

DELFT UNIVERSITY OF TECHNOLOGY

MASTERS THESIS

**An exploratory journey to combine
schema matchers for better relevance
prediction**

Author:
Wang Hao WANG

Supervisor:
Asterios KATSIFODIMOS &
Andra IONESCU &
Jerry BRONS

*A thesis submitted in fulfillment of the requirements
for the degree of Master of Science*

in the

**Web Information Systems Group
Software Technology**

November 23, 2022

Declaration of Authorship

I, Wang Hao WANG, declare that this thesis titled, “An exploratory journey to combine schema matchers for better relevance prediction” and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed:

Date: 23 November 2022

DELFT UNIVERSITY OF TECHNOLOGY

Abstract

Electrical Engineering, Mathematics and Computer Science
Software Technology

Master of Science

**An exploratory journey to combine schema matchers for better relevance
prediction**

by Wang Hao WANG

Current speed of data growth has exponentially increased over the past decade, highlighting the need of modern organizations for data discovery systems. Several (automated) schema matching approaches have been proposed to find related data, exploiting different parts of schema information (e.g. data type, data distribution, column name, etc.). However, research showed that single schema matching techniques fails to effectively match schemas, whilst combinatorial schema matching systems show more promise. With the introduction of combinatorial schema matching systems, new challenges arise regarding selection and combining strategies. This research attempts to explore different techniques for determining the importance of each matcher in a combinatorial schema matching system by determining the weights of each matcher and comparing them through a comprehensive evaluation.

Student Number 4724925

Thesis committee: Prof.dr.ir. G.J.P.M. Houben, TU Delft
Assistant Prof.dr. A. Katsifodimos, TU Delft
Dr. L. Y. Chen, TU Delft

Contents

| | |
|---|------------|
| Declaration of Authorship | iii |
| Abstract | v |
| 1 Introduction | 1 |
| 1.1 Problem description | 2 |
| 1.2 Contribution | 2 |
| 1.3 Research Questions | 3 |
| 1.4 Overview | 3 |
| 2 Background | 5 |
| 2.1 Data discovery | 5 |
| 2.1.1 Applications | 6 |
| Aurum | 6 |
| Attribute discovery | 7 |
| 2.2 Relatedness | 7 |
| 2.3 Schema matching taxonomy | 8 |
| 2.3.1 Schema-based matching | 9 |
| Element-level approaches | 10 |
| Structure-level approaches | 10 |
| 2.3.2 Instance-based matching | 11 |
| 2.4 Combining matchers | 13 |
| 2.4.1 Matcher selection | 13 |
| 2.4.2 Aggregation Strategies | 14 |
| 2.5 Related work | 15 |
| 2.5.1 LSD (2001) | 15 |
| 2.5.2 Cupid (2001) | 16 |
| 2.5.3 Coma (2002) | 16 |
| 2.5.4 Embley et al (2002) | 17 |
| 2.5.5 MKB (2003) | 17 |
| 2.5.6 Harmony (2006) | 18 |
| 2.5.7 YAM (2009) | 18 |
| 2.5.8 Data tamer (2013) | 18 |
| 2.6 Conclusion | 19 |
| 3 Methodology | 21 |
| 3.1 Approach overview | 21 |
| 3.2 Methods | 22 |
| 3.2.1 Multiplicative weight update method | 22 |
| Determining the weights | 22 |
| 3.2.2 Reinforcement learning | 23 |
| Determining the weights | 24 |
| 3.2.3 Linear programming | 24 |

| | | |
|----------|---|-----------|
| | Determining the weights | 25 |
| 3.2.4 | Machine learning | 26 |
| | Machine learning model | 26 |
| 4 | User Interface | 29 |
| 4.1 | Configuration | 29 |
| 4.2 | Labelling | 29 |
| 4.3 | Evaluation | 30 |
| 5 | Experiments | 31 |
| 5.1 | Data | 31 |
| 5.1.1 | TPC-H | 32 |
| | Fabricated TPC-H | 32 |
| 5.1.2 | IMDb | 32 |
| | Fabricated IMDb | 32 |
| 5.1.3 | ING | 33 |
| 5.2 | Metrics | 33 |
| 5.2.1 | Cluster Metrics | 33 |
| 5.2.2 | Traditional Metrics | 34 |
| 5.2.3 | Valentine Metrics for Score Quality | 34 |
| 5.3 | Results | 35 |
| 5.3.1 | TPC-H | 35 |
| 5.3.2 | Fabricated TPC-H | 39 |
| 5.3.3 | IMDb | 42 |
| 5.3.4 | Fabricated IMDb | 43 |
| 5.3.5 | ING | 45 |
| 5.3.6 | Discussion | 49 |
| 5.4 | Machine Learning Results | 49 |
| | Data | 49 |
| 5.4.1 | Evaluating the model | 50 |
| 5.4.2 | Discussion | 52 |
| 6 | Conclusion | 55 |
| 6.1 | Limitations | 55 |
| A | User Interface: 4-step form for configuring our methodology | 57 |
| B | User Interface: Labelling ground truth clusters | 59 |
| C | User Interface: Evaluation tool | 61 |
| D | Manual noise mapping for fabricated TPC-H column names | 63 |
| E | Results: Example of formed clusters compared to the ground truth | 65 |
| | Bibliography | 67 |

List of Figures

| | | |
|------|--|----|
| 2.1 | Example tables | 8 |
| 2.2 | An extended taxonomy of schema matching methods described by Shvaiko and Euzenat, 2005. The taxonomy is divided into two layers: <i>Granularity/input interpretation</i> and <i>Kind of input layer</i> . Granularity/input interpretation layer shows what kind of information is being used by the schema matching technique, whilst the kind of input layer shows the types of possible inputs. | 12 |
| 5.1 | Reduction of clusters into a list of relationships | 34 |
| 5.2 | Valentine precision for each k | 38 |
| 5.3 | Valentine recall for each k | 38 |
| 5.4 | Valentine f-measure for each k | 38 |
| 5.5 | Valentine metrics for each k in the TPC-H dataset | 38 |
| 5.6 | Valentine precision for each k | 41 |
| 5.7 | Valentine recall for each k | 41 |
| 5.8 | Valentine f-measure for each k | 41 |
| 5.9 | Valentine metrics for each k in the fabricated TPC-H dataset | 41 |
| 5.10 | Valentine precision for each k | 44 |
| 5.11 | Valentine recall for each k | 44 |
| 5.12 | Valentine f-measure for each k | 44 |
| 5.13 | Valentine metrics for each k in the IMDb dataset | 44 |
| 5.14 | Valentine precision for each k | 46 |
| 5.15 | Valentine recall for each k | 46 |
| 5.16 | Valentine f-measure for each k | 46 |
| 5.17 | Valentine metrics for each k in the fabricated IMDb dataset | 46 |
| 5.18 | Valentine precision for each k | 48 |
| 5.19 | Valentine recall for each k | 48 |
| 5.20 | Valentine f-measure for each k | 48 |
| 5.21 | Valentine metrics for each k in the ING dataset | 48 |
| 5.22 | Confusion matrix for one execution on the test set | 51 |
| 5.23 | Confusion matrix for one execution on the test set | 52 |
| A.1 | Step 1: Select the matchers the user want to include | 57 |
| A.2 | Step 2: Select the cutoff threshold (described in chapter 3.2.1) | 57 |
| A.3 | Step 3: Select one of our provided methods | 58 |
| A.4 | Step 4: Select the data you want to use | 58 |
| A.5 | User interface to change flexible setting values | 58 |
| B.1 | User interface allowing users to select columns and form a cluster | 59 |
| B.2 | User interface showing the created clusters and button for labelling all the relationships based on the created clusters | 59 |
| C.1 | Formed clusters are shown to the user along with the used settings | 61 |

| | | |
|-----|---|----|
| C.2 | Zoomed-in and interactive dendrogram showing the hierarchy of the columns | 61 |
| C.3 | Evaluation page with amount of clusters and the corresponding metric scores | 62 |
| E.1 | Example chord diagram showing the clusters and comparison with the ground truth | 65 |

List of Tables

| | | |
|------|---|----|
| 3.1 | Selected matcher for each method | 22 |
| 5.1 | Best achieved metrics from running our methods on the original TPC-H data with the optimal settings | 37 |
| 5.2 | Best achieved metrics from running our methods on the fabricated TPC-H data with the optimal settings | 40 |
| 5.3 | Best achieved metrics from changing different fabrication factors for the fabricated TPC-H data with the optimal settings | 42 |
| 5.4 | Best achieved metrics from running our methods on the original IMDb data with the optimal settings | 43 |
| 5.5 | Best achieved metrics from running our methods on the fabricated IMDb data with the optimal settings | 45 |
| 5.6 | Best achieved metrics from running our methods on the ING data with the optimal settings | 47 |
| 5.7 | Amount of used labelled data and its distribution amongst the related and unrelated column-pairs | 50 |
| 5.8 | The mean and standard deviation of the accuracy when we execute the learning and test process 100 times | 50 |
| 5.9 | Achieved metrics of the model on the test set | 50 |
| 5.10 | The mean and standard deviation of the accuracy when we execute the learning and test process 100 times with oversampling | 52 |
| 5.11 | Achieved metrics of the model on the test set with oversampling | 52 |
| D.1 | The mapping between the original fabricated data of TPC-H and the altered data for validation of identified problem | 64 |

Chapter 1

Introduction

According to the recent blog post of Hack, [2021](#), the speed of the growth of data usage has been skyrocketing over the past decade. In the last decade, the data usage has increased with almost 1500% and is forecasted to grow with an annual growth rate of 23%. With the rise of technology, data keeps being collected in a large variety of domains with high velocity and volumes, especially in the digital era (Dong and Srivastava, [2013](#)). Data can be collected through online and traditional transaction systems, sensors, social media, mobile devices, and other diverse sources (Zikopoulos and Eaton, [2011](#)). This brings up the importance of being able to search, discover, compile and analyze relevant information for a user's specific task in science, business, and society at large (Weikum, [2013](#)). An immense amount of data is being generated, whilst there exists no standardized formats, storage, and transfer mechanisms (Christensen et al., [2018](#)). According to an article written by King, [2019](#), it is believed that approximately 80% of the data will be unstructured in 2025. Unstructured data creates a unique challenge for organizations wishing to use their information for analysis. As a result of the growth of the (unstructured) data, data analysts are drowning in data but starving for insights as finding relevant data takes an improper amount of time.

With this immense growth of data, modern organizations will inevitably face a **data discovery problem**, where the analysts has to spend more time finding the relevant data rather than actually analyzing it (Castro Fernandez et al., [2018](#)). For example, an analyst needs to create a feature table for training a machine learning model to determine the stock prices of a set of companies for the upcoming year. The information for each company are stored in multiple locations, each possibly a good feature. With the current growth of data, countless data online are distributed over multiple different data sources. Finding correlated and relevant information regarding each company takes a lot of time as the analyst needs to search through various sources and determine the relevance by going through the data of all the data sources. It is a very labor intensive task which requires a significant amount of time as 80% of a data scientists' time is dedicated for data discovery (Stonebraker, Ilyas, et al., [2018](#)). This example highlights the problem we are facing today where manual discovery has reached its limits as it scales poorly with the amount of data.

In this work, our ultimate goal is to be able to provide a way to help users find relevant information through automation to decrease the labor intensity of finding the relevant data. Rather than automatically combining the discovered related data, we aim to provide suggestions for relevant data to speed up the process of finding them as state-of-the-art tools are not advanced enough to provide fully automation yet. In order to achieve this goal, we need find a way to achieve better performance compared to the state-of-the-art data discovery tools. This work is in collaboration

with ING and therefore can be evaluated using real life data.

1.1 Problem description

For many years, integrating data was done manually by experts in the field (Rahm and Bernstein, 2001). However, with the increasing numbers of external (and internal) datasets, of increasing diversity and complexity, the difficulty of finding relevant data increases as we reach our limitations of scalability for manually discovering data (Bogatu et al., 2020; Castelo et al., 2021; Rahm and Bernstein, 2001). Manually discovering data results in implicitly giving meaning to the data, based on how the user interprets it, and therefore also highlights the challenges to be faced by automated data discovery. Over the past decade, data discovery tools are developed to automate extraction of useful information of the data to assist the analysts in finding relevant data (Goebel and Gruenwald, 1999). To define the scope of our research, we solely look into discovering relatedness between known (internal) datasets, and therefore disregard the discovery process of finding the data sources. We hence focus on detecting relatedness between known datasets. The vast majority of modern data discovery tools are based on one common critical component: **schema matching**, i.e. capturing relationships between elements of different schemas (Koutras et al., 2021). Schema matching is in charge of identifying two objects that are semantically related and finding semantic correspondence between elements of two schemas (Do and Rahm, 2002). Information regarding schemas and/or instances can be exploited for the purpose of extracting the semantic correspondences between different elements (Shvaiko and Euzenat, 2005).

Different data discovery tools exploit a certain selection of the available schema and/or instance data (Rahm and Bernstein, 2001). For example, the work of Zhang et al., 2011 exploits the underlying distribution of the instances, whilst Madhavan, Bernstein, and Rahm, 2001 uses linguistic and structural similarity between the schemas. However, as data keeps advancing due to the rapid changes in technology, individual data discovery tools fail to keep up with the additional introduced complexity (Do and Rahm, 2002). Typically, data discovery tools exploit a selection of data characteristics, e.g. data distribution, but solely depending on one data discovery tool is not sufficient to keep up with technology changes as they are inherently fragile and limited (Bernstein et al., 2004). Studies have therefore tried to combine different discovery tools to improve the performance by compensating for the shortcomings of the individual tools, with success (Do and Rahm, 2002; Engmann and Maßmann, 2007). However, as data progresses, combining different forces from different discovery tools will not suffice with simple aggregation methods in complex cases (Tu and Yu, 2005). Coma (Do and Rahm, 2002) and Coma++ (Engmann and Maßmann, 2007) are examples which tried to combine different discovery tools to achieve better performance. They have implemented different aggregation methods such as MAX, MIN and AVERAGE, but did not report any results for the weighted average, which is supposedly better (Tu and Yu, 2005, Do and Rahm, 2002).

1.2 Contribution

We try to solve this problem by using a weighted approach to determine the importance of each discovery tool and the similarity between columns. The assigned similarity for each column-pair will be used for clustering in order to group related

columns together.

We propose four different methods to address the problem of tuning the weights, each with different characteristics. Additionally, we also propose an user interface to help the user navigate through our methodology. The four different methods are compared with each other through comprehensive evaluation.

1.3 Research Questions

In this thesis report, we will consider two main research questions to guide this research.

- *RQ1: What characteristics determine the selection of discovery tools to use for an effective combinatory method?*
- *RQ2: What method(s) combines solutions of different discovery tools into a better performing solution?*

Research question RQ1 helps us focus on picking the appropriate data discovery tools to use for our methodology, whilst research question RQ2 helps us focus on how to combine the tools to reach our goal.

1.4 Overview

This thesis report is structured as following: we will firstly provide the background information necessary to understand the terminology in this thesis report in chapter 2. Afterwards, the methodology will be explained in chapter 3, which provides insight on the approach itself and its motivations. In chapter 4, the proposed user interface will be described. Afterwards, in chapter 5, we show the comprehensive evaluation of the different methodologies described in chapter 3. At last, a conclusion will be made and remarks for future work will be made in chapter 6.

Chapter 2

Background

This chapter will introduce the research of data discovery in general. To dive deeper into the specifics, we will also describe what relatedness is within the data discovery research and how it can be detected using schema matching. A taxonomy of schema matching approaches will be given and elaborated upon. Furthermore, we dive deeper into the schema matching research and provide information about combining different schema matching approaches and the respective aspects of it. At last we mention some related work.

2.1 Data discovery

In modern enterprises, multiple co-existing information systems are used in parallel to take care of the information of different parts of the enterprise (Ziegler and Dittich, 2007). With different co-existing information systems needing to communicate with each other, especially regarding data to provide a more unified view, the importance of the **data integration problem** rises.

Definition: *Data integration is the problem of combining data residing at different sources and providing the user with a unified view of these data (Lenzerini, 2002)*

Data integration systems allow the users to delegate the tasks such as locating different sources, interact with each source in isolation, and manually combine the data from multiple sources to the data integration machines (Halevy, 2001). A core component of such data integration system, which tries to locate different located data sources, is the **data discovery** component (Koutras et al., 2021). The data discovery component tries to tackle the **data discovery problem**, which is the problem where analysts spend more time finding relevant data than actually analyzing it (Castro Fernandez et al., 2018). By (semi-)automating the process of finding relevant data, it reduces the amount of time needed for analysts to find relevant data. Definitions regarding data discovery slightly varies within literature. However in our research, we will use the term data discovery to describe the process of finding relevant information within known sources, rather than locating data sources and identifying meanings of entities. A more formal definition of data discovery is written below.

Definition: *Data discovery is the process of navigating numerous data sources in order to find relevant datasets as well as the relationships among those datasets (Koutras et al., 2021)*

For many years, integrating data was done manually by experts in the field (Rahm

and Bernstein, 2001). However, with the increasing numbers of external (and internal) datasets, of increasing diversity, that are available for organizations to use increases the difficulty of finding relevant data (Bogatu et al., 2020; Castelo et al., 2021). A number of approaches has been proposed to organize and index data collections, from domain-specific repositories to data lakes to make the data more discoverable for search engines (Chapman et al., 2019). Improving the discoverability of data will benefit the people and organizations involved in the data lifecycle, however, different organizations develop their own portal with support for data discovery (wu et al., 2019). It requires collective efforts from data collectors, data providers, data repositories, data librarians and research trainers to improve the discoverability of the data.

Most search algorithms, used by popular search engines, focus on discovering general web-pages rather than the characteristics of data in a specific domain (Li, 2010). Search engines lack the judgement of the similarity between spatial objects for this purpose. This leads to **schema matching** being the solution, which is responsible for finding semantic correspondences between elements of two schemas (relatedness) (Li and Clifton, 2000). Schema matching has become one of the most fundamental pieces in many schema and data translation and integration applications (Do and Rahm, 2002) and has been a critical component in many data discovery tools (Koutras et al., 2021).

Data discovery comes in different forms, serving different purposes, but all have a common goal: *augment a dataset with data previously unknown to the user* (Koutras et al., 2021). There are three identified purposes for data discovery:

1. Search for tables that can be joined (Cafarella, Halevy, and Khoussainova, 2009)
2. Augment a given table with more data entries or extra attributes (Yakout et al., 2012)
3. Find similar tables to a given one using different similarity measures (Castro Fernandez et al., 2018)

Typically, a data discovery tool receives a dataset as input and tries to find other datasets in the repository which might be related to it (Koutras et al., 2021). As data grows exponentially over the years, data discovery becomes more and more important (MacMillan, 2014). Having mounds of data is useless unless you find a way to extract insight from it. Data discovery is applicable in a wide range of fields. For example, analysts need to find relevant data to be able to extract insights (Daniel, Lee, and Naveen, 2016), whilst machine learning models need to find relevant features to train their model on (Krawiec, 2002).

