OGUNSEMI, A., MCCALL, J., KERN, M., LACROIX, B., CORSAR, D. and OWUSU, G. 2020. Ensemble-based relationship discovery in relational databases. In Bramer, M. and Ellis, R. (eds.) Artificial intelligence XXXVII: proceedings of 40th British Computer Society's Specialist Group on Artificial Intelligence (SGAI) Artificial intelligence international conference 2020 (AI-2020), 15-17 December 2020, [virtual conference]. Lecture notes in artificial intelligence, 12498. Cham: Springer [online], pages 286-300. Available from: https://doi.org/10.1007/978-3-030-63799-6_22

Ensemble-based relationship discovery in relational databases.

OGUNSEMI, A., MCCALL, J., KERN, M., LACROIX, B., CORSAR, D. and OWUSU, G.

2020

The final authenticated version is available online at: https://doi.org/10.1007/978-3-030-63799-6_22. This pre-copyedited version is made available under the Springer terms of reuse for AAMs: https://www.springer.com/gp/open-access/publication-policies/aam-terms-of-use

OpenAIR @RGU

This document was downloaded from https://openair.rgu.ac.uk

SEE TERMS OF USE IN BOX ABOVE

DISTRIBUTED UNDER LICENCE

Ensemble-Based Relationship Discovery in Relational Databases

Akinola Ogunsemi[1], John McCall[1], Mathias Kern[2], Benjamin Lacroix[1], David Corsar[1], and Gilbert Owusu[2]

[1] Robert Gordon University, Aberdeen, UK
{a.ogunsemi, j.mccall, b.m.e.lacroix, d.corsar1}@rgu.ac.uk
[2] BT Applied Research, Ipswich, UK
{mathias.kern, gilbert.owusu}@bt.com

Abstract. We performed an investigation of how several data relationship discovery algorithms can be combined to improve performance. We investigated eight relationship discovery algorithms like Cosine similarity, Soundex similarity, Name similarity, Value range similarity, etc., to identify potential links between database tables in different ways using different categories of database information. We proposed voting system and hierarchical clustering ensemble methods to reduce the generalization error of each algorithm. Voting scheme uses a given weighting metric to combine the predictions of each algorithm. Hierarchical clustering groups predictions into clusters based on similarities and then combine a member from each cluster together. We run experiments to validate the performance of each algorithm and compare performance with our ensemble methods and the state-of-the-art algorithms (FaskFK, Randomness and HoPF) using Precision, Recall and F-Measure evaluation metrics over TPCH and AdvWork datasets. Results show that performance of each algorithm is limited, indicating the importance of combining them to consolidate their strengths.

Keywords: Semantic Relationship · Primary/Foreign key Relationship · Data Discovery · Database Management · Ensemble-Based Discovery

1 Introduction

Data are one of the most important assets in the economy of the 21st century. Entire new industries rely on and are centred around the exploitation of large data sets, as many modern business processes generate millions or even billions of data records every day which are stored in databases. Understanding the relationship between data and gaining insight from data is central to their commercial success.

A user such as a business analyst may gain access to an existing database but expertise about how data is structured and how data tables relate to each other may not be provided, and little or no documentation exists. It could be that the technical and domain experts have moved on or left the business altogether, or many different groups have contributed to the database over time without a single authority fully understanding the overall information. This is a significant roadblock to exploiting this data. This challenge has mostly been addressed through highly time-intensive human analysis and exploration by domain experts [5, 6, 11, 22]. However, such an approach is limited due to time, cost and the amount of information that can be looked at, and is further likely to be error-prone [2, 7, 9]. Clearly, we need an automated mechanism to speed up the data discovery process.

In this paper, we investigate several relationship discovery algorithms that infer links between columns of tables and propose a framework that combines them into an overall framework. To the best of our knowledge, several approaches have been proposed to determine semantic relationships between database schemas and several variations have been reviewed with each having its strengths and weaknesses. See [1]. However, limited research work has been seen in exploring various ensemble strategies for combining several relationship discovery algorithms. One of these strategies was seen in [14] and this is in the space of schema matching which focuses on the manipulation of database schema elements for mapping [21].

Our motivation for combining several algorithms is to reduce the generalization error of the prediction produced by the individual algorithms [12]. Individual algorithms are diverse and independent so, the predictions made by a single algorithm may lead to imperfect discovery compared to a framework that combines several approaches [21].

Our proposed approach emphasizes recall and this is based on the premises that our methods discover different relationship types; primary/foreign key (explicit) and semantically equivalent (implicit) relationships. We only rely on the explicitly defined primary key/foreign key relationship as our gold standard. Thus, false positives (which are more likely to be semantically equivalent relationships) could be discovered due to the impact of the specified gold standard. This paper makes the following contributions;

– We investigated the problem of automatically discovering primary keys and foreign keys as well as semantically equivalent (implicit) relationships by ensemble methods.
– We used hierarchical clustering method as an ensemble framework to combine the prediction of individual discovery algorithms to better provide a comprehensive matching outcome.

