the potential to significantly reduce the human work involving identifying the matching field for graph construction and also be applied for large-scale data matching. Lots of partial matching pairs can be found by our matching algorithm which humans can not easily detect. Furthermore, for industrial database matching to graph construction, this work is a fundamental step for the application of graph.

7. REFERENCES

[1] 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.

[2] J. Berlin and A. Motro. Database schema matching using machine learning with feature selection. In *Advanced information systems engineering*, pages 452–466. Springer, 2002.

[3] 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.

[4] S. Castano and V. De Antonellis. Global viewing of heterogeneous data sources. *Knowledge and Data Engineering, IEEE Transactions on*, 13(2):277–297, 2001.

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

[6] 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.

[7] 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.

[8] H.-H. Do, S. Melnik, and E. Rahm. Comparison of schema matching evaluations. In *Net. ObjectDays: International Conference on Object-Oriented and Internet-Based Technologies, Concepts, and Applications for a Networked World*, pages 221–237. Springer, 2002.

[9] H.-H. Do and E. Rahm. Coma: a system for flexible combination of schema matching approaches. In *Proceedings of the 28th international conference on Very Large Data Bases*, pages 610–621. VLDB Endowment, 2002.

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

[11] J. Kang and J. F. Naughton. On schema matching with opaque column names and data values. In *Proceedings of the 2003 ACM SIGMOD international conference on Management of data*, pages 205–216. ACM, 2003.

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

[13] 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.

[14] J. Madhavan, P. A. Bernstein, and E. Rahm. Generic schema matching with cupid. In *VLDB*, volume 1, pages 49–58, 2001.

[15] 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.

[16] 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.

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

[18] L. Palopoli, G. Terracina, and D. Ursino. The system dike: Towards the semi-automatic synthesis of cooperative information systems and data warehouses. In *ADBIS-DASFAA Symposium*, pages 108–117, 2000.

[19] 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.

[20] E. Rahm and P. A. Bernstein. A survey of approaches to automatic schema matching. *the VLDB Journal*, 10(4):334–350, 2001.

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

[22] 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.

[23] J. Webber. A programmatic introduction to neo4j. In *Proceedings of the 3rd annual conference on Systems, programming, and applications: software for humanity*, pages 217–218. ACM, 2012.