2.1.1 Applications

Aurum

Aurum is a generalized data discovery system which creates Enterprise Knowledge Graphs (EKG) dynamically based on signatures to tackle the general data discovery problem (Castro Fernandez et al., 2018). Generating and maintaining the EKG within Aurum has a time complexity of $O(N)$ due to their two-step process avoiding all-pair comparisons: *signature-building* and *relationship-building*. The signature-building stage summarizes each column into a profile maintaining information such

as content sketches (e.g. MinHash), data distributions, etc. The relationship-building stage uses Locality-Sensitive Hashing (LSH) to reduce the all-pair problem to an approximate nearest neighbor problem. With these two stages, an EKG can be derived representing columns as nodes and candidate relationships as edges with weights. To query in the EKG, Source Query Retrieval Language (SRQL) is being used and fastened by using G-index. Aurum relaxes the discovery problem by using approximation methods and proved to be useful for varied discovery needs.

Attribute discovery

Zhang et. al. studied the possibility of grouping attributes together rather than detecting the same data types (Zhang et al., 2011). As traditional schema matchers often only detect simple foreign key relationships to express the relations, Zhang et. al. tried to group columns together that are from the same attribute. For example, telephone numbers and social security numbers are of the same primitive data type which multiple schema matchers see as a potential relationship. By looking at this problem from a different perspective, Zhang et. al. tried to find strongly related columns and group them together, representing an attribute: *a set of values that appear to have the same or similar meaning within the context of a particular database instance* (Zhang et al., 2011). This is done by calculating the Earth Mover's Distance (EMD) for every pair of columns and use those values to cluster the columns by sorting them and stopping at a certain EMD threshold. The clusters from the EMD method result in distribution clusters which represents major categories. For all these major clusters, a complete graph is created where each edge denotes a positive or negative relationship based on the intersection EMD. The complete graph is then used for correlation clustering to ultimately find the clusters for each attribute. Each attribute therefore contains columns which can be joined with each other.

2.2 Relatedness

Relatedness is a very broad term used in different domains where two activities require similar knowledge or input (Hidalgo et al., 2018). When we look at the relatedness between two schemas in data integration applications, we can use the definition described in the paper of Bogatu et. al. (Bogatu et al., 2020).

Definition: *Relatedness can be defined as having values of some attributes from table S being drawn from the same domain represented by some attribute in table T (Bogatu et al., 2020)*

Two schemas being related means that the attribute(s) drawn from one schema is relevant for populating the other table. To illustrate the definition behind relatedness, we will take a look at our running example in figure 2.1, where we see that the **Movie:MovieID** and **Actor:PlaysIn** columns both contain the same instances and are describing the same attribute. The two respective schemas are related with each other as the actor schema can populate the movie schema with additional information regarding actors for each respective movie.

Relatedness within schemas can be categorized into two fundamental categories according to Koutras et al., 2021:

- Unionable

- Joinable

For the unionable case, relations store data of the same conceptual entity type using the same attributes. In our example data in figure 2.1, we see two schemas which are unionable: Movie and Action Movie. They describe the exact same conceptual entity type and use the same attributes to store information of movies. The main difference between the two schemas is that the attributes have different names. However, the underlying entity is exactly the same, making the schemas perfectly unionable.

For the joinable case, two relations store complementary data of same conceptual entity type. In our example data in figure 2.1, we have a joinable case with the Movie and Actor schemas. The Actor schema has a field **PlaysIn**, which describes the same attribute as the MovieID in the Movie schema. Combining both schemas with a join operator will result in one big schema with information about the movie and the respectful actor.

Schemas can have different degrees of relatedness, subject to how many of their attributes are related to some target attribute. The degree of relatedness (similarity score) is usually normalized and indicates the likeliness of the relationship (Do and Rahm, 2002). The similarity score therefore is a real number value ranging from 0 (strong dissimilarity) to 1 (strong similarity), indicating the similarity and the plausibility of their correspondence (Do and Rahm, 2002). These similarity scores are determined by schema matching.

| Movie | | |
|---------|---------|---------|
| MovieID | Name | Genre |
| #MVA1 | Ip man | Action |
| #MVS2 | Titanic | Romance |

| Actor | | | |
|---------|------------|------------|---------|
| ActorID | Name | BirthDay | PlaysIn |
| #DJSMD | Billy Zane | 24/02/1966 | #MVS2 |
| #FOSPA | Xing Yu | 27/12/1978 | #MVA1 |

| Action Movie | | |
|--------------|-----------|---------|
| M_ID | M_name | m_genre |
| #MVA6 | Nobody | Action |
| #MVS9 | Uncharted | Action |

FIGURE 2.1: Example tables

2.3 Schema matching taxonomy

The relatedness between schemas, which is used to determine the relationship between the schemas, are determined through the **schema matching** component, which is a critical component within data discovery in modern discovery tools (Madhavan, Bernstein, and Rahm, 2001).

Definition: *Schema matching is the task of finding semantic correspondences between elements of two schemas (Li and Clifton, n.d.)*

The schema matching component is therefore in charge of identifying two objects that are semantically related and has the task of finding semantic correspondences

between elements of two schemas (Do and Rahm, 2002). Schema matching used to be a time-consuming, error-prone and tedious manual process, which becomes increasingly impractical with the higher numbers of schemas as the effort required is typically linear with the the number of schemas to be matched (Do, Melnik, and Rahm, 2003; Berlin and Motro, 2002). Moreover, as modern systems become more complex and integrate complex databases and applications, their schemas become larger, increasing the number of matches to be performed. This problem leads to research towards (semi-)automated schema matching, which reduces the amount of effort and time spent on finding matches (Rahm and Bernstein, 2001).

Automated schema matching approaches all have their collection of **match operations**, which receives different types of data and outputs the mapping of the correspondences between the elements (Do and Rahm, 2002). The mapping gives an indication of which elements of the input schemas logically correspond to each other. Match operations can vary from simple similarity measures, such as Jaccard Similarity, to complex algorithms involving machine learning, each exploiting different characteristics of the data. A vast majority of the proposed solutions for automated schema matching exploit various types of schema information and characteristics of data instances, e.g. element names or data types (Bergamaschi et al., 2001; Madhavan, Bernstein, and Rahm, 2001; Milo and Zohar, 1998). A general-purpose automated schema matcher is generally viewed as infeasible, but there is a recognized need for automated schema matchers (Blake, 2007). Studies focus therefore more on specific cases, such as only finding matches for XML data, or finding matches on the web data only (Blake, 2007).

The match operator of the schema matching component have two types of selection criterion available according to Castro Fernandez et al., 2018:

- **Property constraints:** *selects the relevant data sources based on the attributes or properties of the schema (schema similarity).*
- **Relationship constraints:** *selects the relevant data based on relations between other schemas with e.g. structure of the schemas (content similarity).*

These two selection criterion allow for a classification of the two main classes in automated schema matching approaches (Koutras et al., 2021): **Schema-based matching** and **Instance-based matching**. A taxonomy was created by Rahm and Bernstein, 2001 and later extended by Shvaiko and Euzenat, 2005. The extended taxonomy on schema matching, acknowledged by Bernstein, Madhavan, and Rahm, 2011, is shown in figure 2.2. The taxonomy will be further elaborated in the following two sections, split in the two biggest classes in schema matching: *schema-based* and *instance-based matching*.

2.3.1 Schema-based matching

Automated schema-based matching is a schema matching approach based on the schema similarity mentioned in section 2.2. To detect relatedness between schemas using schema-based matching methods, either element-level and/or structure-level matching techniques can be applied (Shvaiko and Euzenat, 2005).

Element-level approaches

Element-level matching techniques solely analyzes entities on schema-level in isolation, disregarding the relationships (Rahm and Bernstein, 2001). Element-level matching approaches can further be divided into two main categories: *Syntactic* or *External*.

Syntactic approaches characterizes themselves by interpreting the structure of the input as following some algorithm. Syntactic approaches have three identified categories:

- **String-based** approaches exploit linguistic schema information, such as names and descriptions to determine the relatedness (Hai, 2007). These approaches assume that similar strings are more likely to denote the same concepts. Some popular string-based approaches which are used in schema matching are: *prefix*, *suffix*, *edit distance* and *n-gram*.
- **Language-based** approaches are based on natural language processing (NLP) and exploit morphological properties of the schema. Strings are regarded as words in some natural language. Some known language-based approaches are: *tokenization* (Wang, Li, and Feng, 2014), *lemmatization* and *elimination*.
- **Constraint-based** approaches exploit information regarding internal constraints being applied to the definition of the entities e.g. types of attributes, cardinality of attributes, ranges of instances and keys (Zhao and Ram, 2007). Known constraint-based approaches compare *data types* and *multiplicity*.

External approaches are techniques which exploit auxiliary resources of a domain and common knowledge to interpret the input. There are three main categories for external approaches:

- **Linguistic resources** can be used to match words based on linguistic relationships such as synonyms and hyponyms. There are two linguistic resources currently identified: *Common knowledge thesauri* and *Domain specific thesauri*.
- **Alignment reuse** approaches exploit alignments of previous matched schemas. The motivation lies within the intuition that schemas to be matched are similar to already matched schemas, especially if they are derived from the same domain. Alignment reuse can be more effectively applied when the match problem is decomposed into smaller sub-problems.
- **Upper level formal ontologies** are logic-based systems (semantic), allowing the matcher exploit the ontologies based on analysis of interpretation. Some resources of this approach are *suggested upper merged ontology (SUMO)* or *descriptive ontology for linguistic and cognitive engineering (DOLCE)*.

Structure-level approaches

Structure-level matching techniques use the structural information about the database schemas to determine the relatedness between schemas (Bernstein, Madhavan, and Rahm, 2011). Unlike element-level matching techniques, structure-level matching techniques only have constraint-level information available (Alwan et al., 2017). Structure-level approaches can, unlike element-level approaches, further be divided into three main categories: *Syntactic*, *External* and *Semantics*.

Syntactic approaches for structure-level information are based on graph techniques. Schemas are transformed into graphs and tried to be matched based on structure, or path relations to determine the relatedness (Nguyen et al., 2014; Madhavan, Bernstein, and Rahm, 2001). Some known syntactic approaches for structure-level information are:

- **Graph-based** approaches determine the similarity by comparing the position of the pair of nodes from the two schemas in the graph structure. The intuition is that the two nodes from two schemas are likely similar if the neighbors are similar. A known graph-based approaches is: *graph matching*. Similarity is often computed using graph elements as *children, leaves* and *relations*.
- **Taxonomy-based** approaches are also graph algorithms only considering specialization relations. The intuition is that often relationships denoting '(A) is a (B)' connects terms that are already similar, therefore the neighbours are also likely similar. Some known taxonomy-based approaches are: *bounded path matching* and *super(sub)-concepts rules*.

External approaches for structure-level information are similar to the element-level information. The difference lies within the information provided by external sources. Known external approaches for structure-level information is:

- **Repository of structures** is an external source which stores schemas and their fragments with pairwise similarities between them. To be matched schemas will have their structures compared with the known structures in the repository to determine the similarity.

Semantic approaches use some formal semantics such as model-theoretic semantics to interpret the input. There is only one category identified for semantic approaches in structure-level information:

- **Model-based** algorithms handle input based on semantic interpretation. They are well grounded deductive methods and some of the identified approaches are: *Propositional satisfiability (SAT)* and *DL-based techniques*

2.3.2 Instance-based matching

Automated instance-based matching is schema matching based on content similarity as mentioned in section 2.2. To detect relatedness in schemas using an instance-based matching approach, one needs a correlation measure that can handle different data types such as real-valued numerical, discrete numerical, and categorical since the information stored in databases are of different formats (Nguyen et al., 2014). Additionally, an instance-based matching method should have a similarity function which can measure the similarity of two elements and determine whether they are related or not (Mehdi, Ibrahim, and Affendey, 2012). The essence of instance-based matching is quite simple: use instance data to determine the matches (Duan et al., 2012).

Similar as in schema-based matching, instance-based matching also makes use of element-level information, and therefore also has syntactic and external approaches which are applicable (Alwan et al., 2017). The key difference between schema-based is that the elements in schema-based matching does not include instance data,

whereas the elements in instance-based does include instance data. So if we look at figure 2.1, we can for instance see that the identifiers of all the movies all adhere to a specific format: starting with #MV. This information in the instance data can therefore be used to determine similarities between columns through for example string-based methods.

A typical approach for instance-based matching would be using pair-wise analysis to detect relatedness between two columns; that is, comparing all pairs of data instances between two columns (Zhu et al., 2019). The relatedness can be measured through similarity metrics like Jaccard similarity or Cosine distance. However, as databases keep on growing in time, the search space also increases exponentially, resulting in unacceptable run time (Dong et al., 2020). Performing exhaustive pair-wise analysis on all available schemas would therefore be infeasible due to the nature of growing data. Additionally, majority of literature focuses on equi-joins for pair-wise analysis, which does not capture the semantics or typographical errors of instances (Dong et al., 2020). This leads to several studies on non-equi-joins, such as string transformations (Zhu, He, and Chaudhuri, 2017), statistical correlation (He, Ganjam, and Chu, 2015) or even approximation techniques to reduce the time complexity (Dong et al., 2020).

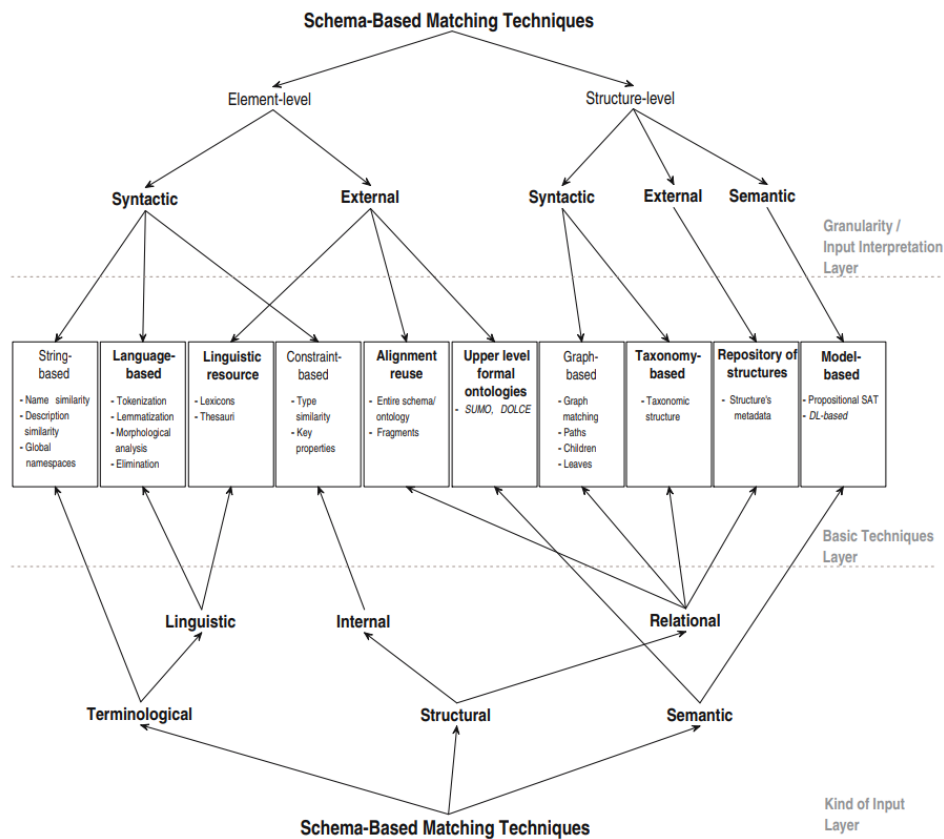


FIGURE 2.2: An extended taxonomy of schema matching methods described by Shvaiko and Euzenat, 2005. The taxonomy is divided into two layers: *Granularity/input interpretation* and *Kind of input* layer. Granularity/input interpretation layer shows what kind of information is being used by the schema matching technique, whilst the kind of input layer shows the types of possible inputs.

2.4 Combining matchers

No single schema matching technique can effectively match schemas, whilst often a combination of schema matching techniques produces more superior results, which is acknowledged by researchers (Blake, 2007). The challenges of schema matching revealed that it is clear that simple matchers or single matching algorithms that use one matching technique or a few ones, will not achieve the good results that we wish, as there is a wide variety and heterogeneity between real life schemas (Al-Ghanim, Noah, and Sembok, 2011). Individual matchers are much faster in process but has a disadvantage that it is only useful in certain cases (Sutanta et al., 2016). Using multiple matchers to account for the shortcomings of each matcher often improves the quality of the similarity score (Peukert, Maßmann, and König, 2010). Different approaches can be used to combine the similarity scores of multiple matchers. However, combining multiple matchers would result in either one of the two known classes: *hybrid matcher* and *composite matcher* (Özsu and Valduriez, 1999).

Hybrid matchers combine multiple matchers within one algorithm. Elements from two schemas can be compared using a number of matchers within one algorithm to determine their overall similarity (Özsu and Valduriez, 1999). The advantage of using a hybrid matcher for combining matchers consists of the improved performance due to the reduction of operations as bad matches can be filtered. The downside however is that a hybrid matcher is typically a hard-wired combination of matching techniques, making it less flexible for different datasets (Rahm and Bernstein, 2001).

Composite matchers apply each matcher to the two schemas individually and in parallel, obtaining individual similarity scores, and then apply a method to combine these similarity scores (Özsu and Valduriez, 1999). The advantage of using a composite matcher, as opposed to the hybrid matcher, is its flexibility to construct new matchers by reusing the results. However, composite approaches scale rather poorly compared to hybrid methods as each individual matcher calculates similarity independently.

Composite matchers consist of three main components which are listed below (Lee et al., 2007):

- **Matcher** is a component which assign column-pairs similarity scores between 0 and 1 based on the above mentioned methods in section 2.3.
- **Combiner (aggregator)** is a component which merges multiple similarity scores into a single one.
- **Match Selector** is a component which determines which of the column-pairs to select as a match.

2.4.1 Matcher selection

Selecting matchers for a composite approach is a known problem within schema matching (Lee et al., 2007). Combining matchers and using their knowledge to improve data discovery requires a rational selection of matchers. Individual matchers may be imprecise, but their combination can effectively improve the quality of the matches (Do and Rahm, 2002). The ability to analyze schema elements under different aspects results in more stable and accurate similarity for heterogeneous schemas

(Embley, Xu, and Ding, 2004, Duchateau et al., 2009).

To identify different aspects of schema elements, the taxonomy described in section 2.3, shows a detailed description of available type of information. The taxonomy in figure 2.2 shows that three kinds of input types can be considered as input: *Terminological*, *Structural*, and *Semantic*. All these types of input are categorized either as element-level or structure-level information, which can be further divided into *Synthetic*, *External*, and *Semantic* information. Therefore, to cover all different aspects of the available information, different types of information should be used accordingly to the different granularity of the available information.

Research regarding data discovery methods typically implement their own matcher, by either combining or customizing existing methods (Koutras et al., 2021). Due to the lack of availability of schema matching methods in current research, the majority of the methods do not take advantage of the schema matching methods in current literature (Rahm and Bernstein, 2001; Do, Melnik, and Rahm, 2003). The reproducibility of modern schema matching methods are also very low due to the vague descriptions or required settings for parameters. Recent work from Koutras et al., 2021 implemented a selection of the state-of-the-art machines and made it accessible for public use. These include: *Coma*, *Coma++*, *Cupid*, *Similarity Flooding*, *Distribution-based*, *SemProp* and *EmbDI*.

2.4.2 Aggregation Strategies

Each composite matcher requires an aggregation strategy (combiner) to combine different matchers with each other (Do and Rahm, 2002). Matchers produce mappings of correspondences between the elements and the similarity score for those correspondences. Popular and intuitive strategies to aggregate the similarity scores of individual matchers are MIN, MAX, AVERAGE and WEIGHTED (Peukert, Maßmann, and König, 2010).