The rest of this paper is structured as follows. Section 2 briefly explores some related work in relationship discovery. Section 3 defines the problem and describes the individual algorithms and their ensemble strategies. The experimental evaluations are provided in Section 4. Finally, a conclusion is given in Section 5.

2 Related Work

Several approaches have been proposed in the literature using different categories of data. For instance, Jiang and Naumann [11] proposed a holistic discovery of both primary key and foreign key (HoPF) as a subset of sets of unique column combinations and inclusion dependencies based on score function and several pruning rules. [22] proposed ten feature-based approach to automatically detect foreign keys using a machine learning model. In [6], K-Means clustering was used to solve multi-schema matching problem. They used a well-known TFIDF weighting to convert attributes to points in a vector space model and used cosine measure as a distance metric between attributes. [15] proposed a content based matching approach to determine the relationship between attributes which rely on the combined strengths of Google as a web semantic and regular expression as pattern recognition. [27] proposed an unsupervised solution that clusters set of columns to identify attribute relationships based on similar value characteristics using Earth Mover's Distance (EMD) as distance measures.

3 Ensemble-Based Discovery

3.1 Problem Definition

For a given database of n tables, $T = \{t_1, t_2, \cdots, t_n\}$, let $C = \{c_1, c_2, \cdots, c_\theta\}$ be the set of all columns of tables T where θ is the number of columns in the database. We define $t_i(c_i)$ as a table with an associated column where c_i is an i-th column of table t_i. Let $\Delta = \{(c_i, c_j) \colon \exists \quad t_i(c_i) = t_i(c_j), (c_i, c_j) \in C \times C\}$ be a set of column pairs (c_i, c_j) of the same table. We define $g_k = (C_k, E_k)$ as a graph of inferred relationships between set of columns (nodes) C_k and $E_k \subseteq C_k \times C_k \subset (C \times C) \setminus \Delta$ as the set of edges of g_k. Columns c_i and c_j are nodes in C_k, and each pair of columns (c_i, c_j) represents an edge in E_k, such that $(c_i, c_j) \in C_k \times C_k$. Let $f_k : C \times C \to g_k$ be a given discovery algorithm that produces graph g_k.

Our task is to determine the relationships between database tables which forms a graph G. The relationships include both primary/foreign key and semantic relationships which are determined by different discovery techniques to produce graphs, whereby the graphs are combined, with appropriate ensemble methods, to produce a global graph. The discovery techniques exploit metadata/schema information and column values available in relational database model.

1. Input Parameters
   (a) C - A set of all columns of the tables in T in the database DB.
   (b) f_k – A suitable method for discovering table relationships.
2. Output Parameters
   (a) $g_k = (C_k, E_k)$ - A graph containing a set of column pairs (c_i, c_j) in C_k where $C_k \in C$.

3.2 Relationship Discovery Algorithms

Pseudo-Primary Key Discovery (Pri) Pri is important in an application area, where no explicit definition of primary and foreign key constraints is available [22]. Existing work in this area can be explored in [11, 18, 22, 26]. We denote

A as the subset of C, $A \subseteq C$, which contains all columns with explicitly defined primary key columns in a database. Let B be defined as the subset of $C \setminus A$, $B \subseteq C \setminus A$, which are columns qualified as potential primary key candidates. The sets A and B do not share any columns. Let X be the union of A and B: $X = A \cup B$. We then calculate a graph g_k in which the nodes are columns from X plus their associated foreign key columns. Two column nodes are linked in the graph if they are in a primary/foreign key candidate relationship. We use the following four tests to infer B.

- Alphanumeric Datatypes Test: Columns with alphanumeric datatypes.
- Nullability Test: Non-null columns.
- Uniqueness Test: Columns with unique values.
- Word Character Test: Columns with letter, digit or underscore character.

We distinguish two cases in primary key/foreign key column pairs:

- Either a column is explicitly marked as a foreign key in the database itself,
- Or we need to establish that the second (foreign key) column only contain values that appear in the first (primary key candidate) column.

In Equation (1), $values(c_i)$ denotes values in column c_i. $w(c_i, c_j)$ returns 1 if two columns c_i and c_j are in a (potential) primary / foreign key relationship, and 0 otherwise:

$$w(c_i, c_j) = \begin{cases} 1, & \text{if } c_i \in A \text{ and } c_j \text{ is foreign key for } c_i \text{ and } t_i(c_i) \neq t_i(c_j) \\ 1, & \text{if } c_i \in B \text{ and } values(c_j) \subset values(c_i) \text{ and } t_i(c_i) \neq t_i(c_j) \\ 0, & \text{otherwise} \end{cases} \tag{1}$$

Name Similarity (NSim) Nsim is used to determine the linkages between tables by identifying the similarity between column names associated with each table. Several names used in identifying tables and columns are usually designated based on the nature of the business activities. Thus, column names may have inconsistent designations across tables. For instance, a column name "Customer Name", might be represented either as "CustName", "CustomerN" or "CstName". We used Jaro-Winkler to discover the similarity between two columns names (c_i and c_j) because it is a well-known algorithm used as far back in the 80s. This has currently been used in name similarity matching like entity matching [24]. See [10] and [25] for detailed mathematical definitions. We used java-string-similarity [3] library for our implementation. In equation (2), we define the $Score(c_i, c_j)$ function for all threshold dependent algorithms. The $Score(c_i, c_j)$ function returns 1 if a given $Metric$ produces a value greater than or equal to a given $Threshold$ and if c_i and c_j are not from the same table t_i. $Score(c_i, c_j)$ returns 0 otherwise. In the NSim algorithm, we implement $JWinkler(c_i,\, c_j)$ as the $Metric$ function. The value of $JWinkler(c_i,\, c_j)$ is a real number between the range of 0 and 1. If this value is greater than or equal to the $Threshold$, 1

[3] `https://github.com/tdebatty/java-string-similarity`

is assigned to $Score(c_i, c_j)$ which allows us to add the two columns as nodes to a graph g_k and connect them in the graph.

$$Score(c_i, c_j) = \begin{cases} 1, & \text{if } Metric \geq Threshold \text{ and } t_i(c_i) \neq t_i(c_j) \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

Usage-Based Approach (Usage) Usage uses a set of existing scripts from the database to infer relationship between tables. Scripts may include existing database logic such as procedures, functions, views or user queries. From these scripts, we extract all pairs of columns that co-occur in linking tables together. This approach was first introduced in [8]. Usage-based approach is suitable, in special cases, where column names are opaque or where there are no sufficient information about schema and data instance. However, it is often difficult to obtain suitable usage data [20]. We used General SQL Parser (GSP) library [4] to implement this approach. Let $S = \{s_1, \cdots, s_q\}$ be the set of existing scripts for a database. s_i denotes a single script and references a set of tables T_{s_i} in its logic. We define $T_{s_i} = \{t_{s_i 1}, \cdots, t_{s_i \iota}\}$, where ι is the number of tables in T_{s_i}. If script s_i contains a link statement, e.g. a join statement, between tables $t_{s_i x}$ and $t_{s_i y}$, and more specifically links the referenced columns in $t_{s_i x}$ and $t_{s_i y}$ respectively, we then, infer a link between those two columns and add the two columns as nodes to graph g_k.

Cosine Similarity Approach (Cosine) Cosine uses vector representation to measure the cosine angle between two vectors. Cosine was used in [6] as a distance metric measure for clustering attributes. We adopt cosine similarity to represent each attribute/column as a vector using Term Frequency Inverse Document Frequency (TFIDF) weighting computation. TFIDF is a term weighting scheme for cosine computation. TFIDF is a product of a term frequency (TF) weight factor and an inverse document frequency (IDF) weight factor. We define the cosine similarity metric between a pair of columns as $CoSim(c_i, c_j)$. See detailed computation of cosine similarity $CoSim(c_i, c_j)$ in [23]. The cosine similarity value $CoSim(c_i, c_j)$ is a real number between 0 and 1 and it represents the $Metric$ function defined in equation (2). If the value is greater than or equal to the $Threshold$ in equation (2), we then assign 1 to $Score(c_i, c_j)$ or 0 otherwise. A $Score(c_i, c_j)$ of 1 will add the two columns as nodes to graph g_k and connect them in the graph.

Semantic Similarity in a Taxonomy (Sem) Sem exploits additional, external information to measure the similarity between a pair of words or concepts. The key resource used is a knowledge-based database, such as a business-specific ontology or a general-purpose database like WordNet [17], which encodes relations between concepts. For example, when column headers are described slightly

[4] `http:/dpriver/www..com/`

differently e.g., "AUTOMOBILE_NO" can conceptually mean the same as "VEHICLE_ID". We used a knowledge based function in [16], to measure the similarity between a pair of columns. We define the knowledge metric as $Sem(c_i, c_j)$ which computes the average similarity score by combining resultant similarity scores of substrings of c_i and c_j. We define v_{ik} as the $k-th$ substring / term associated with the name for column c_i. The $SemSim(v_{ik}, v_{jk})$ metric in equation (3) is used in the $Sem(c_i, c_j)$ metric computation (see [16]) which returns a similarity score between a pair of terms v_{ik} and v_{jk} associated with the names of columns c_i and c_j respectively. A stopword (i.e, most common word in a language) term returns score 0. If both terms are not in the knowledge networks, name similarity $JWinkler(v_{ik}, v_{jk})$ is used. $JWinkler(v_{ik}, v_{jk})$ is also used for terms that are either adjectives or adverbs in the knowledge network. Lastly, if the pair of terms are both verbs or nouns in the knowledge networks, we then compute $sim_{Lin}(v_{ik}, v_{jk})$, otherwise score returns 0.