- **MIN aggregator** always chooses the minimum value of the given set of similarity scores computed by the individual matchers. The approach is very pessimistic as it requires all matchers to return high scores in order to have a good score.
- **MAX aggregator** always chooses the maximum value of the given set of similarity scores computed by the individual matchers. As opposed to the MIN aggregator, the MAX aggregator is quite optimistic since it only requires one matcher to give a high score
- **AVERAGE aggregator** computes the average score of the given set of similarity scores computed by the individual matchers. The AVERAGE aggregator assumes that the importance of each individual matcher is equal.
- **WEIGHTED aggregator** computes the weighted average score of the given set of similarity scores computed by the individual matchers. The WEIGHTED aggregator assumes that the importance of each individual matcher is not equal and therefore requires relative weights which should correspond to the expected importance of the individual matcher.

Over the years, more aggregation strategies has been developed such as SIGMOID, HADAPT, OWA and NONLINEAR, but they are all based on weights (Elshwemy

et al., 2014). For example, SIGMOID attempts to prepare matcher results by increasing high similarity values and decreasing low similarity values before computing the weighted sum, introducing more parameters to tune (Peukert, Maßmann, and König, 2010). In the research of Peukert, Maßmann, and König, 2010, it was shown that among the above mentioned aggregation strategies (excluding WEIGHTED), the AVERAGE and NONLINEAR strategy performed the best, of which the AVERAGE aggregator was more stable (also according to Do and Rahm, 2002). Studies excluded the evaluation of the WEIGHTED aggregator due to the need to set the weights manually (Peukert, Maßmann, and König, 2010; Do and Rahm, 2002). However, the studies also showed that the WEIGHTED approach had more potential compared to the AVERAGE approach due to its flexibility and essentially being capable of performing the same as the AVERAGE aggregator.

2.5 Related work

2.5.1 LSD (2001)

LSD is a **composite** approach based on machine learning predictors for XML data (Doan, Domingos, and Halevy, 2001). It consists of four major components: *base learners*, *meta-learner*, *prediction converter* and *constraint handler*. LSD operates in two main phases, a *training phase* and a *matching phase*.

The training phase is used to extract some information from each source and train the base learners by user labelled data. A base learner uses extracted information to learn the similarity based on the type of base learner. A selection of base learners should be made where different aspects of the available schema elements is taken advantage of. Before using the base learners, the user will be asked to specify mappings of the given sources. After these mappings are made, LSD extracts data instances from the sources and manufactures training data based on the given mappings. Each base learner needs training data in different aspects, therefore needing different features respectfully. The last step in the training phase is to train the meta-learner, which is in charge of combining the predictions of the base learners. The meta-learner learns the weights of each base learner by its predictions of the training examples.

The matching phase uses the trained learners to match schemas. This consists of three steps: extracting data from source, application of base learners and meta-learner, and compute mapping by the constraint handler. The matching phases starts off by extracting (instance) data from the source. Afterwards, the base learners will try to predict the similarity and use the meta-learner to combine the predictions by using the weighted sum approach. This process is repeated for each instance of the same type and the combined predictions of each instance will be combined into one single prediction by the prediction converter. After these predictions are made, the constraint handler uses domain constraints to reduce the amount of false predictions by filtering the predictions based on semantic regularities. These can be hard constraints (can not be violated) or soft constraints (minimized violation). At last, the constraint handler outputs the mappings of the matches.

2.5.2 Cupid (2001)

Cupid is a **hybrid** schema-based general-purpose schema matching strategy (Madhavan, Bernstein, and Rahm, 2001). Schemas are represented as schema trees where the nodes represent schema elements. The similarity value is calculated using two phases, linguistic matching phase and structuring matching phase.

The linguistic matching phase matches individual schema elements based on their names, data types, domains, etc. This is done by using a thesaurus to help match names by identifying short-forms, acronyms and synonyms. It involves three steps: normalization, categorization and comparison:

1. **Normalization** makes the elements comparable by using means as tokenization or expansion.
2. **Categorization** then makes groups based on concept tagging, data types and container to reduce the amount of needed comparisons by only comparing compatible categories. Categories are deemed as compatible when their name similarity does not exceed some given threshold.
3. The **comparison** step then compares the elements of the compatible categories based on name similarity as well. This results in a linguistic similarity values between elements used to determine the final linguistic similarity value.

The structural matching phase is based on the similarity of their context. It looks into the tree structure of the schema and uses part of linguistic matches to calculate the similarity between columns. The approach is based on the *TreeMatch* algorithm which determines similarity based on three intuitions:

1. Leaves of two trees are similar if they are individually similar and if their parents are similar
2. Two non-leaf elements are similar when they are linguistically similar and their subtrees are similar
3. Two non-leaf schema elements are structurally similar if their leaves are highly similar

The similarity between leaves are initialised based on their data types to a similarity value between 0 and 0.5. Setting the maximum to 0.5 allows for later increases when more of the intuitions is satisfied. This is done by using a bottom-up approach and the similarities are adjusted based on thresholds. When the similarity of the comparison is below a threshold Th_{low} , the structural similarity value will decrease. When the similarity is over a predetermined threshold Th_{high} , the structural similarity value will increase. Going over the entire tree will result in a final structural similarity. In order to combine both the linguistic and structural similarity, a weighted sum is calculated resulting in the final score.

2.5.3 Coma (2002)

Coma is a generic **composite** schema matching approach allowing the user to combine multiple matchers in a flexible way (Do and Rahm, 2002). Coma takes input schemas and converts them into a directed acyclic graph where schema elements are graph nodes connected by directed links of different types. Once the schemas are

imported and processed, the match process takes place in one or multiple iterations, depending on the mode that the user had chosen (automatic or interactive). For our case, the automatic mode is chosen as we use several matchers to find meaningful relationships, which already takes quite some time. Each match iteration consists of three phases:

1. An (optional) user feedback phase
2. The execution of different matchers
3. The combination of individual match results

The first phase allows the user to determine the match strategy. This includes choosing the different matchers and the strategy to combine the similarity values. In automatic mode, the settings are pre-determined.

The second phase is the execution of the chosen matchers. The matchers are all executed in parallel and the similarity values of each matcher are stored in a so-called similarity cube. This cube is a $k \times m \times n$ cube where k is the number of chosen matchers, m is the schema elements of the first schema and n is the schema elements of the second schema.

The third and final step is the combination of the similarity values in the similarity cube. This is done by aggregation of the similarity values for every element pair. Coma knows different aggregation techniques as: Max, Weighted, Average and Min. These simple aggregation methods are included in the evaluation of Coma, whereas the Weighted strategy was excluded since they do not want to assume the importance of each individual matcher.

2.5.4 Embley et al (2002)

The work of Embley, Jackman, and Xu, 2002 is a **composite** approach which combines three matchers by averaging the similarity scores. The three matchers all target different information as we have a terminological matcher (e.g. synonyms, word senses and hypernym grouping), data-value matcher (e.g. average values, variances and string lengths) and target-specific matcher (e.g. regular expression). Each matcher calculates a similarity value between source and target schemas and will be combined into a combined similarity score, which is the average. The averaged score will be refined using structural tests and calculated the product of the chosen structural features for the final scores (e.g. importance feature, distance feature). The path is chosen with the highest scores for each adjacent object and averaged, which represents the correlation score between two schemas. The approach obtained a recall and precision scores of 90% and above.

2.5.5 MKB (2003)

Mapping Knowledge Base (MKB) is an extension of the LSD system where the previous matching tasks are not limited to a single mediated schema but can be used for any pair of schemas (Madhavan et al., 2003). The MKB system is a **composite** approach which includes five base learners: name learner (exploit names of elements), description learner (exploit available text descriptions), instance learner (exploit data instances), data type learner (exploit data types), and structure learner (identifies elements which frequently co-occur with an element). Similar to the LSD

system, MKB also adopts a meta learner for learning the relevance of each base learner. However, as opposed to the LSD system which adopts a linear combination for combining the learners, MKB adopts the sigmoid or logistic function for more flexibility in choosing base learners. The experiments show that combining the learners with the meta-learner and applying the MKB shows potential as it improves the recall compared to the baseline.

2.5.6 Harmony (2006)

The system behind Harmony is also a **composite** approach which combines linguistic matchers by iteratively updating the weights of the matchers by user feedback or performance (Mork et al., 2006). Harmony processes the schemas linguistically to conventional schema matching techniques and use different linguistic matchers to determine similarity. These matchers include: bag of words (for names), bag of words (with thesaurus expansion), edit distance (for names), acronym matcher. Each matcher gives a confidence score in the range of $[-1, +1]$, rather than the traditional range of $[0, 1]$ where -1 denotes no correspondence and $+1$ denotes definite correspondence. The individual matcher scores are combined by weighting the confidence of each matcher by their score and the total performance of each matcher (in an iterative way). The implemented Harmony GUI is an user interface which filter the confidence scores by either setting a threshold, categorizing the links as human-generated or machine-suggested, or only display confidence scores of $+1$ supporting the user to iteratively mark the matches and adjust the weights.

2.5.7 YAM (2009)

Combining similarity values of different schema matchers resulted in different results to a given schema matching scenario. YAM is another **composite** approach which tries to learn a matcher (dedicated matcher) over a large set of matchers and similarity measures to determine the weights of the matchers (Duchateau et al., 2009). The dedicated matcher is trained over a small training set by selecting similarity measures for maximum correctly classified correspondences. Then afterwards, similarity measures which solve harder cases are involved. The dedicated matcher selects the matchers by the accuracy of the matchers on the test data (f-measure). YAM very much depends on the diversity of the selected matchers. Compared to COMA++ and Similarity Flooding (two high-performing traditional matchers), learning a matcher performed better when trained on relevant matching scenarios.

2.5.8 Data tamer (2013)

Data tamer is a data curation system which includes a **composite** schema integration approach to determine similarity between schemas available from online sites (Stonebraker et al., 2013). The main goal of data tamer is to group sites which describe the same entity type. The data tamer system consists of four matchers: fuzzy string comparisons, tokenization, minimum description length, and Welch's t-test. The amount of matchers can be extended through API's but are not included in the system. Each individual matcher computes a similarity score in the range of $[0, 1]$, and will be aggregated using weights. The system allows for crowd-sourcing the training data for each matcher and human interaction in the sub-processes which are deemed effective for enterprises for scaling.

2.6 Conclusion

Data discovery is an inevitable process for modern enterprises. Tasks surrounding data discovery are: searching for joinable tables, augmenting tables with extra entries/attributes, and finding similar tables. To address these tasks, schema matching, the dedicated component for data discovery, has been developed. Over the years, different schema matching techniques were developed resulting in several sub-classes of schema matching approaches exploiting different parts of (schema) information. Rather than using one schema matching approach, combining multiple schema matching approaches is the way to go as it exploits more parts of the schemas and compensates for an individual matcher's shortcoming. There are two identified types of combination approaches: hybrid or composite matchers. Hybrid matchers are better for performance, whilst composite matchers are more flexible. Both should include a wide range of matchers exploiting different parts of available information to obtain better results, which is a challenge on its own. A disadvantage of hybrid approaches is that they are hardwired and therefore less robust for different scenarios. The disadvantage of composite approaches is that the performance is scaling poorly. Composite approaches need an additional aggregation strategy to combine the scores of individual matchers. Amongst different researched aggregation strategies, the weighted average is the most promising aggregation method according to several studies. Different work shows that going for a composite approach should include matchers exploiting different parts of schema information and should aggregate the matchers using weights to indicate the importance.

Chapter 3

Methodology

3.1 Approach overview

For our methodology, we will make use of a composite approach (described in section 2.4) for combining different matchers. Going for a composite approach allows us to flexibly add more matchers and analyze the contribution of each matcher towards the final similarity score. We therefore adopt the same strategy as Coma (Do and Rahm, 2002): run a chosen set of matchers and store them for (re)usage. Combination techniques in research for combining multiple matchers do not have analysis on the **weighted** approach, which is supposedly the best performing aggregation strategy (Tu and Yu, 2005).

We therefore extend the aggregation approach (weighted) by empirically proposing different techniques for determining the importance of each matcher. We conduct an exploratory research in search for positive results for a generic way of combining matchers. We explored different techniques trying to achieve better predictive results compared to the individual matchers. Additionally, we implement the MAX aggregator which allows us to compare it to the aggregation method which has been evaluated. For each method, the preliminary results does not show improvements, which led us to explore different methods.

Our methods primarily rely on the weights of the matchers indicating the importance of each matcher. Therefore the way we reduce the weights is a very important matter for each method. Additionally, we only focus on CSV formatted data for simplicity sake as we want to observe the aggregation approach without added complexity. Aside from the weights and data format restrictions, our approach also makes use of **agglomerative hierarchical clustering** to cluster related columns together. Clustering allows us to find related columns through witness columns (Zhang et al., 2011). E.g. Columns A, B and C are related, all have high similarity scores except for the column-pair (A, C), which has a similarity score of 0. Through clustering, we allow column B to be an intermediary column to cluster all columns A, B and C together into one cluster due to the high similarities. The output of our methods would be a set of clusters of where each cluster has columns which are *inter-joinable* with each other (related). We use clustering by setting the distances between columns equal to $1 - score_{weighted}$, which sets the distance between highly similar columns closer to 0 and highly dissimilar columns closer to 1. The distances between columns will always be in the range of 0 and 1, as all the weighted scores are already normalized.

3.2 Methods

As our methodology tries to find observe the aggregation approach, we have made a selection of matchers for each approach (shown in table 3.1). The selection these matchers are based on the credibility of its publication and the type of information they use to match schemas. The chosen 5 matchers all explore different aspects of available data such as distribution, attribute overlap, value overlap, data type, and semantic overlap (Koutras et al., 2021) and have relative acceptable run time.

| | Coma | Coma++ | Cupid | Distribution based | Similarity flooding |
|------------------------------|------|--------|-------|--------------------|---------------------|
| Multiplicative weight update | x | - | x | x | x |
| Reinforcement learning | x | x | x | x | x |
| Linear programming | x | x | x | x | x |
| Machine learning | x | x | x | x | x |

TABLE 3.1: Selected matcher for each method

3.2.1 Multiplicative weight update method

The multiplicative weight update method is our firstly implemented method. For this method, we made the assumption that the similarity values ranging from $[0, 1]$ actually depicts the likeliness of the columns being related as described by Do and Rahm, 2002. Therefore, we **only included** similarity scores of > 0.5 as those scores lean more towards 1, meaning that it is more likely to be related than unrelated. This is what we would like to call the **cutoff threshold of 0.5**. Similarity scores of < 0.5 are all rewritten as 0. The formula below shows the nature of the cutoff threshold for a similarity score x :

$$Cutoff(x) = \begin{cases} x & \text{if } x \geq 0.5 \\ 0, & \text{otherwise} \end{cases}$$

Determining the weights

When we run all the (chosen) n matchers in parallel, we will obtain n similarity scores for each column-pair across the available datasets. With the cutoff threshold of 0.5 introduced above, the scores given to each column-pair must be either 0 or in the range of $[0.5, 1]$.

To determine the weights for each matcher, we use the multiplicative weight update method, more specifically: **the weighted majority algorithm** (Arora, Hazan, and Kale, 2012). This method is designed for determining the weights of experts in decision making, but can also be used for our use case. The algorithm works in a form of reinforcement learning, where the essence is to punish experts who are wrong about the predictions and decrease their weights based on a learning rate. The pseudo-code of the algorithm is depicted in algorithm 1.

Algorithm 1 Multiplicative weight update method for n matchers and m column-pairs

Inputs:

$$W = \{w_0, w_1, \dots, w_n\}$$

$$T = \{t_0, t_1, t_m\}$$

Initialize:

$$w_i \leftarrow 1, i = 0, \dots, n$$

$$\epsilon \leftarrow 0.05$$

▷ Learning Rate

for $t \in T$ **do**

for $i = 0, 1, \dots, n$ **do**

$similarity \leftarrow$ similarity score given by expert i for column-pair t

if t is a related pair & $similarity = 0$ **then**

$$w_i \leftarrow \text{MAX}(w_i \cdot (1 - \epsilon), 0.0)$$

else if t is an unrelated pair & $similarity \neq 0$ **then**

$$w_i \leftarrow \text{MAX}(w_i \cdot (1 - \epsilon \cdot (-similarity)), 0.0)$$

end if

end for

end for

3.2.2 Reinforcement learning

For this method, we took information from the preliminary experiments of the multiplicative weight update method and tried to improve the overall method through the observations. The identified observations are:

- **OB1:** All matchers have different similarity score distribution
- **OB2:** Majority of the matchers are schema-dominant, potentially leading to bias for schema information
- **OB3:** Matchers who has assigned a lot of scores ≥ 0.5 lead to weights close to 0
- **OB4:** Multiplicative weight update method only considers individual performance for the matchers

From observation **OB1** we find that the cutoff threshold of 0.5 is not reasonable as a 0.5 could have a very different meaning for matcher X compared to matcher Y. This leads to the end of using cutoff thresholds.

Observation **OB2** shows that we use more schema-related information compared to the instance-related information. We therefore introduce the matcher: coma++, which has the same essentials as coma, but includes support for instance-based matchers (Engmann and Maßmann, 2007).

Observation **OB3** and **OB4** are targeted at the derivation of the weights for the matchers. The nature of the multiplicative weight update method is to **only** punish the wrong predictions of the matchers and award **no rewards** for correct predictions. Additionally, the punishment is only considering the individual performance of each matcher. As the final similarity score is aggregated through the weighted approach, there should be a collaborative correlation between the matchers, which is excluded from the derivation of the weights.

Determining the weights

In order to learn the weights collaboratively, we need to look more into group decision making. In group decision making approaches, a classification can be made (Koksalmis and Kabak, 2019). The two main classes within group decision making approaches are: *process-oriented* or *content-oriented* approaches. As our reinforcement learning approach is based on the content of the matchers, we dive deeper into content-oriented approaches.

Content-oriented approaches can also be divided into four main classes: *Similarity-based*, *Index-based*, *Clustering-based* and *Integrated*. The most suitable class for our use case is a tweaked version of similarity-based approaches. Similarity-based approaches determine the weights based on how similar the decision makers are compared to either each other or the solution.

In our case, we do not want to compare it to the aggregated score, but to the truth. Which means that if there is a relation of which we know that it belongs together, we know that the best similarity score for that given relation would be 1 and for a relation which does not belong there, a similarity score of 0. Knowing this, we design our reinforcement learning approach for determining the weights of the matchers by comparing the contribution of the matcher.

To calculate the contribution of each matcher for the aggregated similarity score of a column-pair, we seek to determine what effect the matcher had on the aggregated similarity score and whether it pushes the aggregated similarity score to the truth when included. The contribution of each matcher can be determined with the following equation where S denotes the set of experts and i the expert whose contribution we are calculating for a given column-pair:

$$\text{Contribution}(S, i) = \text{Weighted_Average}(S) - \text{Weighted_Average}(S - \{i\})$$

Note that the contribution can either be a negative or positive number. Whether a positive number is desired, depends on the truth of the column-pair. When the truth is that the given column-pair is a related pair, a positive contribution depicts that the given matcher contributes positively to the aggregated similarity score. When the truth of the given column-pair is an unrelated pair, a negative contribution is desired. We adjust the weights of each matcher depending on its degree of contribution to the score moving towards the truth with the following equation where the truth is either 1 (denoting a related pair) or -1 (denoting an unrelated pair) and ϵ denoting the learning rate:

$$\text{New_Weight}(i) = \text{Weight}(i) + ((\text{truth}) \cdot (\text{Contribution}(S, i) \cdot \epsilon))$$

By adjusting the weights every time a new column-pair gets labelled by the user, we reward the matchers who help to increase the aggregated similarity score for related pairs and punish the matchers who do not decrease the aggregated similarity score for unrelated pairs.

3.2.3 Linear programming

Based on the preliminary experiments for the reinforcement learning approach, we identified the following observations:

- **OB5:** Reinforcement learning approach does not learn the weights near optimal
- **OB6:** Mismatched similarity score distributions of different matchers potentially cause extra difficulty for maintaining higher scores for related column-pairs

From observation **OB5** we find that dynamically updating the weights of the matchers through our reinforcement learning approach does not result in optimal weights. Therefore, we will view the weight problem as a mathematical problem and solve it mathematically using linear programming.

Observation **OB6** show potential difficulty introduced due to the mismatching similarity score distributions. The mismatching could potentially cause extra difficulty as matchers who give lower scores in general will have difficulty keeping the aggregated similarity score up. We therefore introduce to **normalize** all the scores with the given similarity scores we have, to attempt to match the distributions of other matchers.

We will have two versions to experiment on. The first version will be called *Linear Programming*, where we try to optimize the weights using linear programming and exclude the normalization. The second version will be called *Normalized Linear Programming*, which includes the normalization of the similarity scores. This allows us to show the effects of normalizing the similarity scores.

Determining the weights

Linear programming is a method which tries to achieve the best outcome for the objective function in a mathematical model whose requirements are represented by linear relationships. In order to use linear programming in our methodology we need to determine what the objective functions are and what the mathematical model would be.

We are trying to optimize the aggregated similarity score by adjusting the weights of the matchers. The variables which linear programming tries to optimize would therefore be the weights of the matchers.

The objective function would either be to **maximize** the scores of the related column-pairs or **minimize** the scores of the unrelated column-pairs. As we use clustering to find the relationships, maximizing the related column-pairs would suit the most as the higher scores are clustered first. Hence, we use maximization for our objective function and try to keep the scores of unrelated column-pairs below a certain threshold.

The objective function for our problem can be defined as shown below where w denotes the list of weights, w_i denotes the weight of matcher i , and S_i denotes the sum of all similarity scores provided by matcher i for all the (known) related column-pairs:

$$\text{Maximize } \sum_{i=0}^{|w|} w_i \cdot S_i$$

The mathematical model we adopt for our methodology will contain the following variable: the weights of the matchers w_i . As mentioned before, these weights can be

any real number between 0 and 1. Hence we can write the constraint of the weight variable for each matcher i :

$$w_i \in \mathbb{R}$$

We also need to have a constraint for the variables as they have to be bounded. Since we use the weighted aggregation method for the calculation of the final similarity score, we can bound the sum of the weights to 1, resulting in the following constraint, where w is the list of weights:

$$\sum_{i=0}^{|w|} w_i = 1$$

As we have included the related column-pairs into the objective function, we need to include the unrelated column-pairs as constraints to the mathematical model. To incorporate this, we define for each unrelated column-pair j a constraint, where $score$ is a list of scores for unrelated column-pair j :

$$\sum_{i=0}^{|w|} w_i \cdot score_i < t$$

The value t is the threshold, which should be as low as possible in order for the scores of the false matches to be minimized, however, this would directly influence the best score for the true matches. Therefore, it is recommended to try to set the threshold in a small range of the lowest possible value and find what is most appropriate. A very high value, would just result in giving all matchers a weight of 0 except for one, which has a weight of 1.

3.2.4 Machine learning

Based on the preliminary experiments for all the previously mentioned approaches, we made the following observation:

- **OB7:** Plain weights are unable to acquire good representative aggregated similarity scores for both related and unrelated column-pairs

From observation **OB7** we determined that the relation between the weights and the good scores are probably much more complex than anticipated. This makes learning the weights also very complicated and therefore worthy to explore the possibility of using machine learning techniques to learn whether the column-pair is related or not.

Machine learning model

Exploring the potential in the world of machine learning for combining multiple matchers could be done through a simple regression model, rather than a complicated neural network model. Neural network models are hard to explain due to the black box nature, whilst regression models are more explainable (Burkart and Huber, 2021). For classification, Support Vector Machine (SVM) has been a very powerful tool for solving binary classification (Cervantes et al., 2020).

SVM is well known for its ability to learn well with only a very small number of features, its robustness against the error of models, and its computational efficiency compared to other machine learning methods such as neural networks (Gholami and Fakhari, 2017).

For our SVM model, we use the similarity scores of each matcher as features. If we have n matchers, then we will have n features, where the values of those features are the achieved similarity scores for the respective matcher. Our SVM model will try to classify the column-pair as **related** or **unrelated** based on the similarity scores of the matchers (features). We use the radial basis function kernel, kept the C value neutral at 1 and used the scaling gamma value mentioned in the documentation of sklearn¹.

¹<https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html>

Chapter 4

User Interface

To ease the user experience of using our methodology and obtain valuable insights, an user interface is implemented¹. This section describes the user interface in more detail and its usage.

4.1 Configuration

Our methodology consists of quite a few configuration points, including the selection of matchers, the cutoff threshold (mentioned in chapter 3.2.1), the selected method and the data space. To easily allow users to adjust the different configurations such that it is easier to experiment with different settings, we implemented an user interface which allows the user to set the configurations in a 4-step form. After setting the configurations, we will perform all the steps for retrieving the similarity scores with the chosen settings. The user interfaces are shown in appendix A.

The first step is to allow the user to select the matchers which are included in the methodology. The user interface consists of a list of switches, which allows the user to either include or exclude the prompted matchers based on whether they are switched on.

The second step is to allow the user to select the cutoff threshold. The form prompts a slider allowing the user to set the threshold between $[0, 1]$.

The third step is to allow the user to select the method to determine the weights. These are the methods described in chapter 3. The methods are depicted as radio buttons, allowing the user to only choose one method.

The fourth step is to let the user choose the data space which the methodology will be ran on. These data spaces are the directories in the *tables* folder within the project.

After setting these configurations, we run the matchers individually on the given dataset and store those similarity scores. After this process, the user can still change the cutoff threshold and the method described in chapter 3 through our user interface, shown in appendix A figure A.5.

4.2 Labelling

All our methods are based on labelled relationships between the schemas. To make it easier for the user to label the relationships, we provide an user interface where

¹<https://github.com/wanghaojiang1/thesis>

we allow the users to create ground truth clusters. Since the ground truth is the minority class, we can simply label all the column-pairs once we know the ground truth. These clusters are regarded as the only true relationships and therefore be labelled as positive relationship. The rest of the data will be regarded as negative relationships. The user can therefore label all the relationships by only labelling the ground truth, reducing the time needed to label all column-pairs manually (minority group). The user interface for this functionality is shown in appendix B.

4.3 Evaluation

To ease the evaluation of our method, we implemented an user interface which shows the clusters (appendix C figure C.1), allows the user to tweak the threshold, provide the user a dendrogram and show the metric scores of the given threshold (shown in appendix C figure C.2). Allowing the user to review the clusters enables them to see and experience the effectiveness of the approach. Showing the corresponding dendrogram gives the user insight on what gets clustered first and which threshold is needed. Therefore, by allowing the user to change the threshold, the user can tweak the clusters to a certain extend and observe which threshold leads to the best accuracy. Additionally a page with all the metric scores of the formed clusters are saved in a data table (appendix C figure C.3).

Chapter 5

Experiments

This section presents the experimental evaluation of our proposed approaches. The experiments will revolve around the comparison between the individual matchers and the different methods which we mentioned in chapter 3.

Our goal is to evaluate the robustness and effects of our proposed methods on different datasets and make them comparable with each other. To evaluate the robustness, we conduct experiments on five different datasets, whereas two datasets are fabricated using a fabrication tool. To compare the effects of our proposed methods revolving around the weighted aggregation method, we implement a basic MAX aggregation method to show the improvements. And in order to achieve our goal, we designed metrics and use them to make the results of the experiments comparable (described in section 5.2).

5.1 Data

Evaluating the methodology requires a sufficient amount of labelled data. In order to evaluate our methodology properly, open-source datasets are used and used to fabricate new datasets. The positives of fabricating the data is that we can simulate joinable/unionable schemas and we know the ground truth. The negatives of fabricating the data is that it does not represent a real-life situation. In our experiment, we will be using TPC-H, IMDb and ING dataset to evaluate our methods. Additionally, we will have two fabricated datasets which originates from the TPC-H and IMDb datasets respectively. We will use the TPC-H, IMDb and their fabricated version to evaluate our methodology and report findings. We will then use the ING dataset to confirm the findings and effect of our methodology on real-life data.

Fabricating data can create interesting situations which tries to depict a more complex dataset. By splitting a base tables, the data can lead to several different cases. Fabricating data allows splitting the table horizontally and vertically. Splitting a table horizontally means that we keep the columns as it is, but split the rows into two separate tables where the rows can still have some overlap. This creates a *unionable* situation. Vertically splitting creates two tables where the rows are kept, but the columns are split into two tables which has at least one column in common, which serves as the primary key. This creates a *joinable* situation. The two types of splitting can used interchangeably. Additionally, noise can be introduced in the data instances and/or column names to introduce extra complexity. This allows for a more representative dataset for real life scenarios as identical names for every instance or column name is only present in an ideal world.

5.1.1 TPC-H

TPC-H¹ is a benchmark tool used for decision support. The data model of TPC-H simulates an industry which must sell, or distribute a product worldwide. We used TPC-H version 3.0.0 with a scale factor of 1. Although TPC-DI is more suitable for data integration tools, we chose for TPC-H over TPC-DI since we need to know the relationships between different tables beforehand. Running our methodology over the eight tables from TPC-H, resulted in some insight over how well our method can find relationships for simple and small sets of columns.

Fabricated TPC-H

The fabricated TPC-H dataset (sixteen tables) is created by splitting all the TPC-H (eight) tables horizontally and vertically. Each split table contains noise in approximately 30% of the rows. The two split tables (T_1 and T_2) have approximately an overlap of 50% in the rows, meaning that there is enough evidence for a correlation. Additionally, the column overlap vary between one or three columns and all the column names contain noise. The column names of T_1 are unaltered whilst the column names of T_2 contains the noise. After splitting the tables, thirteen relationships are forged due to column overlap. When we combine these forged relationships with the actual relationships, we get a total amount of 37 relationships which our method needs to find.

5.1.2 IMDb

IMDb² is an online Internet Movie Database which contains information related to movies, television series, home videos, video games and streaming content online. It is a open database which differs each day as data keeps growing over time. The used dataset of IMDb is downloaded at 11th of May in 2022. as the database is humongous in terms of rows, it has been decided to trim the data as our used computer for evaluation is incapable of handling such big amount of data. In order to keep the foreign key references, the data will first be ordered based on the keys and then trimmed down to having 100.000 instances per table at max. The available tables of the IMDb dataset have two kinds of keys: tconst and nconst, where tconst denotes the key of the movie and nconst denotes the keys of individuals (writers and directors etc.).

Fabricated IMDb

The fabricated IMDb dataset (fourteen tables) is created by splitting all the IMDb (seven) tables. The amount of instances are too big. Therefore, all the tables has been trimmed to 100.000 instances before the fabrication process. Each IMDb table is split horizontally and vertically into two separate tables contain a row overlap of 80%. Additionally, similar as the fabricated TPC-H, the columns contain noise. No row noise is introduced in the fabrication of the tables. These fabrication settings are drawn from the preliminary results from experiments on the fabricated TPC-H dataset.

¹<https://www.tpc.org/tpch/default5.asp>

²<https://www.imdb.com/interfaces/>

5.1.3 ING

The ING³ dataset is an actual dataset used for their discovery application, UCMBD TADMM. The dataset consists of 2045 tables of which the majority are skeleton tables, consisting of no instance data. Due to the database being stored on a virtual machine, extracting those data requires a file transfer service for security reasons. The file transfer service is limited to 3GB files, whereas transferring files over the limit results in failed transfers. Therefore, a selection of the tables has been made. In the selection, all foreign keys are preserved and some skeleton tables are added as well, since excluding those would result in explicitly removing potential false positive column-pairs. In total there will be 23 tables with varying amount of columns and instances. The total column-pair combinations across the columns of the tables results in a total of 46713 column-pairs.

5.2 Metrics

To evaluate the methodology on how good it finds relationships, we need metrics to be able to compare the results. The common evaluation metrics used in data discovery, is adopted from information retrieval, namely: **precision**, **recall**, and **f-measure** (Bellahsene et al., 2011). Precision reflects the share of real correspondences among all found ones, whilst recall specifies the share of real correspondences that is found (Do, Melnik, and Rahm, 2003). Solely using precision or recall can not accurately evaluate the methodology as both can easily be maximized at the expense of the other metric. This calls for a combined measure, which is the f-measure where both precision and recall are equally as important (Do, Melnik, and Rahm, 2003).

Our methodology produces clusters where each cluster consists of columns, which represents an attribute. Therefore, we can calculate two kinds of metrics: one to measure the quality of the clusters and one for the relations (in a traditional fashion). We also introduce Valentine metrics to evaluate the score quality of the aggregated similarity score (without clustering) to cover the entirety of the methodology.

5.2.1 Cluster Metrics

The metrics with clusters work a little different compared to the traditional metrics. Whereas traditionally, each entry represents a single entry, with our methodology, each entry represents a cluster which consists of multiple columns. Hence, to determine the metrics accurately, we adopt the same strategy as Zhang et al., 2011 to fit our use case more. The metrics that we will use consists of two parts, one for the entire list of clusters A and one for the individual cluster A_i . Only clusters with more than one element is considered an attribute which should be included in the calculations. Singular clusters (clusters with 1 column) can lead to deceptive results as the majority of the columns are probably singulars. The metrics used to determine the scores of the individual cluster A_i are the traditional metrics: **precision** and **recall**, which is known within the community. To be able to calculate these metrics, the ground truth cluster T_j is necessary. This is the cluster known in the ground truth where A_i has the most overlap with. The metrics for each individual clusters are defined as:

³<https://www.ing.com/Home.htm>s

$$Precision(A_i) = \frac{|A_i \cap T_j|}{|A_i|} \quad Recall(A_i) = \frac{|A_i \cap T_j|}{|T_j|}$$

In order to have the metrics correspond to the entire list of clusters A , the scores of the individual clusters should be combined. To do this, the sum of all the individual clusters will be divided over the amount of clusters in the ground truth T . These metrics are defined as:

$$Precision(A) = \frac{\sum_{i=1}^{|A|} Precision(A_i)}{|T|} \quad Recall(A) = \frac{\sum_{i=1}^{|A|} Recall(A_i)}{|T|}$$

Since recall can easily be maximized at the expense of poor precision and vice versa (Do and Rahm, 2002), an additional metric (f-measure) was introduced to tackle this problem using both metrics. The f-measure metric is defined as:

$$F - measure(A) = \frac{2 \cdot Precision(A) \cdot Recall(A)}{Precision(A) + Recall(A)}$$

5.2.2 Traditional Metrics

The metrics used for data discovery often revolves around the relationships. The more comparable way of calculating the metrics is through traditional approaches, where we compare a list of relationships with the ground truth. In order to calculate the metrics in this way, the clusters need to be reduced to relationships. In our method, we use similarity scores between columns to cluster them together. This means that similar/related columns should in theory be clustered together. Hence, we can reduce the clusters into relationships by regarding them as cliques. This means that each column c in cluster i is connected to all other columns in cluster i . The reduction from clusters into relationships is shown in figure 5.1. The list of relationships can then be used to determine the metric scores in a more traditional way with the formulas shown below where A is the list or relationships reduced from the clusters and T is the list of relationships representing the ground truth:

$$Precision(A) = \frac{|A \cap T|}{|A|} \quad Recall(A) = \frac{|A \cap T|}{|T|}$$

$$F - measure(A) = \frac{2 \cdot Precision(A) \cdot Recall(A)}{Precision(A) + Recall(A)}$$

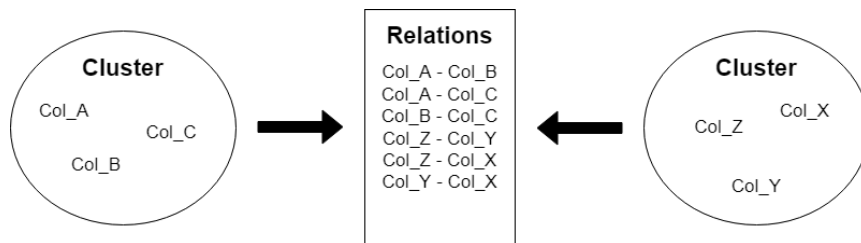


FIGURE 5.1: Reduction of clusters into a list of relationships

5.2.3 Valentine Metrics for Score Quality

As we have the similarity scores of all the matchers and the aggregated similarity scores of our methods, we can compare those in order to gain more insight in how meaningful the aggregated similarity scores are. For this sole purpose, we adopt

the same idea as described in the work of Koutras et al., 2021, where we calculate metrics at a certain number k , where k means how much of the top scoring matches we will consider for the calculation of the metrics. E.g. for a precision metric with $k = 10$, we will firstly sort all the candidate matches based on their score and then only use the top 10 scoring matches to calculate the precision metric. Using this way of calculating the metrics, we can find more about how the similarity scores are for each individual matcher and our methods. The precision, recall and f-measure are exactly calculated as the Traditional Metrics. **Note that clustering is excluded in the calculation of the Valentine metric scores. It is hard for clustering to determine which new match has more importance compared to others. Thus we only consider the scores to determine these metrics and use these scores to determine how good the aggregated similarity scores are.** The equations used to determine the metrics are shown below where A_k denotes the list of k top scoring similarity scores in the list A and T denoting the list of relationships representing the ground truth:

$$Precision(A, k) = \frac{|A_k \cap T|}{k} \quad Recall(A, k) = \frac{|A_k \cap T|}{|T|}$$

$$F - measure(A, k) = \frac{2 \cdot Precision(A, k) \cdot Recall(A, k)}{Precision(A, k) + Recall(A, k)}$$

5.3 Results

This section is dedicated to depict the results of the experiments. The results are categorized based on each dataset. Only the best f-measure score of each method is considered along with the corresponding precision and recall score since the f-measure is supposed to determine the effectiveness of the method with importance to both recall and precision. All methods, except for machine learning will be compared with each other. Machine learning does not produce the same output and therefore is hard to compare to the other proposed methods. Our machine learning method will therefore be evaluated separately. Additionally, all best scoring formed clusters and the correspondence with the ground truth are available on [as HTML pages](#), since adding 50 diagrams would result in too many pages for this report. Additionally, screenshots of the diagram result in poor quality, hence we have HTML pages which allows the viewer to interactively change the view of the diagrams. For security reasons, the ING data has been mapped to custom table (table_1, table_2, etc.) and column (A, B, C, etc.) names. How to interpret these chord diagrams is explained in appendix [E](#).