$$SemSim(v_{ik}, v_{jk}) = \begin{cases} 0, & \text{if } v_{ik} \text{ or } v_{jk} = stopword \\ JWinkler(v_{ik}, v_{jk}), & \text{if } v_{ik} \text{ or } v_{jk} \notin ontologies \\ sim_{Lin}(v_{ik}, v_{jk}), & \text{if } v_{ik} \text{ and } v_{jk} \in ontologies(noun) \\ sim_{Lin}(v_{ik}, v_{jk}), & \text{if } v_{ik} \text{ and } v_{jk} \in ontologies(verb) \\ JWinkler(v_{ik}, v_{jk}), & \text{if } v_{ik} \text{ or } v_{jk} \in ontologies(adv) \\ JWinkler(v_{ik}, v_{jk}), & \text{if } v_{ik} \text{ or } v_{jk} \in ontologies(adj) \\ 0, & \text{otherwise} \end{cases} \tag{3}$$

We implemented $sim_{Lin}(v_{ik}, v_{jk})$ using Semantic Measures library [5]. See computation in [13]. It takes two concepts and returns their semantic relatedness value. Let $Sem(c_i, c_j)$ represents the $Metric$ function in equation (2). The $Score(c_i, c_j)$ function defined in equation(2) is assigned 1 if the $Sem(c_i, c_j)$ is greater than or equal to the $Threshold$ and if the column pair are not from the same table.

Soundex Similarity (Soundex) It is a phonetic algorithm that indexes a string by sound in English. It simply evaluates letters of a string and assigns a numeric value. Soundex is used in the context of identifying the relationship between two tables based on the phonetic similarity between their column names. See computation in [19]. We implemented Soundex using Apache Commons library [6] in Java. We denote the phonetic similarity as $Sdex(c_i, c_j)$. The value of $Sdex(c_i, c_j)$ is between 0 and 4. A $Sdex(c_i, c_j)$ value of 4 means that a pair of column names sound strongly similar and 0 means otherwise. $Sdex(c_i, c_j)$ computes the $Metric$ value in equation (2) to assign $Score(c_i, c_j)$ a score 0 or 1.

Value Ranges Similarity (Val) Val uses minimum and maximum values of column pairs to determine whether they are linked. Val works with numeric,

[5] `https://www.semantic-measures-library.org/sml/index.php?q=downloads#sml`
[6] `https://commons.apache.org/proper/commons-codec/download_codec.cgi`

strings or date datatypes. Two columns of the same datatype are similar if they have similar value range pattern. We denote a_i as a pair of minimum and maximum values $\langle min(c_i), max(c_i)\rangle$ for column c_i. Columns c_i and c_j are logically equivalent ($a_i \equiv a_j$), if a_i is similar to a_j or vice versa. The check $range(c_i, c_j)$ in equation (4) returns 1 for similar value ranges between two columns c_i and c_j or 0 otherwise. (c_i, c_j) is added to g_k if $range(c_i, c_j)$ is 1.

$$range(c_i, c_j) = \begin{cases} 0, & \text{if } datatype(c_i) \neq datatype(c_j) \text{ or } t_i(c_i) = t_i(c_j) \\ 1, & \text{if } a_i \equiv a_j \text{ and } t_i(c_i) \neq t_i(c_j) \\ 0, & \text{otherwise} \end{cases} \tag{4}$$

Content-Based Similarity (Col) Col exploits and compares data instances to determine the relationship between columns pair. $content(c_i, c_j)$ returns 1 if the set of value samples in column c_j is a subset of unique values of column c_i, and 0 otherwise. (c_i, c_j) is added to graph g_k if $content(c_i, c_j)$ is 1.

$$content(c_i, c_j) = \begin{cases} 0, & \text{if } datatype(c_i) \neq datatype(c_j) \text{ or } t_i(c_i) = t_i(c_j) \\ 1, & \text{if } samplevalues(c_j) \subset values(c_i) \text{ and } t_i(c_i) \neq t_i(c_j) \\ 0, & \text{otherwise} \end{cases} \tag{5}$$

3.3 Ensemble Strategies

We used voting scheme and hierarchical clustering to find the best combination of graphs generated by the discovery algorithms.

Voting Scheme The voting scheme checks if the individual graphs share common edges. It uses a weighting measure to determine the proportion of graphs that contain a pair of columns (c_i, c_j). Given, g_k and $P_{weighting}$, we can generate a global graph G. We defined w_k in equation(6) as a score that indicates whether a pair of columns (c_i, c_j) exists in graph g_k. w_k returns 1 if a pair of columns (c_i, c_j) is an element of $E_k \in g_k$ and 0 otherwise. We compute the weighted value of $p_{(c_i,c_j)}$ in equation(7) for each pair of columns (c_i, c_j) as the sum of scores of w_k divided by the number of graphs m. We then generate a global graph G by adding a pair of columns (c_i, c_j) to graph G where the obtained weighting value of $p_{(c_i,c_j)}$ is equal or greater than a given $P_{weighting}$.

$$w_k = \begin{cases} 1, & \text{if } (c_i, c_j) \in E_k \\ 0, & Otherwise \end{cases} \tag{6}$$

$$p_{(c_i,c_j)} = \frac{\sum_{k=1}^{m} w_k}{m} \tag{7}$$

Hierarchical Clustering We used the clustering approach (hierarchical clustering) proposed in [3] to group together variables which are strongly related to each other into homogeneous clusters. Each variable represents a graph g_k. We used hierarchical clustering proposed in [3] to group the variables (graphs) into clusters based on how they are strongly linked. See [3] for detailed formulation of the hierarchical clustering method proposed in the study.