5.3.1 TPC-H

When we run the experiments on the TPC-H dataset, we see different metric scores from different methods in table [5.1](#). What is noticeable at first sight is that the metric scores are relatively low, all hovering around 0.5. When we look at the individual methods, we see that the Multiplicative Weight Update method had a relatively high recall score compared to the other methods for both metrics. On the contrary, the Normalized Linear Programming method had the maximum precision score, but a poor recall score. This indicates that the Normalized Linear Programming method assigned such weights that the related column-pairs were clustered first, but could not find all of them. Out of our methods, (Normalized) Linear Programming method had the highest metric scores for both cluster metrics and traditional metrics. And above all, all our proposed methods performed better than the MAX aggregator.

Looking at our overall metric scores, our proposed methods do not perform near the state-of-the-art solutions where f-measure scores are around 0.9 for specific domains (Zhang et al., 2011; Pei, Hong, and Bell, 2006).

To analyze the behaviour of our methodology more, we take a deeper dive into the assigned scores for each column-pair. We see that there are a lot of unrelated column-pairs with relatively high similarity scores (> 0.5) assigned by the matchers. Majority of the high similarity scores came from cupid, where the average score for unrelated column-pairs is 0.8. This is very interesting as coma, coma++ and cupid are the matchers which found evidence for most related column-pairs. Analyzing the assigned similarity scores, we found out that different matchers have different score distributions. The mismatching score distribution could potentially make it harder to combine the similarity scores as matchers with a big weight who gives lower scores in general have difficulty keeping the scores as high as possible. However, when we look at table 5.1, we see that the Normalized version of our Linear Programming method did not yield any better metric scores compared to the regular Linear Programming method.

To compare the quality of the aggregated scores, we will calculate the Valentine Metrics for every possible k and visualise it in a graph. These graphs can be found in figure 5.5 where we show the behaviour of the precision, recall and f-measure scores as we progress the value k . When we look at the precision graph, we see that the multiplicative weight update method and the normalized linear programming method are leading for all k . This means that those two methods are the best at giving higher scores to relevant column-pairs. When we take a look at the recall, we see that all our methods quickly rise to a score of 1.0 compared to the individual matchers. This means that our proposed methods found all the related column-pairs faster compared to the individual matchers. This means that the scores for unrelated column-pairs are lower compared to the scores assigned from the individual matchers. The proposed methods have all the related column-pairs in the top 200 scoring matches, whilst some individual matchers take 400+ to 1500+ matches before they found all the matches. When we take a look at the f-measure, we see that the multiplicative weight update method is clearly the winner as it stays at top for almost all k . The runner up is the Normalized Linear programming method, reaching a peak f-measure score close to the multiplicative weight update method. When we compare those metric scores to the scores mentioned in 5.1, we see that clustering can potentially help the metric scores, but also harm them as the metric scores decrease. Clustering helps us grouping potential related columns together, helping the user to search for relevant columns, but the metric scores are declining when we take a look at the traditional metrics.

Although multiplicative weight update and normalized linear programming method both got better f-measure scores when we solely look at the Valentine metrics, we see however from the metrics in table 5.1 that linear programming performs the best. This either means that adding clustering to the aggregated similarity scores either decreased the quality of the matches for some proposed methods or either increased the quality of some proposed methods. As we can not directly compare the two metrics, the traditional metrics are the most comparable to the Valentine metrics. If we were to compare it with the traditional metrics, the quality of all the proposed methods dropped the quality of the matches. However, this can not be concluded as we would then neglect that clusters help us group column-pairs and simplify the

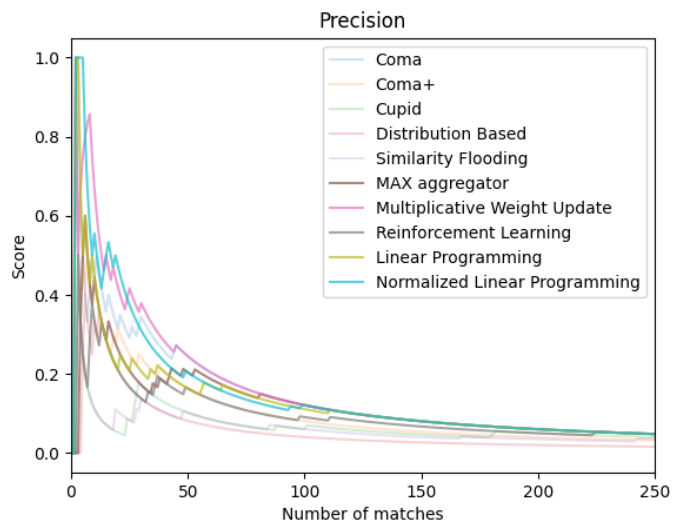
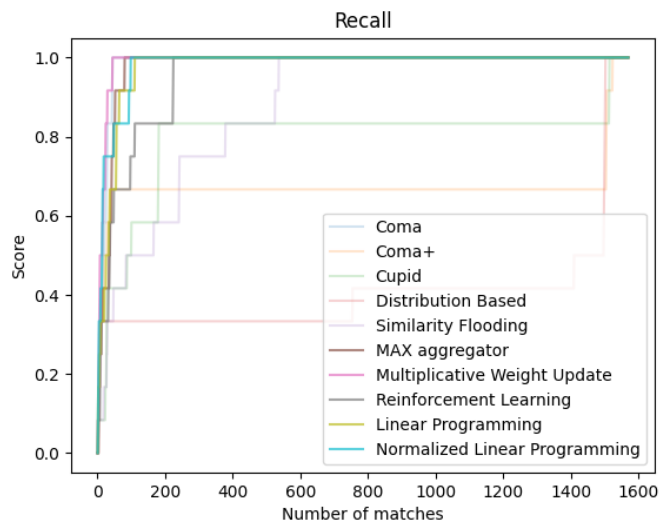
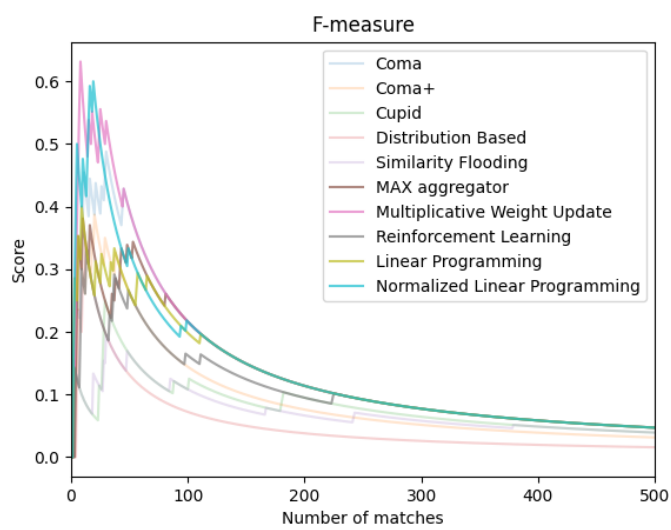
search for matches.

Summary

The matchers Coma, Coma++ and Cupid gave relatively high similarity scores for all the related column-pairs. However, Cupid assigns relatively high similarity scores even for columns with no relation. The matchers all have a different score distribution, but they do not affect the effectiveness of the method as the normalized version also yields similar results. Comparing the scores only, the aggregated score can rank the column-pairs better than each individual matcher.

| | Cluster metrics | | | Traditional metrics | | |
|--------------------------------------|-----------------|--------|-----------|---------------------|--------|-----------|
| | precision | recall | f-measure | precision | recall | f-measure |
| Max aggregator | 0.33 | 0.69 | 0.45 | 0.17 | 0.25 | 0.2 |
| Multiplicative Weight Update | 0.27 | 0.86 | 0.41 | 0.13 | 0.75 | 0.23 |
| Reinforcement Learning | 0.38 | 0.94 | 0.54 | 0.31 | 0.42 | 0.36 |
| Linear Programming | 0.56 | 0.67 | 0.61 | 0.50 | 0.42 | 0.45 |
| Normalized Linear Programming | 1.0 | 0.39 | 0.56 | 1.0 | 0.25 | 0.4 |

TABLE 5.1: Best achieved metrics from running our methods on the original TPC-H data with the optimal settings

FIGURE 5.2: Valentine precision for each k FIGURE 5.3: Valentine recall for each k FIGURE 5.4: Valentine f-measure for each k FIGURE 5.5: Valentine metrics for each k in the TPC-H dataset

5.3.2 Fabricated TPC-H

When we run the experiments on the fabricated TPC-H dataset, we see that the metric scores all decreased compared to the original TPC-H when we look at the scores in table 5.2. The total amount of relationships our methods should find are 37 relationships, where 13 relationships are forged and 24 were already implicitly present. As we can see in table 5.2, our proposed methods did not improve the cluster metric scores. The MAX aggregator, multiplicative weight update, and reinforcement learning methods all increased their maximum f-measure score compared to the original TPC-H dataset. All the proposed methods perform relatively the same when we solely look at the cluster metrics. However, when we also take the traditional metrics into consideration, we see that the (normalized) linear programming performs the best in that aspect. This indicates that linear programming clusters relevant columns faster/better compared to the other methods.

What is noticeable is that the multiplicative weight update method resulted in only 14 clusters (including singular clusters) where one big cluster is formed with little to no related columns for the best f-measure for the clustered metrics. Since the multiplicative weight update method had a cutoff threshold of 0.5, only 19 out of 37 relationships had a similarity score assigned from at least one matcher. This implies that the relationships of this dataset are harder to find for the individual matchers.

We conducted a preliminary experiment with different fabrication settings with our multiplicative weight update method to see the effects of the different settings. We only considered contributing factors which could potentially influence the similarity scores. The different contributing factors of the fabrication tool are:

- **Row overlap:** Percentage of overlapping instances
- **Row noise:** Percentage of instances to be affected by noise
- **Column noise:** Noise in the column names in the form of abbreviation/prefixing/dropping vowels

The column overlap should not be the issue as the overlap is exactly what our methods try to find. Therefore, increasing or decreasing the overlap of the columns is only making more or less relationships for the methods to find. The other factors can influence the evidence for matchers to find related column-pairs and are therefore compared to each other. The different factors are compared with each other through the multiplicative weight update method. We compare the results of the multiplicative weight update method with our fabrication settings and try to tweak every contributing factor from the same fabrication settings individually to see the effects. The results of the experiments are shown in table 5.3. Note that the column overlap for checking the different factors differs compared to our baseline experiment due to the randomization nature of the fabrication tool.

From table 5.3 we can see that increasing the row overlap to 80% immediately increases the recall and f-measure. Although the amount of found relationships (column-pairs with scores > 0.5) remained quite similar, there is a slight increase compared to the baseline. The increase in recall and f-measure can be explained as the related column-pairs have higher scores compared to the baseline, ultimately leading to being earlier discovered. Removing the row noise, we see a similar increase in recall and f-measure, resulting in similar behaviour as the increased row overlap.

When we change the column names into custom made names (mapping is shown in appendix D), we see that although the recall increased, the precision dropped, resulting similar f-measure as our baseline. What is very noticeable from the experiment with custom made column names is that all the related column-pairs had a similarity score of at least 0.5 assigned to it. The reason for the similar f-measure score is due to the fact that the amount of unrelated column-pairs with similarity scores of at least 0.5 assigned to it, is 3 times as much as our baseline. Therefore, the custom made column names introduces more confusion. Although these different factors of fabrication can influence the results, we believe that the methods are not affected that much and that the performance would not increase much more when we change all these factors at once.

When we look at the quality of the aggregated score of our proposed methods and individual matchers in figure 5.9, we see that the scores of the (normalized) linear programming method(s) are leading in all three metrics. However, it is noticeable that the multiplicative weight update method worked less good for the fabricated TPC-H dataset when it performed the best on the TPC-H dataset. What is also noticeable is that the maximum reaching f-measure scores is relatively low compared to the TPC-H dataset. We only achieve a top f-measure score of 0.45 whilst for the TPC-H dataset we had a top f-measure score of 0.6. This is due to the recall of all the methods. In TPC-H, the related column-pairs were relatively easy to find as they found all the related column-pairs within the top 200 scoring pairs. With the fabricated TPC-H, we see however that it has more difficulty finding all the relevant matches as all the methods (including the individual matchers) find all the relevant matches after 1700+ matches.

Summary

Our methodology performs relatively the same with this dataset compared to the original TPC-H dataset. The different settings of the fabrication tool used to create the dataset could improve the scores slightly. Multiplicative weight update method created one big cluster with little to no relevant column-pairs to maximize the f-score. Comparing the aggregated score with the similarity scores of the individual matchers, we see that it still performed better compared to the individual matchers. However, this dataset is supposedly harder as the max f-measure for the Valentine metrics is 0.45 as opposed to the original TPC-H dataset, which had a max of around 0.6 for the f-measure.

| | Cluster metrics | | | Traditional metrics | | |
|--------------------------------------|-----------------|--------|-----------|---------------------|--------|-----------|
| | precision | recall | f-measure | precision | recall | f-measure |
| Max aggregator | 0.47 | 0.74 | 0.57 | 0.19 | 0.17 | 0.18 |
| Multiplicative Weight Update | 0.54 | 0.62 | 0.58 | 0.25 | 0.14 | 0.18 |
| Reinforcement Learning | 0.46 | 0.66 | 0.54 | 0.70 | 0.20 | 0.31 |
| Linear Programming | 0.45 | 0.83 | 0.58 | 0.24 | 0.74 | 0.36 |
| Normalized Linear Programming | 0.43 | 0.71 | 0.54 | 0.38 | 0.34 | 0.36 |

TABLE 5.2: Best achieved metrics from running our methods on the fabricated TPC-H data with the optimal settings

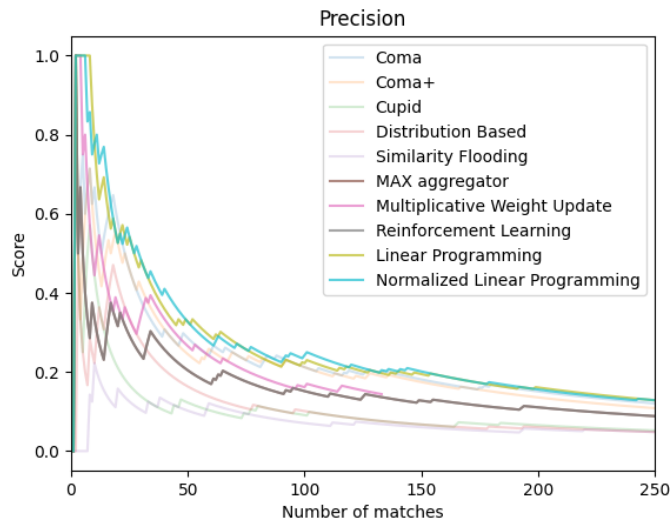


FIGURE 5.6: Valentine precision for each k

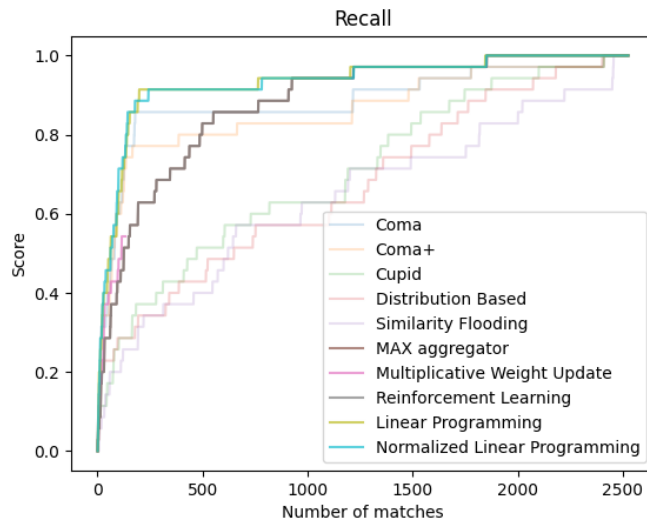


FIGURE 5.7: Valentine recall for each k

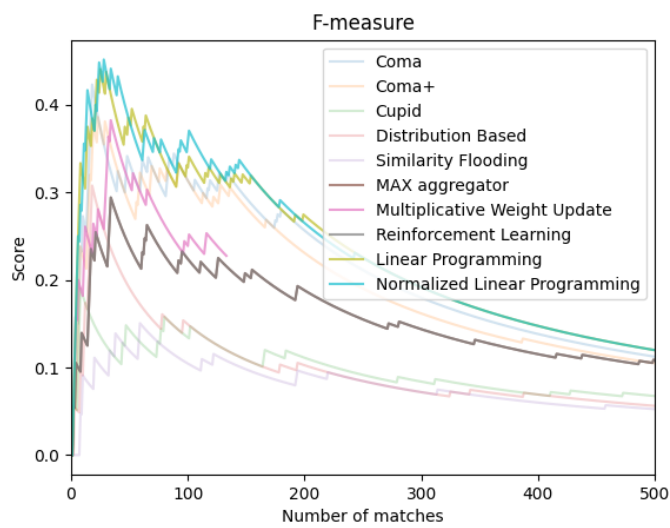


FIGURE 5.8: Valentine f-measure for each k

FIGURE 5.9: Valentine metrics for each k in the fabricated TPC-H dataset

| Multiplicative Weight Update | Cluster metrics | | |
|--|-----------------|--------|-----------|
| | precision | recall | f-measure |
| Baseline | 0.55 | 0.63 | 0.58 |
| Row overlap (50% →80%) | 0.54 | 0.82 | 0.65 |
| Row noise (30% →0%) | 0.53 | 0.78 | 0.63 |
| Column noise (custom made column names) | 0.46 | 0.79 | 0.58 |

TABLE 5.3: Best achieved metrics from changing different fabrication factors for the fabricated TPC-H data with the optimal settings

5.3.3 IMDb

There are a few things to take into consideration when evaluating our methods with the IMDb dataset. The trimmed IMDb dataset contains two main clusters (attributes) which we try to create: unique IDs for titles (tconst) and unique IDs for persons (nconst). As quite a lot of column names are similar (tconst and nconst), what is more interesting are the columns which have different column names, e.g. titleId and tconst. What is also interesting is that there exists a parentTconst column in the title_episode schema, which was unable to match with the tconst in the same schema, because they are already related. In fact, they refer to the same entity type, but are unable to be clustered together as fast as wished due to the distance being set to max for columns in the same table. When inspecting the similarity scores, we can see that parentTconst has quite high similarity scores with other tconst columns, but was unable to be clustered together with the tconst columns due to the distance between the parentTconst and tconst from the same schema. Changing the max distance yields in similar behaviour, which therefore is a drawback of our method with clustering. Additionally, a lot of foreign key references in the rows are most likely removed due to the trimming, which should also be taken into consideration. As the IMDb data tables keeps growing over time (currently at max 50 million rows per table), it is infeasible for our available computer to run the method over all the data.

When we run the experiments on the IMDb data, we get the results shown in table 5.4. As can be seen from the table, the (normalized) linear programming method(s) increases the cluster metric scores compared to the other methods. Compared to the MAX aggregator, the other methods perform better when we solely look at the cluster metrics. When we only look at the traditional metrics, the MAX aggregator performs similar compared to all the other methods except for the normalized linear programming method. What is interesting from the results is that the normalized linear programming performs the best when we look at the cluster metrics, but the worst when we look at the traditional metrics.

To check whether the results were tainted due to trimming, we checked by using the max size our computer was able to handle (500.000 rows per schema). Hence, we have conducted experiments for 100.000 and 500.000 rows per schema. The scores and clusters remained exactly the same for both sizes. Therefore, the amount of rows in this case did not have any effect on the method, probably due to the proportion that is trimmed compared to the original. It is therefore impossible to say that the given experiments showed that the matchers do not work properly in the given methods, however, it is shown that the method is limited to 500.000 rows per table due to the available tools and RAM. When we inspect the rows that we have, we see in the title_crew table have references for writers and directors with values

as nm0721526, whilst we only have columns up to nm0500000. Additionally, some tables contain duplicate titleId titles whilst other tables contain only unique titleIds, which might be a possible explanation why the distribution based matcher is having more trouble finding the relationships.

When we take a look at the score quality in figure 5.13, we see again that the top scoring f-measure is the multiplicative weight update method and the linear programming method. Again, the top f-measure is still relatively low compared to the state-of-the-art methods. Just as with the fabricated TPC-H dataset, the relevant column-pairs are rather hard to find as we can see in the recall graph. As for precision, almost all the methods are quite similar. This implies that the dataset is harder for the matchers to find the relevant column-pairs.

Summary

Our methodology was unable to run the IMDb dataset due to the amount of instances. Trimming the dataset most likely resulted in less evidence, as referencing columns instances could be removed due to trimming. The weights obtained by linear programming gave the best performance in all metrics.

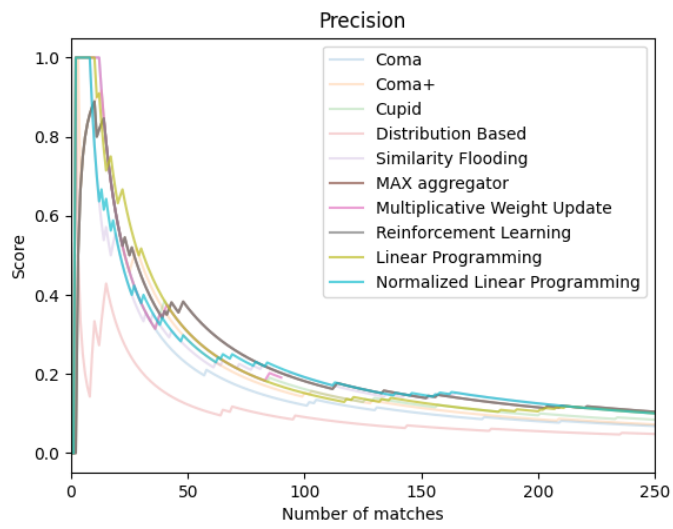
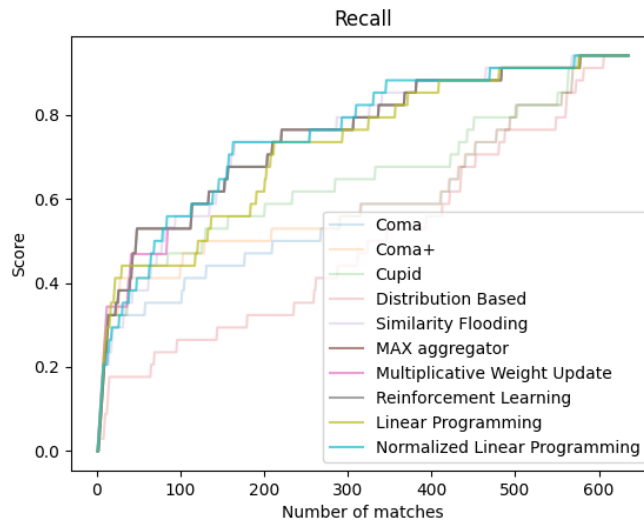
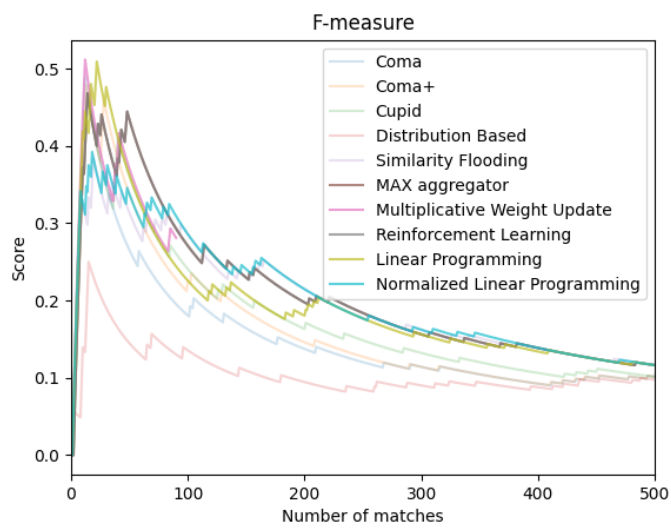
| | Cluster metrics | | | Traditional metrics | | |
|--------------------------------------|-----------------|--------|-----------|---------------------|--------|-----------|
| | precision | recall | f-measure | precision | recall | f-measure |
| Max aggregator | 0.41 | 0.63 | 0.49 | 0.44 | 0.47 | 0.46 |
| Multiplicative Weight Update | 0.41 | 0.63 | 0.49 | 0.46 | 0.47 | 0.46 |
| Reinforcement Learning | 0.41 | 0.63 | 0.49 | 0.44 | 0.47 | 0.46 |
| Linear Programming | 0.75 | 0.56 | 0.64 | 0.73 | 0.32 | 0.45 |
| Normalized Linear Programming | 0.75 | 0.56 | 0.64 | 0.83 | 0.15 | 0.25 |

TABLE 5.4: Best achieved metrics from running our methods on the original IMDb data with the optimal settings

5.3.4 Fabricated IMDb

When we run the experiments on the fabricated IMDb dataset, we see that the metric scores are actually higher compared to the original IMDb dataset (table 5.5). However, when we take a look at the traditional metrics, our methods perform even worse compared to the MAX aggregator. What is more interesting to see is that the normalized linear programming method is better when we use the Cluster metrics compared to the linear programming method, but worse when we use the traditional metrics. Overall the findings are similar compared to the other datasets. All the methods do not perform up to par with the state-of-the-art.

What is interesting to see when we compare the aggregated score quality, is that Coma and Coma++ are actually leading as we can see in the f-measure in figure 5.17. The lead does not come from recall, but from precision. When we take a look at the precision graph, we see that for the top 100 scoring matches, coma and coma++ are better compared to every other method/matcher. It is rather interesting because the weights of Coma and Coma++ were 0 for the (normalized) linear programming method. For the multiplicative weight update method, Coma was included, but it had a hard time finding the relevant matches because a large part was already pruned due to the cutoff threshold of 0.5. Which means that for this dataset, normalizing the scores for the multiplicative weight update method could potentially have

FIGURE 5.10: Valentine precision for each k FIGURE 5.11: Valentine recall for each k FIGURE 5.12: Valentine f-measure for each k FIGURE 5.13: Valentine metrics for each k in the IMDb dataset

high metric scores. However, as we can see from the other datasets, the results are not robust and vary dependent on the dataset.

Summary

Our methodology scored higher cluster metrics scores compared to the original IMDb dataset, but worse traditional metric scores. Our approaches all performed similar or worse compared the the MAX aggregator when we solely look at the traditional metrics. Looking at the aggregated scores and the Valentine metrics, Coma and Coma++ performed significantly better due to better precision.

| | Cluster metrics | | | Traditional metrics | | |
|--------------------------------------|-----------------|--------|-----------|---------------------|--------|-----------|
| | precision | recall | f-measure | precision | recall | f-measure |
| Max aggregator | 0.56 | 0.82 | 0.67 | 0.42 | 0.29 | 0.34 |
| Multiplicative Weight Update | 0.59 | 0.79 | 0.68 | 0.26 | 0.26 | 0.26 |
| Reinforcement Learning | 0.56 | 0.82 | 0.67 | 0.42 | 0.29 | 0.34 |
| Linear Programming | 0.65 | 0.75 | 0.70 | 0.48 | 0.23 | 0.32 |
| Normalized Linear Programming | 0.70 | 0.90 | 0.78 | 0.31 | 0.22 | 0.26 |

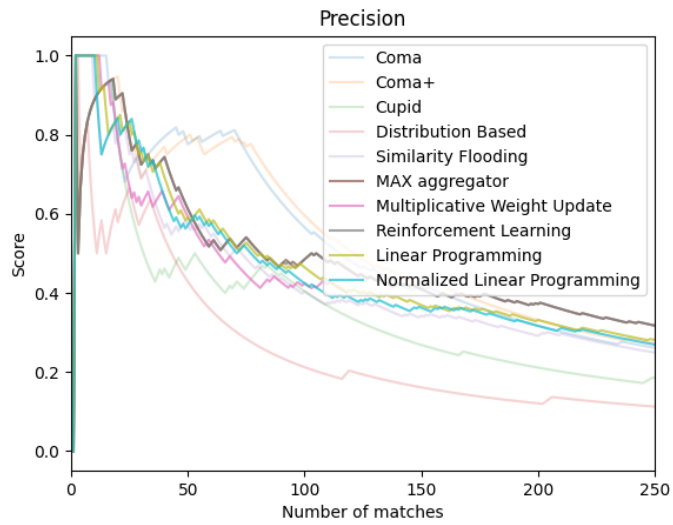
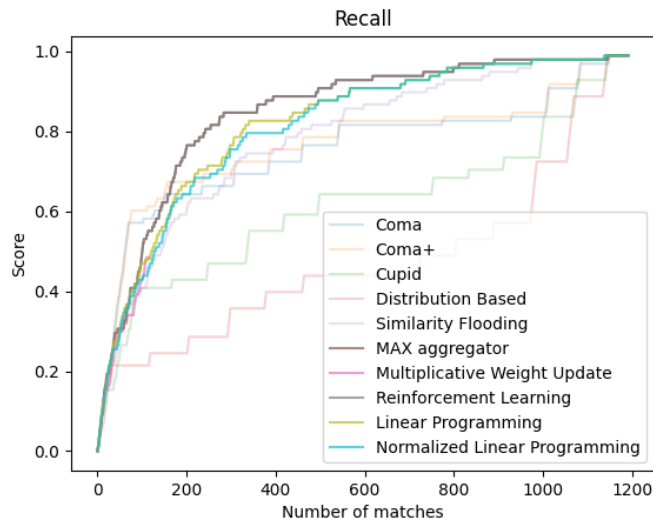
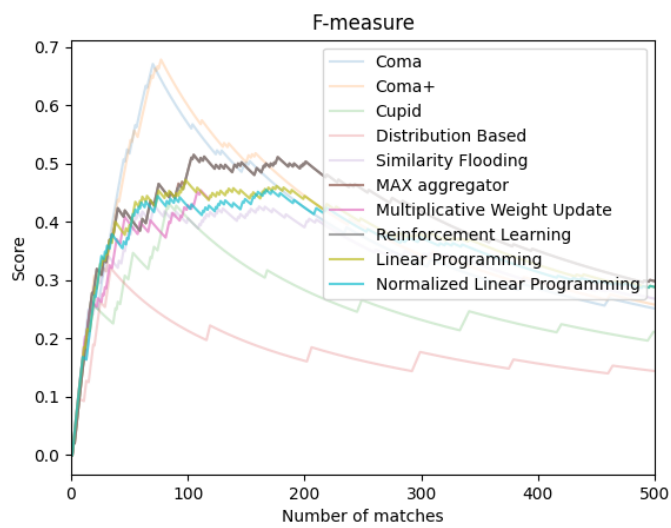
TABLE 5.5: Best achieved metrics from running our methods on the fabricated IMDb data with the optimal settings

5.3.5 ING

Evaluating with the ING dataset allows us to see how well our approach holds in real-life situations as it is sampled from an actual used application. When running the experiments on the ING dataset, we noticed that the scores were relatively bad compared to the other datasets. What is even more noticeable is that the multiplicative weight update approach scores very low on the traditional metrics. Partially is due to the deducted weights, which practically gave all the matchers a weight very close to 0, whilst the distribution-based matcher has a significantly higher weight than the rest. The reason for this is that the distribution based often gives scores 0, which makes it easier for the matcher to guess the incorrect column-pairs as opposed to the minority correct column-pairs. Similarly, the reinforcement learning approach also give the distribution-based matcher significantly higher weights, however, the rest of the matchers did not have a weight close to 0 as opposed to the multiplicative weight update method. From the results in table 5.6, we see that the linear programming method outperforms the rest of the methods.

Inspecting the scores and behaviour, we found that majority of the columns which got clustered together first were columns with similar to identical names, which are not explicitly stated as key-pairs. For example, all primary keys are given the name 'PID' or all tables have a 'comments' field, which in fact have little to no correlation with each other. This is where the distribution-based matcher outperforms the rest, resulting in getting higher weights compared to the others.

From the Valentine metrics in figure 5.21, we can see that the MAX aggregator performs the best amongst the other methods. When diving into the scores, we see that for the majority of the column-pairs, cupid assigns a score of 1 because the names are identical. For some of the column-pairs, the distribution based gives a high score, whereas cupid gives lower scores. There were also some column-pairs which are

FIGURE 5.14: Valentine precision for each k FIGURE 5.15: Valentine recall for each k FIGURE 5.16: Valentine f-measure for each k FIGURE 5.17: Valentine metrics for each k in the fabricated IMDB dataset

related had very poor similarity scores where all the matchers gave scores close to 0. This is due to the fact that the tables did not have any instance data, and the present deviating column names (e.g. MovieID and ID). Comparing the recall however, we see that the MAX aggregator finds all the relevant column-pairs later than the other methods, whilst the precision starts off quite high. this can be explained as the majority of the related column-pairs have the exact same column name, resulting in cupid similarity scores of 1.

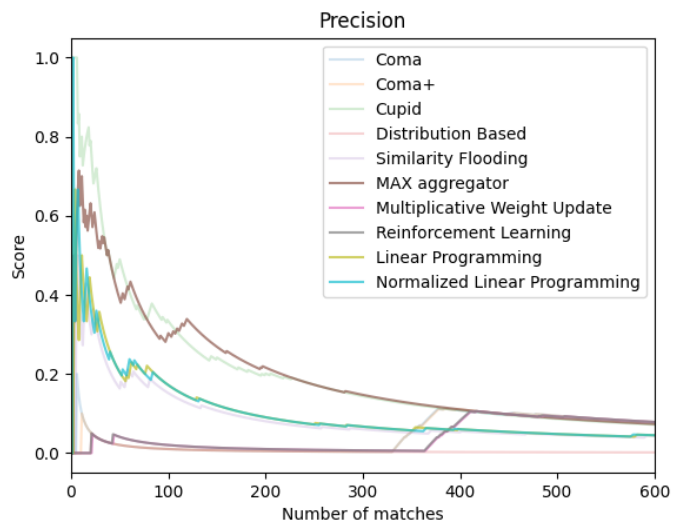
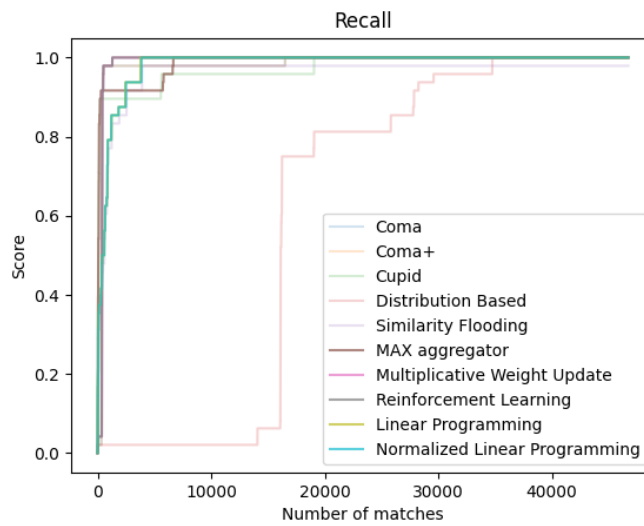
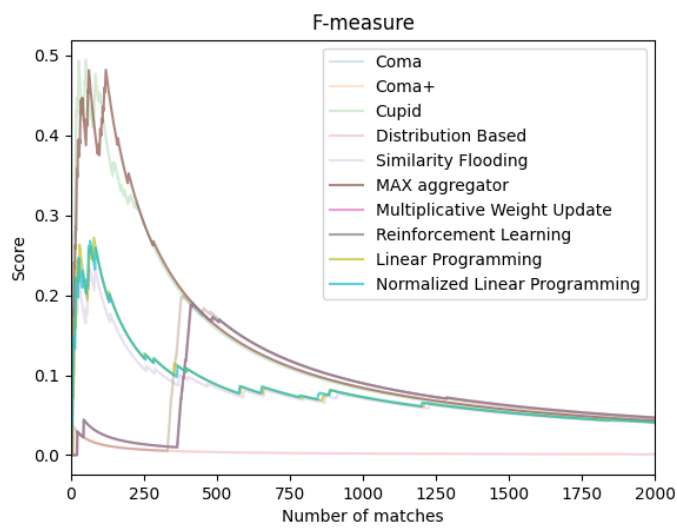
As the used ING data was a small portion of the entire application, it can be expected that our methodology works even worse when more 'noise' tables are added. The fraction that we have used for our evaluation still contained some 'noise' tables, but not to the same extent as the actual database, making it easier for our methodology. Additionally, the run-time for forming the clusters and evaluating the clusters increased exponentially as the run-time scales exponentially with the amount of columns and tables, making evaluating all the tables take forever.

Summary

Our methodology scored the worst for this dataset compared to the other ones. The ING dataset is supposed to depict the real-life scenario, where we use in-use application data to determine the effectiveness of our methodology. Nevertheless, we see that even if we use a selected fraction of the data, our methodology scored rather disappointing. The data contained a lot of identical names, which resulted in those being clustered together first (e.g. abbreviations). The Valentine metrics shows that the MAX aggregator creates the best scores, which is explainable as the majority of related columns have identical names.

| | Cluster metrics | | | Traditional metrics | | |
|--------------------------------------|-----------------|--------|-----------|---------------------|--------|-----------|
| | precision | recall | f-measure | precision | recall | f-measure |
| Max aggregator | 0.20 | 0.46 | 0.28 | 0.17 | 0.69 | 0.27 |
| Multiplicative Weight Update | 0.14 | 0.48 | 0.21 | 0.2 | 0.02 | 0.04 |
| Reinforcement Learning | 0.14 | 0.74 | 0.23 | 0.13 | 0.92 | 0.23 |
| Linear Programming | 0.28 | 0.89 | 0.43 | 0.26 | 0.40 | 0.32 |
| Normalized Linear Programming | 0.30 | 0.60 | 0.40 | 0.32 | 0.13 | 0.18 |

TABLE 5.6: Best achieved metrics from running our methods on the ING data with the optimal settings

FIGURE 5.18: Valentine precision for each k FIGURE 5.19: Valentine recall for each k FIGURE 5.20: Valentine f-measure for each k FIGURE 5.21: Valentine metrics for each k in the ING dataset

5.3.6 Discussion

For data discovery tools, an f-measure of 0.6-0.7 is relatively low. In current existing papers, the f-measure score revolves around 0.9 or higher (Zhang et al., 2011; Pei, Hong, and Bell, 2006). In all aspects (scores, clusters, different metrics), the eventual metric scores were relatively disappointing compared to the state-of-the-art methods. Inclusion of different matchers can indeed improve the overall score quality with the weighted aggregation approach. However, it requires relatively good weights, which is hard to obtain due to the fact that related column-pairs should be maximized, whilst unrelated column-pairs should be minimized. As there are numerous options, determining the optimal weights is a very hard task to do. Normalizing the different score distributions of the individual matchers did not yield robust positive results.