The rationale for this strategy is that members in each cluster contains similar prediction pattern. We can therefore, select a member of each cluster and combine with a selected member of another cluster to exploit diversity and reduce error in prediction.

We represent each graph g_k as a categorical variable $\varPhi_k$ and we defined $\{\varPhi_1, \cdots, \varPhi_m\}$ as a set of $\varPhi$ categorical variables where $\varPhi_k \in \varPhi$ and $k = 1, \cdots, m$. m is denoted as the total number of variables (number of graphs). Then, let x be a set of all pairs of columns in $(C \times C)\backslash\varDelta$, such that $E_k \subset x$. $\varPhi_k$ has the same dimension (number of column pairs) as x, and for each variable $\varPhi_k$, contains binary strings of 0 and 1. String 1 indicates that $(c_i, c_j) \in E_k$ and 0 otherwise.

Let $\mathcal{P} = (\mathcal{P}_1, \ldots, \mathcal{P}_q)$ be a partition into q clusters of $\varPhi$ variables. q denotes the total number of clusters and $\mathcal{P}_l$ is the $l - th$ cluster of $\mathcal{P}$.

We generate $\{G_1, \ldots, G_\alpha\}$ as a set of graphs G where G_i is the ith graph in G. We expect to obtain at least a graph from G graphs which gives a strong and improved prediction of relationship between column pairs. We denote α as the total number of graphs (i.e, number of possible combination of variables from each cluster). This is expressed in the equation below;

$$\alpha = \prod_{l=1}^{q} |\mathcal{P}_l|$$

$|\mathcal{P}_l|$ denotes the number of $\varPhi$ variables in cluster $\mathcal{P}_l$. Let $\varPhi_{jl}$ be a variable in cluster $\mathcal{P}_l$, so that each graph $G_i \in G$ is produced by combining a set of q variables selected from each cluster using intersection operation. This is expressed below as follows;

$$G_i = \bigcap_{l=1}^{q} \varPhi_{jl}$$

4 Experimental Evaluation

4.1 Dataset description

The two datasets (TPCH[7] and AdvWork[8]) used for this paper are synthetic datasets. For ease of comparison, the TPCH used the same parameter setting

[7] http:/www.tpc.org/tpch

[8] https://github.com/Microsoft/sql-server-samples/releases/tag/adventureworks

used in [11]. We stored the individual datasets in an Oracle database. The characteristics of the two datasets are given in Table 1. Both synthetic datasets contain database views and procedures we used as existing database queries.

Table 1. Data Characteristics

Data	No of Tables	No of Columns	AvgNo of Columns per Table	MaxNo of Columns per Table	Total Rows	No of Queries	Primary keys	Foreign keys
TPCH	8	61	8	16	6,885,051	22	8	8
AdvWork	71	486	7.5	26	754,248	33	27	45

4.2 Experimental Set-up

We implemented our algorithms in Java and performed experiments on an Intel Core i5 vPro 2.4GHz CPU with 8GB Ram. We first run experiments for threshold dependent algorithms to select appropriate thresholds required for an overall comparative analysis. The range of thresholds include; NSim (0.50 - 0.95), Soundex (1 - 4), Sem (0.50 - 0.95) and Cosine (0.50 and 0.95). Next, we explored the performance of individual algorithms based on mean completion time over 20 runs. We then combined their predictions based on voting scheme and hierarchical clustering. Finally, we compared performance with state-of-the-art algorithms (FaskFK [4], Randomness [26] and HoPF [11]). FastFK combines heuristic features with different rules to detect foreign keys, which assumes that each table pair can hold only one foreign key. Randomness algorithm uses a randomness metric to discover both single-column and multi-column foreign keys by using the earth-mover distance (EMD) to measure the data distribution similarity between foreign key candidates. HoPF uses score function and pruning rules for holistic discovery of both primary and foreign keys as a subset of sets of unique column combinations and inclusion dependencies.

4.3 Evaluation Metrics

We employ three standard evaluation metrics to measure the performance of individual algorithms; Precision, Recall and F-Measure. Let g_1 be a graph of actual relationships between set of columns (nodes) C_1 and E_1 be the set of edges of g_1. Let g_2 be another graph containing inferred relationships between columns discovered by a discovery algorithm with set of columns C_2 as nodes and E_2 as edges of g_2. Let $TP = E_1 \cap E_2$. TP represents true positives, a set of edges common to both E_1 and E_2 and $|TP|$ is the number of edges in TP. Let $FP \subseteq E_2 \setminus TP$ be a subset of $E_2 \setminus TP$ which represents false positives. FP and TP do not share common edges and $|FP|$ is the number of edges in FP. Let $FN \subseteq E_1 \setminus TP$ and $|FN|$ represents the number of edges in FN.