Regardless of the weights, we see from the results that it still outperforms individual matchers when using one of our methods. The weighted aggregation is more stable compared to the MAX aggregator, as we see that for the majority the weighted aggregation stayed ahead of the MAX aggregator. In almost all cases, the MAX aggregator performed worse compared to any of the methods we had tried with far from optimal weights. Which shows us that the weighted aggregation is the better approach compared to the MAX aggregator. However, from our experiment with the ING data, we saw that the MAX aggregator actually performed better, due to the circumstances of having identical naming for IDs.

By all those attempts for finding the best weights, we can actually see that the metrics does not go up by much. The best metric scores we got for the f-measure was always in the range of 0.5 and 0.7. Plainly combining the weights is probably not the way to go as the relations are most likely more complex than a linear combination of the weights. This leads to an indication of using non-linear machine learning to combine different matchers to find related column-pairs.

5.4 Machine Learning Results

As indicated from the experiments, the relation between the similarity scores of the matchers are probably more complex than a linear combination. Therefore, we implemented a Support Vector Machine model (described in chapter 3.2.4), which tries to explore the possibilities within the machine learning world for combining similarity scores of different matchers.

Data

Machine learning models need labelled data in order to learn and test the models. We therefore use all the labelled matches from the preliminary experiments to train and test our general model. The used labelled data and its distribution is shown in table 5.7. The ratio indicates a typical imbalanced classification problem, where we have a minority class and a majority class (Nguyen, Cooper, and Kamei, 2009).

The total amount of data will be split into a train set and a test set. The split will be 80% for the train set and 20% for the test set. The split will keep the ratio of 1:32 for the related and unrelated column-pairs, meaning that both the train and test set will have an approximate ratio of 1:32 for the related and unrelated column pairs.

| Related Column-pairs | Unrelated Column-pairs | Total Labelled Data Points |
|----------------------|------------------------|----------------------------|
| 176 | 5646 | 5992 |

TABLE 5.7: Amount of used labelled data and its distribution amongst the related and unrelated column-pairs

5.4.1 Evaluating the model

Plainly training the model with the train set and testing the accuracy with the test set results in quite high accuracy of 98%. To ensure that the accuracy is not a result of a lucky partition, we conduct the same experiment 100 times and denote the mean and standard deviation of the accuracy in table 5.8. The mean accuracy of 0.977 is very high and seems like a very good working model.

Since our data is quite imbalanced as shown in table 5.7, we have to keep in mind that the test set only contains 35 related column-pairs and 1150 unrelated column-pairs. With the original dataset being 5922 data points where only 176 (3%) are related column-pairs, the high accuracy can in fact be explained. If the majority of the unrelated column-pairs are correctly predicted, the accuracy will naturally be very high. The model can simply label all the column-pairs as unrelated and will still achieve a very high accuracy.

Hence, we will look at the confusion matrix (shown in figure 5.22) to see how the model behaves. As we can see, the true negatives is quite high, meaning that the unrelated column-pairs are being classified quite well for the test set. However, we can see that the related column-pairs are harder for the model to predict as only 12 out of 35 (34.3%) related column-pairs were correctly classified as related column-pairs. Although the unrelated column-pairs are for the majority correctly classified, the accuracy is quite deceiving due to the skewed dataset. If we would have calculated the metrics using the classified matches (all the datasets that we had used), we would have achieved the metric scores given in table 5.9.

| Mean | Standard Deviation |
|-------------------|-----------------------|
| 0.976835443037974 | 0.0026238484069268796 |

TABLE 5.8: The mean and standard deviation of the accuracy when we execute the learning and test process 100 times

| precision | recall | f-measure |
|-----------|--------|-----------|
| 0.86 | 0.34 | 0.49 |

TABLE 5.9: Achieved metrics of the model on the test set

As the labelled data is quite unbalanced, we are dealing with an imbalanced classification problem. As generating more related column-pairs requires us to label a huge amount of datasets, it is quite unfeasible to acquire a good amount of related column-pairs by labelling alone. Therefore, with the given data that we have, we should either use oversampling or undersampling to make the data more balanced (Sun, Wong, and Kamel, 2009). As undersampling requires a large amount of the minority class (related column-pairs), it is not optimal with our given dataset as we only have 176 instances in the minority class. Oversampling, on the other hand, is

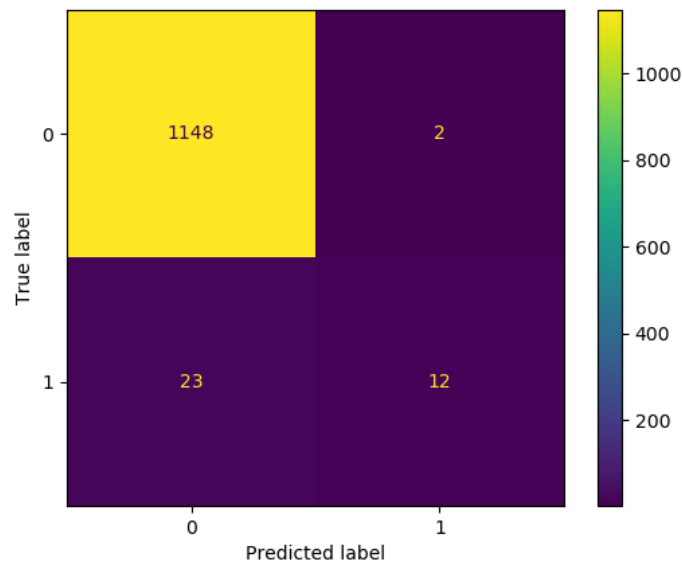


FIGURE 5.22: Confusion matrix for one execution on the test set

very prone to overfitting as the data gets copied until the data is balanced, especially since our ratio is 5746:176 (close to 33:1). This would mean that we have to copy the true match class 33 times in order to make the data balanced. This would very much enforce overfitting behaviour and is thus undesired. An oversampling approach which could potentially help is the Synthetic Minority Oversampling Technique (SMOTE), which samples the data by randomly picking a point of the minority class and compute the k -nearest neighbours for the point (Chawla et al., 2002). This way we get enough data points, without having too much replica's. The down side of using SMOTE is that the majority of the data points in the minority class is generated using the already known data points. This results in a lot of artificial data points which could potentially lose merit of the actual related column-pairs.

Oversampling should only be done on the training set, rather than the entire dataset. When we oversample the entire dataset, we bleed the information of the future test set into the model. Hence, we should only oversample the training set in order to have sufficient amount of training data. The test data still consists of the actual data points and can thus be validly used to determine the performance of the trained model, although the ratio between false and true matches is still 33:1. Doing this 100 times results in a mean accuracy and standard deviation shown in table 5.10. When we also look at the confusion matrix in figure 5.23, we see that amount of related column-pairs are being classified properly increased compared to the confusion matrix in figure 5.22. Along with the increased accuracy for the related column-pairs, the amount of false positives also increased. As the data used to train the model has approximately 3% actual data and 97% oversampled data, it is quite safe to say that the inaccuracy is likely to be influenced by the amount of actual related column-pairs. However, even with oversampling, the model was able to increase the amount of true positives, which tells us that there lies potential within the machine learning world. If we were to calculate the metrics based on the confusion matrix, we would get the results showed in table 5.11. Although the f-measure score is lower than

table than the experiment without oversampling, the recall score went up significantly. The impreciseness can most likely be influenced by the approximated data points and therefore shows potential when an appropriate amount of labelled data is available.

| Mean | Standard Deviation |
|--------------------|----------------------|
| 0.9197552742616035 | 0.008004961163524752 |

TABLE 5.10: The mean and standard deviation of the accuracy when we execute the learning and test process 100 times with oversampling

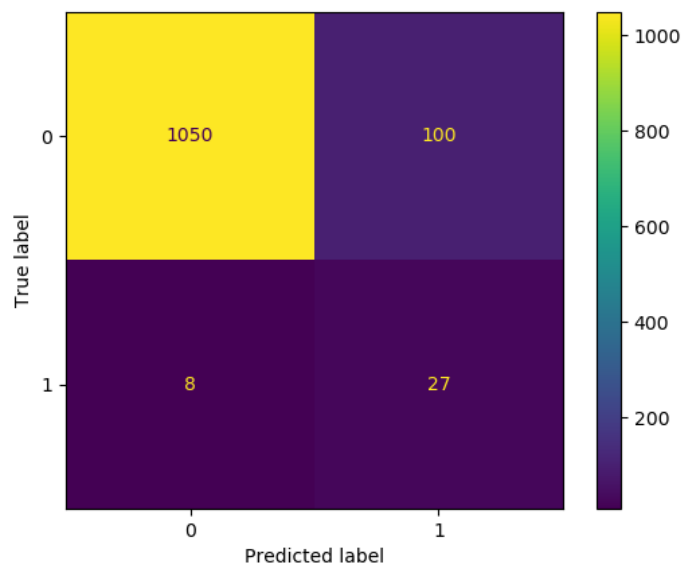


FIGURE 5.23: Confusion matrix for one execution on the test set

| | precision | recall | f-measure |
|-----------------------------|-----------|--------|-----------|
| Without oversampling | 0.86 | 0.34 | 0.49 |
| With oversampling | 0.22 | 0.77 | 0.33 |

TABLE 5.11: Achieved metrics of the model on the test set with oversampling

5.4.2 Discussion

Our small experiment with Support Vector Machine shows that there lies potential in learning whether candidate matches are actual related column-pairs by using the similarity scores of matchers as features. However, in our experiment, it showed that quite a generous amount of unrelated column-pairs is still being recognized as related column-pairs. This could be a result due to the oversampling method SMOTE of the related column-pairs as 97% of the data for the related column-pairs is forged. With enough data available from different datasets, we believe that it is worthy to take a look at machine learning for combining similarity scores of different matchers to find matches. An additional step would be to add statistics of the candidate

matches, such as data type or data distribution. This could give more insight on how to look at the data. However, this would probably require even more data to make the model more accurate since we add more features.

Chapter 6

Conclusion

We have looked into whether combining different similarity scores of different matchers with the weighted aggregation approach could potentially help us find better matches. We have created different methods, each targeting different factors of the approach and tried to learn the weights in different approaches. The aggregated scores show an improvement compared to the individual matchers, but the scores are far from the state-of-the-art solutions. Compared to the MAX aggregator, the weighted aggregation showed more positive results. We attempted clustering to find related columns for columns which have no direct similarity but a common similar column. The clusters are rather hard to compare with the individual matchers as the way of calculating the metrics are not similar. However, we see that the clusters did more harm than good as the different metrics showed more disappointing results compared to the score quality. Plainly using weights to combine similarity scores of matchers is not good enough to compete with the state-of-the-art as the model is most likely not linear. We showed with a small experiment with Support Vector Machine, that machine learning has potential as long as there is enough labelled data available and potential for more accuracy.

6.1 Limitations

The foremost limiting factor is the computer on which the experiments are ran on. The amount of data which needed to be used for the matchers were taking hours and could even lead to termination of the script due to memory shortage. The IMDb data contains millions of rows, but we were only able to use 500.000 rows per table due to the limitation of the computer. This could have influenced the scores given by the instance-based matchers and therefore also influence the methods.

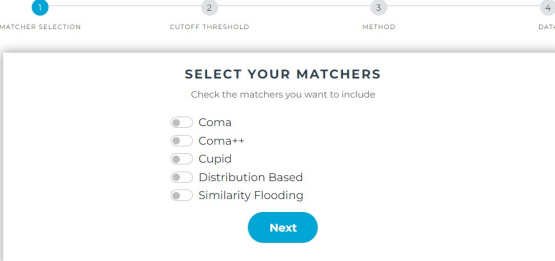
Another limiting factor is the lack of research regarding metrics for clusters. As the metrics of the clusters are calculated differently, it makes it harder to compare the results with the individual matchers. We can plainly compare the metric scores, but they both mean something different. Hence, an attempt was made to translate the clusters into relations and calculate the metrics similar to how we calculate the metrics for the individual matchers, but the results were rather disappointing. The translation of the clusters into relations is a limiting factor of the metric scores, still making it hard to compare it to the individual matchers.

The machine learning approach showed potential for usage in the data discovery world, but was limiting due to the lack of labelled data. The imbalanced problem resulted in oversampling, making the data balanced, but very artificial. The amount of the actual data about the related column-pairs is 3% of the available data where 97% is fabricated. It is believed that the inaccuracy of the model is caused due to the

extreme amount of fabricated data points, but it is hard to show due to the lack of data.

Appendix A

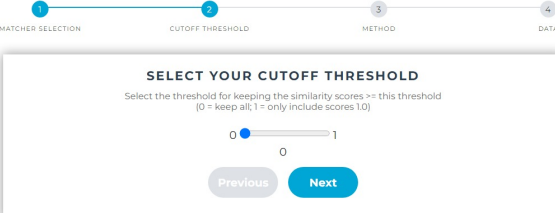
User Interface: 4-step form for configuring our methodology



A horizontal progress bar at the top shows four steps: 1. MATCHER SELECTION (highlighted in blue), 2. CUTOFF THRESHOLD, 3. METHOD, and 4. DATA. Below the progress bar is a white card with the title "SELECT YOUR MATCHERS" and the instruction "Check the matchers you want to include". There are five radio button options: Coma, Coma++, Cupid, Distribution Based, and Similarity Flooding. A blue "Next" button is positioned at the bottom right of the card.

© 2022 Copyright: Delft University of Technology

FIGURE A.1: Step 1: Select the matchers the user want to include



A horizontal progress bar at the top shows four steps: 1. MATCHER SELECTION, 2. CUTOFF THRESHOLD (highlighted in blue), 3. METHOD, and 4. DATA. Below the progress bar is a white card with the title "SELECT YOUR CUTOFF THRESHOLD" and the instruction "Select the threshold for keeping the similarity scores >= this threshold (0 = keep all; 1 = only include scores 1.0)". A slider control is shown with a blue dot at 0 and a range from 0 to 1. Below the slider are two buttons: "Previous" (disabled) and "Next" (active).

© 2022 Copyright: Delft University of Technology

FIGURE A.2: Step 2: Select the cutoff threshold (described in chapter 3.2.1)

1 MATCHER SELECTION 2 CUTOFF THRESHOLD 3 METHOD 4 DATA

SELECT YOUR METHOD

Select the method you want to learn the weights for

- Multiplicative Weight Update
- Reinforcement Learning
- Linear Programming
- Machine Learning

Previous Next

© 2022 Copyright: Delft University of Technology

FIGURE A.3: Step 3: Select one of our provided methods

1 MATCHER SELECTION 2 CUTOFF THRESHOLD 3 METHOD 4 DATA

SELECT YOUR DATA SPACE

Select the data you want to use

- IMDB
- IMDB-forged
- IMDB-trimmed
- TPC-H
- TPC-H-forged

Previous Submit

© 2022 Copyright: Delft University of Technology

FIGURE A.4: Step 4: Select the data you want to use

Label edges

Settings

Selected matchers
coma_schema
similarity_flooding

Cutoff threshold
0.09

Method
Multiplicative Weight Update Method

Data space
TPC-H

Close Save changes

orders.csv/o_shippriority
orders.csv/o_comment
part.csv/p_name
part.csv/p_mfgr
part.csv/p_brand
part.csv/p_type
part.csv/p_size
part.csv/p_container
part.csv/p_retailprice
part.csv/p_comment
partsupp.csv/ps_avaliqty
partsupp.csv/ps_supplycost
partsupp.csv/ps_comment
region.csv/r_name
region.csv/r_comment
supplier.csv/s_name
supplier.csv/s_address
supplier.csv/s_phone
supplier.csv/s_acctbal
supplier.csv/s_comment

Create cluster

FIGURE A.5: User interface to change flexible setting values

Appendix B

User Interface: Labelling ground truth clusters

Construct ground truth clusters

Select all correlated columns and click on the 'Create cluster' button to submit the cluster
Scroll down to see the submitted clusters

Table columns

- customer.csv/c_acctbal
- customer.csv/c_address
- customer.csv/c_comment
- customer.csv/c_mktsegment
- customer.csv/c_name
- customer.csv/c_phone
- lineitem.csv/l_linenum
- lineitem.csv/l_quantity
- lineitem.csv/l_extendedprice
- lineitem.csv/l_discount
- lineitem.csv/l_tax
- lineitem.csv/l_returnflag
- lineitem.csv/l_linestatus
- lineitem.csv/l_shipdate
- lineitem.csv/l_commitdate
- lineitem.csv/l_receiptdate
- lineitem.csv/l_shipinstruct
- lineitem.csv/l_shipmode
- lineitem.csv/l_comment
- nation.csv/n_name

Create cluster

FIGURE B.1: User interface allowing users to select columns and form a cluster

Submitted ground truth clusters

The submitted clusters are shown below

| Cluster 1 | Cluster 2 |
|--|---|
| customer.csv/c_nationkey nation.csv/n_nationkey supplier.csv/s_nationkey | nation.csv/n_regionkey region.csv/r_regionkey |
| Cluster 3 | Cluster 4 |
| customer.csv/c_custkey orders.csv/o_custkey | lineitem.csv/l_orderkey orders.csv/o_orderkey |
| Cluster 5 | Cluster 6 |
| lineitem.csv/l_partkey part.csv/p_partkey partsupp.csv/ps_partkey | lineitem.csv/l_suppkey partsupp.csv/ps_suppkey supplier.csv/s_suppkey |

Submit truth

FIGURE B.2: User interface showing the created clusters and button for labelling all the relationships based on the created clusters