Let $x = (C \times C) \setminus \Delta$ be all edges formed from all pairs of columns, such that E_1 and E_2 are both subsets of x. Then, we define TN (True negatives) as $TN = x \setminus (E_1 \cup E_2)$ and $|TN|$ is the number of edges in TN.

Precision is computed as $\frac{|TP|}{|TP|+|FP|}$ which evaluates the percentage of relevant outcomes discovered by our algorithms. We compute recall as $\frac{|TP|}{|TP|+|FN|}$. Recall evaluates the percentage of relevant outcomes that were discovered by a discovery algorithm over the total relevant outcomes. We then compute F-measure as $\frac{2*Precision*Recall}{Precision+Recall}$ to measure the weighted harmonic mean of precision and recall.

4.4 Comparative Analysis

Discovery Completion Time The mean completion time of individual algorithms is shown in Table 2. This involves 20 experimental runs over the TPCH dataset. Name similarity (NSim) algorithm records the lowest mean time of 4.25 milliseconds with a minimum time 0 millisecond and maximum time 16 milliseconds. On the other hand, content-based (Col) approach takes longer time than other discovery algorithms with recorded mean time of 2868283.3 milliseconds (47.81 minutes). Figure 1 shows example of graphs generated by Sem (a) and Soundex (b) algorithms over the TPCH dataset.

Table 2. Completion Time of Discovery Algorithms in Milliseconds

Algorithms	MinTime	AveTime	MaxTime
Cosine	72	136.15	351
Pri	1228045	1501709.15	2926454
NSim	0	4.25	16
Col	2107575	2868283.3	7901629
Val	15671	16170.85	19454
Sem	3481	4491.35	8711
Soundex	3	5.95	34
Usage	144	342.2	1552

Comparison with existing techniques We compared our results with the results already reported in [11]. The specified gold standard used for evaluation is based on primary/foreign key relationship. Performance is shown in Table 3. The best performance for the TPCH dataset results in f-measure of 1.00 which was achieved by Randomness. The Randomness performance is largely attributed to the assumption that true primary keys exist and are known. Randomness exactly matches the known primary keys to columns with the same names which makes it possible for the algorithm to achieve that score. Our methods exploit database information differently without any known assumptions about true primary key existence. Three of our methods (Sem, Usage, Cosine) outperformed the FastFK algorithm on TPCH dataset with respective f-measure scores 0.83, 0.80 and 0.73.

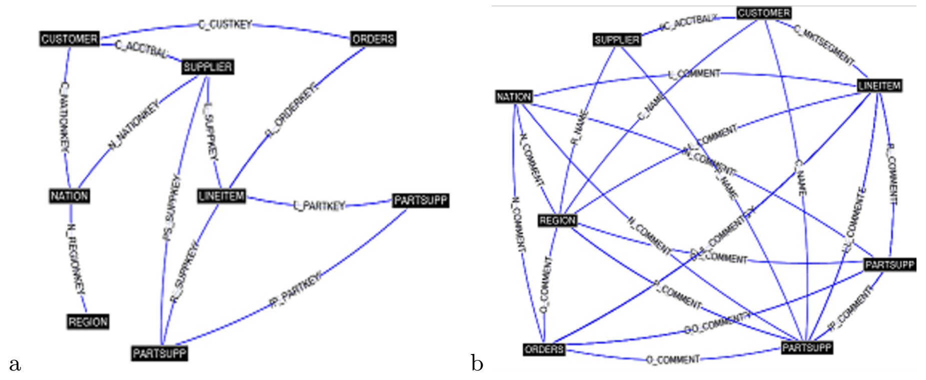

Fig. 1. Graph Example

The performance of the Usage based approach is highly dependent on the quality of existing queries. For instance, if the queries use all the true primary keys to link tables then f-measure of 1.00 is possible.

Table 3. Comparison of Proposed Discovery Algorithms, Ensemble Strategies and state of the art results already reported in [11]

Categories	Algorithm	TPCH			Algorithm	AdvWork		
		Precision	Recall	F-Measure		Precision	Recall	F-Measure
Individual Algorithms	Cosine	0.73	0.73	0.73	Cosine	0.02	0.86	0.04
	Pri	0.24	0.91	0.38	Pri	0.03	0.90	0.06
	NSim	0.22	1.00	0.37	NSim	0.02	0.83	0.04
	Col	0.15	0.91	0.26	Col	0.01	0.93	0.03
	Val	0.08	1.00	0.15	Val	0.01	0.04	0.02
	Sem	0.77	0.91	0.83	Sem	0.02	0.83	0.04
	Soundex	0.07	0.27	0.12	Soundex	0.02	0.83	0.04
	Usage	0.89	0.72	0.80	Usage	0.14	0.58	0.23
2-Clusters Combination	Sem_Pri	1.00	0.91	0.95	Sem_Pri	0.18	0.73	0.29
	NSim_Val	0.85	1.00	0.9	Cosine_Pri	0.19	0.76	0.30
	NSim_Pri	0.91	0.91	0.91	NSim_Pri	0.18	0.73	0.29
3-Clusters Combination	Sem_NSim_Pri	1.00	0.91	0.95				
	Sem_NSim_Col	1.00	0.82	0.90				
	Sem_NSim_Val	0.91	0.91	0.91				
Voting	PVote50	0.79	1.00	0.88	PVote75	0.18	0.80	0.29
	PVote62.5	0.85	1.00	0.92	PVote87.5	0.16	0.49	0.25
	PVote75	1.00	0.91	0.95				
State-of-the-Art	FastFK	0.56	0.90	0.69	FastFK	0.32	0.97	0.49
	Randomness	1.00	1.00	1.00	Randomness	0.90	0.41	0.56
	HoPF	0.88	0.88	0.88	HoPF	0.31	0.84	0.46

In respect to the AdvWork dataset in Table 3, our algorithms could not achieve significant f-measure results apart from the Usage-based algorithm that achieved f-measure score of 0.23. The poor performance is largely attributed to huge number of false positives discovered by our methods. These false positives are caused by the inherent semantic relationships which are not defined in the primary/foreign key relationship that we have used as gold standard in our evaluation. In terms of recall, Content-based (Col) and Primary key (Pri) algorithms achieved 0.93 and 0.90 respectively.

Overall, the diversity displayed by the individual algorithms are based on the characteristics of data. The algorithms have performed in different ways over the two datasets. However, the diversity of the independent algorithms can be exploited by combining their outcomes in different ways to improve performance.

Ensemble Performance We used voting strategy and hierarchical clustering to combine the diversity of the outcomes produced by the individual algorithms and compare performance with the results of the state of the art algorithms reported in [11]. Voting thresholds are given as;
$P_{weighting} = (12.5\%, 25\%, 37.5\%, 50\%, 62.5\%, 75\%, 87.5\%, 100\%)$. We include results of the top three voting thresholds over the two datasets (TPCH and AdvWork) in Table 3. For TPCH dataset, the three top voting thresholds, 75%, 62.5% and 50% (i.e., PVote75, PVote62.5 and PVote50) achieve respective f-measure scores 0.95, 0.92 and 0.88, precision scores 1.00, .85 and 0.79 and recall scores 0.91, 1.00 and 1.00. The voting scheme could not reach the f-measure score (1.00) delivered by the Randomness algorithm, however, a 0.95 score was achieved which outperformed HoPF and FastFK.

In the AdvWork dataset, despite the poor performance of the individual approaches, the voting scheme helped in improving the performance. Although, this strategy could not outperform the selected state of the art algorithms. The reason for this is due to the existence of several semantic relationships which are not explicitly specified in the database structure. We only relied on the explicit specifications of primary key/foreign key relationships for our evaluation.

The drawback in the voting strategy is that the voting strategy takes all the discovery algorithms into consideration. This is quite expensive in terms of the computational time. For instance, based on Table 2, the total average completion time to implement a voting strategy will take about 4391143.2 milliseconds (73.19 minutes). However, this could be addressed by using an appropriate sophisticated parallel computing approach which is beyond the scope of this paper.

In terms of the hierarchical clustering strategy, with the TPCH dataset, we evaluate two clusters and three clusters combinations. We obtain 15 unique combinations of algorithms with two clusters and 18 unique combinations with three clusters. The best performance in the two clusters combination for instance, is produced by Sem_Pri. Sem_Pri gives an f-measure score of 0.95 with precision score equal to 1.00. This means that no false positives were predicted with the combined efforts of both Sem and Pri algorithms. Sem_NSim_Pri obtains an f-measure score of 0.95 with precision score of 1.00. When comparing performance with state of the art algorithms, Sem_Pri and Sem_Nsim_Pri give better performance than HoPF and FastFK.

In the AdvWork dataset, two clusters were predicted by the clustering algorithm. We obtained top three unique combinations of the two cluster based on f-measure performance. The best performance is produced by the combination of Cosine_Pri with f-measure score 0.30, precision score 0.19 and recall score

0.76. The results reported by the state of the art algorithms outperformed the combined efforts of Cosine and Pri. See Table 3. We have earlier attributed the poor performance over the AdvWork dataset to lack of sufficient gold standard used in the study. We only relied on the primary/foreign key relationships which is specified in the database. An expert opinion would be needed for additional information about semantic relationship.

Overall, results show clearly that some specific algorithms are relevant when combined in certain ways. The Pri for instance, has the tendency of performing well when combined with algorithms like Sem, Nsim or Cosine irrespective of the data characteristics. However, the suitability of Pri is impaired due to speed considerations. Therefore, the choice of algorithms to combine depends largely on user's compromise on speed, reliability and sufficiency.

5 Conclusion

We investigated eight discovery algorithms and showed how their predictions can be combined to identify more comprehensive links between database tables involving both primary/foreign key and semantic relationships. The discovery algorithms identify potential links in different ways based on different levels of database information. In evaluating the performance of our approaches, based on two diverse datasets, we showed that different levels of schema information can be exploited and combined in a view to reduce the generalization error associated with each algorithm. We showed in our experiment that an appropriate combination strategy can be adopted to improve relationship discovery outcomes. The performance of individual discovery algorithm is limited, indicating the necessity to combine several algorithms to bring together their strengths. We compared precision, recall and f-measure with state of the art algorithms.

References

1. Alwan, A.A., Nordin, A., Alzeber, M., Abualkishik, A.Z.: A survey of schema matching research using database schemas and instances. International Journal Of Advanced Computer Science And Applications **8**(10) (2017)
2. Bellahsene, Z., Bonifati, A., Rahm, E.: Schema matching and mapping, section 6. Data-Centric Systems (2011)
3. Chavent, M., Kuentz, V., Liquet, B., Saracco, L.: Clustofvar: an r package for the clustering of variables. arXiv preprint arXiv:1112.0295 (2011)
4. Chen, Z., Narasayya, V., Chaudhuri, S.: Fast foreign-key detection in microsoft sql server powerpivot for excel. Proceedings of the VLDB Endowment **7**(13), 1417–1428 (2014)
5. De Carvalho, M.G., Laender, A.H., GonçAlves, M.A., Da Silva, A.S.: An evolutionary approach to complex schema matching. Information Systems **38**(3), 302–316 (2013)
6. Ding, G., Sun, T., Xu, Y.: Multi-schema matching based on clustering techniques. In: 2013 10th International Conference on Fuzzy Systems and Knowledge Discovery (FSKD). pp. 778–782. IEEE (2013)

7. Do, H.H.: Schema matching and mapping-based data integration (2006)
8. Elmeleegy, H., Ouzzani, M., Elmagarmid, A.: Usage-based schema matching. In: 2008 IEEE 24th International Conference on Data Engineering. pp. 20–29. IEEE (2008)
9. Hai, D.H.: Schema matching and mapping-based data integration. University of Leipzig (2005)
10. Jaro, M.A.: Advances in record-linkage methodology as applied to matching the 1985 census of tampa, florida. Journal of the American Statistical Association **84**(406), 414–420 (1989)
11. Jiang, L., Naumann, F.: Holistic primary key and foreign key detection. Journal of Intelligent Information Systems pp. 1–23 (2019)
12. Kotu, V., Deshpande, B.: Data Science: Concepts and Practice. Morgan Kaufmann (2018)
13. Lin, D., et al.: An information-theoretic definition of similarity. In: Icml. vol. 98, pp. 296–304. Citeseer (1998)
14. Marie, A., Gal, A.: Managing uncertainty in schema matcher ensembles. In: International Conference on Scalable Uncertainty Management. pp. 60–73. Springer (2007)
15. Mehdi, O.A., Ibrahim, H., Affendey, L.S.: An approach for instance based schema matching with google similarity and regular expression. Int. Arab J. Inf. Technol. **14**(5), 755–763 (2017)
16. Mihalcea, R., Corley, C., Strapparava, C., et al.: Corpus-based and knowledge-based measures of text semantic similarity. In: Aaai. vol. 6, pp. 775–780 (2006)
17. Miller, G.A.: Wordnet: a lexical database for english. Communications of the ACM **38**(11), 39–41 (1995)
18. Papenbrock, T., Naumann, F.: A hybrid approach for efficient unique column combination discovery. Datenbanksysteme für Business, Technologie und Web (BTW 2017) (2017)
19. Pinto, D., Vilarino, D., Alemán, Y., Gómez, H., Loya, N.: The soundex phonetic algorithm revisited for sms-based information retrieval. In: II Spanish Conference on Information Retrieval CERI (2012)
20. Rahm, E.: Towards large-scale schema and ontology matching. In: Schema matching and mapping, pp. 3–27. Springer (2011)
21. Rahm, E., Bernstein, P.A.: A survey of approaches to automatic schema matching. the VLDB Journal **10**(4), 334–350 (2001)
22. Rostin, A., Albrecht, O., Bauckmann, J., Naumann, F., Leser, U.: A machine learning approach to foreign key discovery. In: WebDB (2009)
23. Salton, G., Buckley, C.: Term-weighting approaches in automatic text retrieval. Information processing & management **24**(5), 513–523 (1988)
24. Wang, Y., Qin, J., Wang, W.: Efficient approximate entity matching using jaro-winkler distance. In: International Conference on Web Information Systems Engineering. pp. 231–239. Springer (2017)
25. Winkler, W.E.: Frequency-based matching in fellegi-sunter model of record linkage. Bureau of the Census Statistical Research Division **14** (2000)
26. Zhang, M., Hadjieleftheriou, M., Ooi, B.C., Procopiuc, C.M., Srivastava, D.: On multi-column foreign key discovery. Proceedings of the VLDB Endowment **3**(1-2), 805–814 (2010)
27. Zhang, M., Hadjieleftheriou, M., Ooi, B.C., Procopiuc, C.M., Srivastava, D.: Automatic discovery of attributes in relational databases. In: Proceedings of the 2011 ACM SIGMOD International Conference on Management of data. pp. 109–120. ACM (2011)