Appendix C

User Interface: Evaluation tool

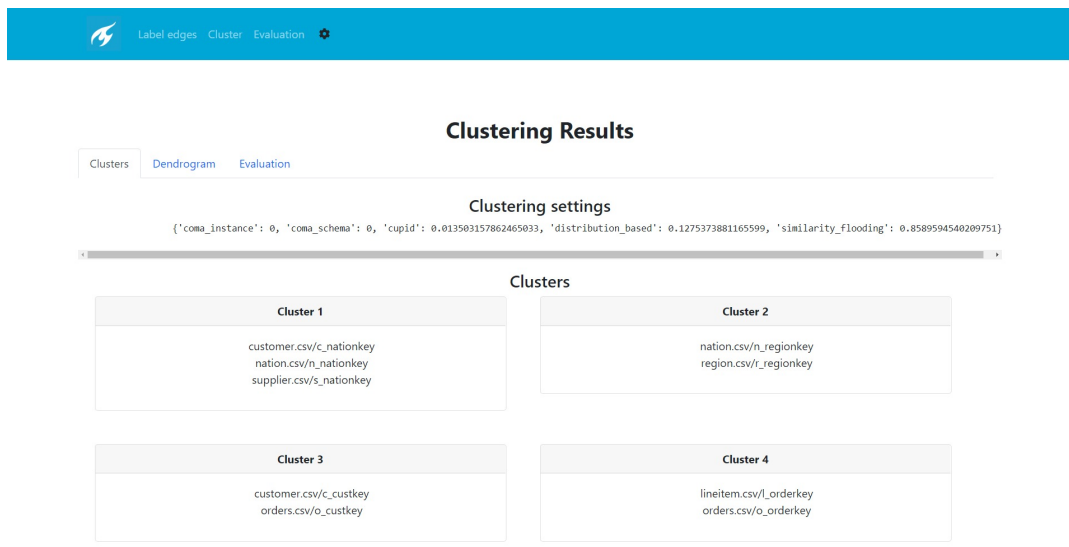


FIGURE C.1: Formed clusters are shown to the user along with the used settings

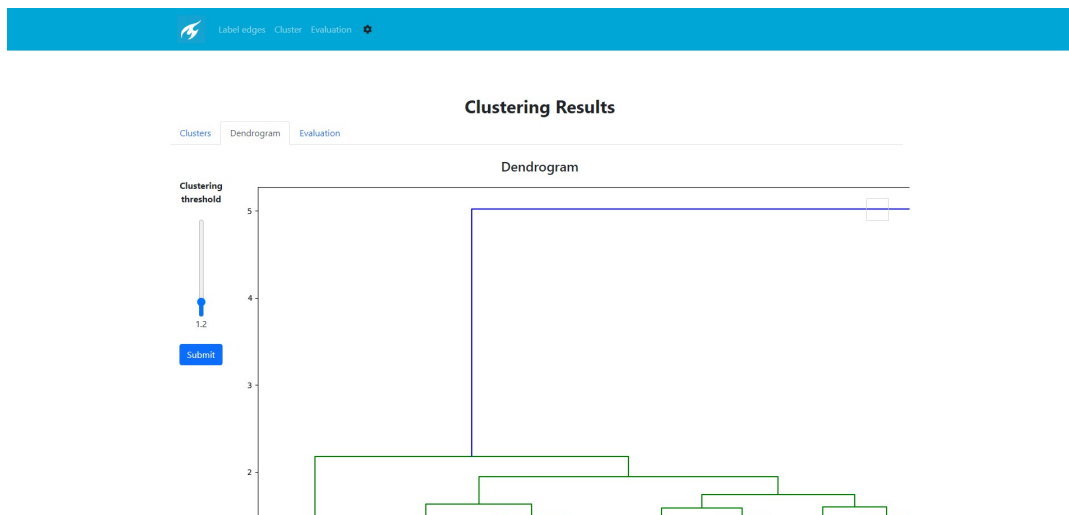


FIGURE C.2: Zoomed-in and interactive dendrogram showing the hierarchy of the columns

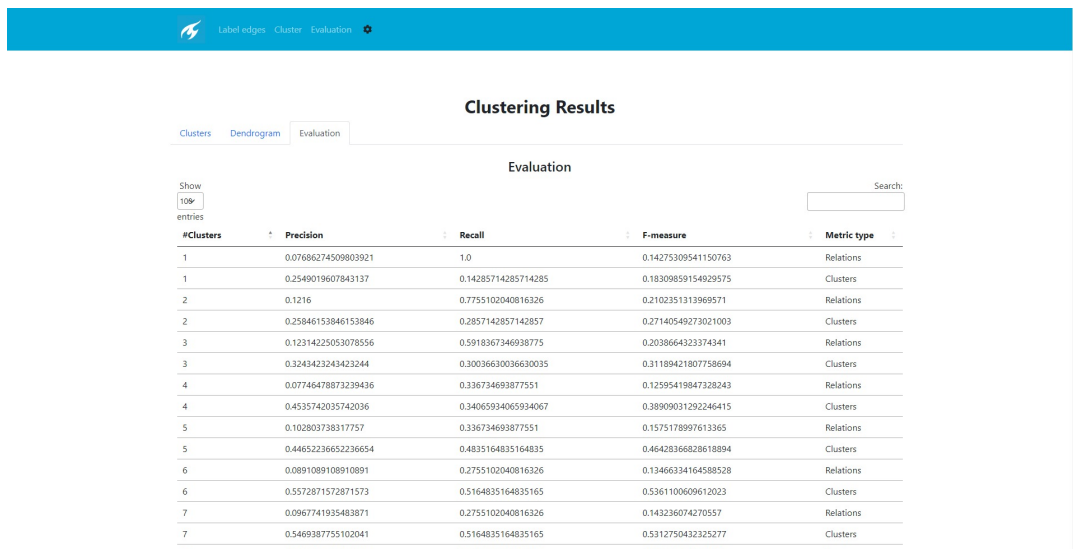


FIGURE C.3: Evaluation page with amount of clusters and the corresponding metric scores

Appendix D

Manual noise mapping for fabricated TPC-H column names

| Column name mapping | | |
|---------------------|--|--|
| Table name | Original column name | New column name |
| customer_2 | CNA CPH CACCT CMKT CCOM | customer_name customer_phone customer_account_balance customer_market_segment customer_comment |
| lineitem_2 | LORDE LQUA LSHIP LRETUR LLIN LCOM LREC LSHI LSH l_comment | lineitem_order_key lineitem_quantity lineitem_shipment_date lineitem_return lineitem_line_status lineitem_commitment_date lineitem_receipt_date lineitem_shipment_instruction lineitem_shipment_mode lineitem_comment |
| nation_2 | NNATIO NREGI NCO | nation_key nation_region_key nation_comment |
| orders_2 | OCL OCUS OORDER OORD OSHIP OCOMM | order_clerk_number order_customer_key order_priority order_placement_date order_shipping_priority order_comment |
| part_2 | PPAR PSI PCON PRETA PCOM | part_number part_size part_container part_retail_price part_comment |
| partsupp_2 | PSUPP PSUPPL PCOMM | partsupp_supply_key partsupp_supply_cost partsupp_comment |
| region_2 | RREGI RCOMM | region_key region_comment |
| supplier_2 | SNA SSUP SPHO SACCT SCOM | supplier_name supplier_key supplier_phone supplier_account_balance supplier_comment |

TABLE D.1: The mapping between the original fabricated data of TPC-H and the altered data for validation of identified problem

Appendix E

Results: Example of formed clusters compared to the ground truth

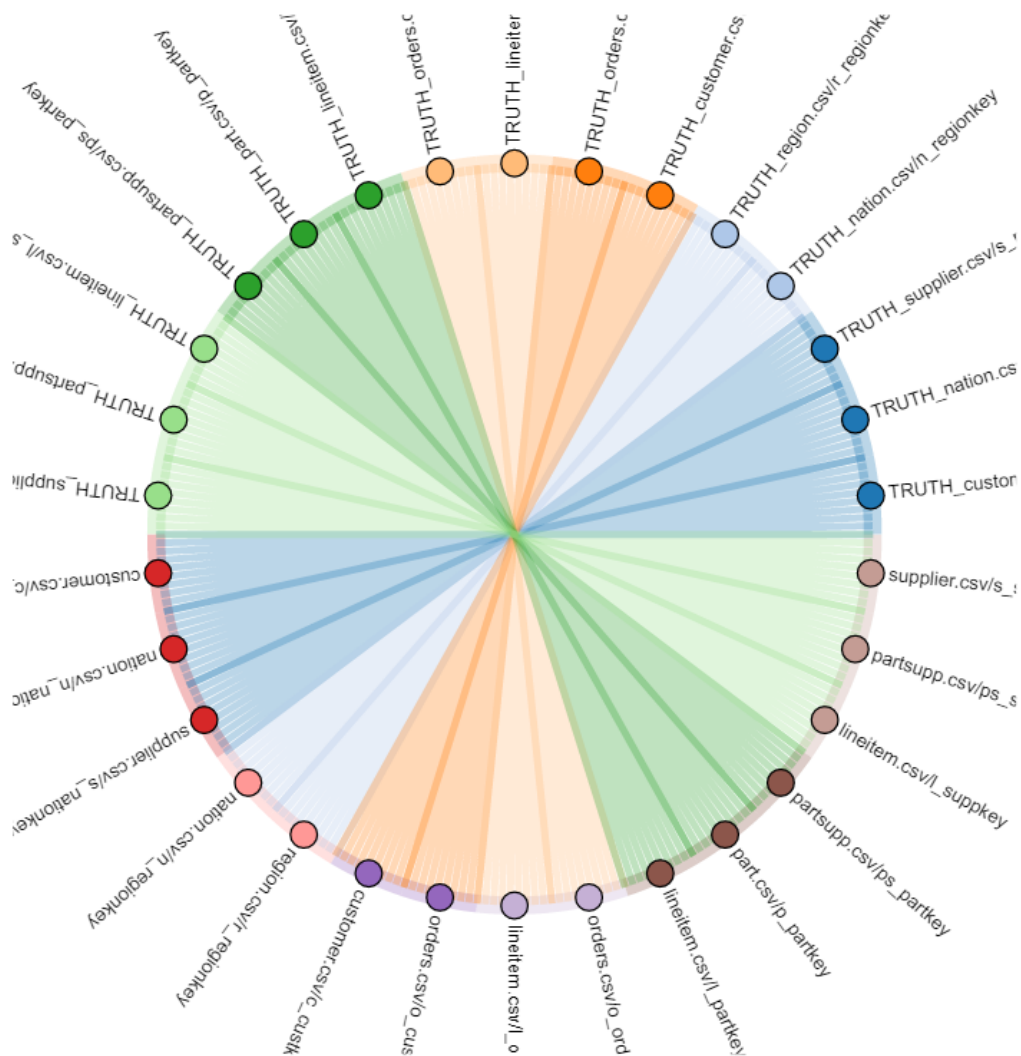


FIGURE E.1: Example chord diagram showing the clusters and comparison with the ground truth

Interpretation: Each column is represented with a circle in the outer circle of the diagram. We distinguish the formed clusters and the ground truth clusters by

prepending 'TRUTH_' before the column name of the ground truth clusters. The clusters that are formed, can be seen from the color of the circle representing the column. E.g. in figure E.1 we see 3 dark brown circles in the lower right corner, forming a cluster (partsupp.csv/ps_partkey, part.csv/p_partkey and lineitem.csv/l_partkey). The ground truth clusters depict how the clusters should actually be, whilst the other clusters are formed by our methodology. The chord which connects the columns, show where the columns are in our formed clusters, allowing us to see whether the supposedly clustered columns are clustered together or in some other cluster. Figure E.1 depicts the perfect scenario, where the formed clusters corresponds exactly to the ground truth clusters.

Bibliography

- Al-Ghanim, M., S. A. Noah, and T. M. Sembok (2011). "Automating XML schema matching: A composite approach". In: *Proceedings of the 2011 International Conference on Electrical Engineering and Informatics*, pp. 1–6. DOI: [10.1109/ICEEI.2011.6021797](https://doi.org/10.1109/ICEEI.2011.6021797).
- Alwan, Ali A et al. (2017). "A survey of schema matching research using database schemas and instances". In: *International Journal of Advanced Computer Science and Applications* 8.10.
- Arora, Sanjeev, Elad Hazan, and Satyen Kale (2012). "The Multiplicative Weights Update Method: a Meta-Algorithm and Applications". In: *Theory of Computing* 8.6, pp. 121–164. DOI: [10.4086/toc.2012.v008a006](https://doi.org/10.4086/toc.2012.v008a006). URL: <https://theoryofcomputing.org/articles/v008a006>.
- Bellahsene, Zohra et al. (2011). "On Evaluating Schema Matching and Mapping". In: *Schema Matching and Mapping*. Ed. by Zohra Bellahsene, Angela Bonifati, and Erhard Rahm. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 253–291. ISBN: 978-3-642-16518-4. DOI: [10.1007/978-3-642-16518-4_9](https://doi.org/10.1007/978-3-642-16518-4_9). URL: https://doi.org/10.1007/978-3-642-16518-4_9.
- Bergamaschi, Sonia et al. (Mar. 2001). "Semantic integration of heterogeneous information sources". In: *Data Knowledge Engineering* 36, pp. 215–249. DOI: [10.1016/S0169-023X\(00\)00047-1](https://doi.org/10.1016/S0169-023X(00)00047-1).
- Berlin, Jacob and Amihai Motro (2002). "Database Schema Matching Using Machine Learning with Feature Selection". In: *Advanced Information Systems Engineering*. Ed. by Anne Banks Pidduck et al. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 452–466. ISBN: 978-3-540-47961-1.
- Bernstein, Philip A., Jayant Madhavan, and Erhard Rahm (2011). "Generic Schema Matching, Ten Years Later". In: *Proc. VLDB Endow.* 4.11, 695–701. ISSN: 2150-8097. DOI: [10.14778/3402707.3402710](https://doi.org/10.14778/3402707.3402710). URL: <https://doi-org.tudelft.idm.oclc.org/10.14778/3402707.3402710>.
- Bernstein, Philip A. et al. (2004). "Industrial-Strength Schema Matching". In: *SIGMOD Rec.* 33.4, 38–43. ISSN: 0163-5808. DOI: [10.1145/1041410.1041417](https://doi.org/10.1145/1041410.1041417). URL: <https://doi-org.tudelft.idm.oclc.org/10.1145/1041410.1041417>.
- Blake, Roger (Jan. 2007). "A Survey of Schema Matching Research". In.
- Bogatu, Alex et al. (2020). "Dataset Discovery in Data Lakes". In: *CoRR abs/2011.10427*. arXiv: [2011.10427](https://arxiv.org/abs/2011.10427). URL: <https://arxiv.org/abs/2011.10427>.
- Burkart, Nadia and Marco F Huber (2021). "A survey on the explainability of supervised machine learning". In: *Journal of Artificial Intelligence Research* 70, pp. 245–317.
- Cafarella, Michael J., Alon Halevy, and Nodira Khoussainova (2009). "Data Integration for the Relational Web". In: *Proc. VLDB Endow.* 2.1, 1090–1101. ISSN: 2150-8097. DOI: [10.14778/1687627.1687750](https://doi.org/10.14778/1687627.1687750). URL: <https://doi-org.tudelft.idm.oclc.org/10.14778/1687627.1687750>.
- Castelo, Sonia et al. (2021). "Auctus: A Dataset Search Engine for Data Discovery and Augmentation". In: *Proc. VLDB Endow.* 14.12, 2791–2794. ISSN: 2150-8097.

- DOI: [10.14778/3476311.3476346](https://doi-org.tudelft.idm.oclc.org/10.14778/3476311.3476346). URL: <https://doi-org.tudelft.idm.oclc.org/10.14778/3476311.3476346>.
- Castro Fernandez, Raul et al. (2018). "Aurum: A Data Discovery System". In: *2018 IEEE 34th International Conference on Data Engineering (ICDE)*, pp. 1001–1012. DOI: [10.1109/ICDE.2018.00094](https://doi.org/10.1109/ICDE.2018.00094).
- Cervantes, Jair et al. (2020). "A comprehensive survey on support vector machine classification: Applications, challenges and trends". In: *Neurocomputing* 408, pp. 189–215. ISSN: 0925-2312. DOI: <https://doi.org/10.1016/j.neucom.2019.10.118>. URL: <https://www.sciencedirect.com/science/article/pii/S0925231220307153>.
- Chapman, Adriane et al. (2019). "Dataset search: a survey". In: *CoRR* abs/1901.00735. arXiv: [1901.00735](https://arxiv.org/abs/1901.00735). URL: <http://arxiv.org/abs/1901.00735>.
- Chawla, Nitesh V et al. (2002). "SMOTE: synthetic minority over-sampling technique". In: *Journal of artificial intelligence research* 16, pp. 321–357.
- Christensen, Scott D. et al. (2018). *Automated Data Discovery, Retrieval, Manipulation, and Publication using Python, Tethys, and HydroShare*. URL: <https://scholarsarchive.byu.edu/iemssconference/2018/Stream-B/14/>.
- Daniel, Naveen, S Lee, and Lalitha Naveen (2016). "Information discovery by analysts". In: *2016 American Finance Association annual meeting working paper*.
- Do, Hong and Erhard Rahm (Aug. 2002). "COMA - A System for Flexible Combination of Schema Matching Approaches." In: pp. 610–621. ISBN: 9781558608696. DOI: [10.1016/B978-155860869-6/50060-3](https://doi.org/10.1016/B978-155860869-6/50060-3).
- Do, Hong-Hai, Sergey Melnik, and Erhard Rahm (2003). "Comparison of Schema Matching Evaluations". In: *Web, Web-Services, and Database Systems*. Ed. by Akmal B. Chaudhri et al. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 221–237. ISBN: 978-3-540-36560-0.
- Doan, AnHai, Pedro Domingos, and Alon Y. Halevy (2001). "Reconciling Schemas of Disparate Data Sources: A Machine-Learning Approach". In: *Proceedings of the 2001 ACM SIGMOD International Conference on Management of Data*. SIGMOD '01. Santa Barbara, California, USA: Association for Computing Machinery, 509–520. ISBN: 1581133324. DOI: [10.1145/375663.375731](https://doi-org.tudelft.idm.oclc.org/10.1145/375663.375731). URL: <https://doi-org.tudelft.idm.oclc.org/10.1145/375663.375731>.
- Dong, Xin Luna and Divesh Srivastava (2013). "Big data integration". In: *2013 IEEE 29th International Conference on Data Engineering (ICDE)*, pp. 1245–1248. DOI: [10.1109/ICDE.2013.6544914](https://doi.org/10.1109/ICDE.2013.6544914).
- Dong, Yuyang et al. (2020). "Efficient Joinable Table Discovery in Data Lakes: A High-Dimensional Similarity-Based Approach". In: *CoRR* abs/2010.13273. arXiv: [2010.13273](https://arxiv.org/abs/2010.13273). URL: <https://arxiv.org/abs/2010.13273>.
- Duan, Songyun et al. (2012). "Instance-Based Matching of Large Ontologies Using Locality-Sensitive Hashing". In: *The Semantic Web – ISWC 2012*. Ed. by Philippe Cudré-Mauroux et al. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 49–64. ISBN: 978-3-642-35176-1.
- Duchateau, Fabien et al. (2009). "(Not) yet Another Matcher". In: *Proceedings of the 18th ACM Conference on Information and Knowledge Management*. CIKM '09. Hong Kong, China: Association for Computing Machinery, 1537–1540. ISBN: 9781605585123. DOI: [10.1145/1645953.1646165](https://doi-org.tudelft.idm.oclc.org/10.1145/1645953.1646165). URL: <https://doi-org.tudelft.idm.oclc.org/10.1145/1645953.1646165>.
- Elshwemy, Faten et al. (Mar. 2014). "Aggregation of similarity measures in schema matching based on generalized mean". In: pp. 74–79. ISBN: 978-1-4799-3481-2. DOI: [10.1109/ICDEW.2014.6818306](https://doi.org/10.1109/ICDEW.2014.6818306).

- Embley, David, David Jackman, and Li Xu (Nov. 2002). "Attribute Match Discovery in Information Integration: Exploiting Multiple Facets of Metadata". In: *Journal of the Brazilian Computer Society* 8. DOI: [10.1590/S0104-65002002000200004](https://doi.org/10.1590/S0104-65002002000200004).
- Embley, David W., Li Xu, and Yihong Ding (2004). "Automatic Direct and Indirect Schema Mapping: Experiences and Lessons Learned". In: *SIGMOD Rec.* 33.4, 14–19. ISSN: 0163-5808. DOI: [10.1145/1041410.1041413](https://doi.org/10.1145/1041410.1041413). URL: <https://doi.org/10.1145/1041410.1041413>.
- Engmann, Daniel and Sabine Maßmann (2007). "Instance Matching with COMA++". In: *BTW Workshops*.
- Gholami, Raof and Nikoo Fakhari (2017). "Chapter 27 - Support Vector Machine: Principles, Parameters, and Applications". In: *Handbook of Neural Computation*. Ed. by Pijush Samui, Sanjiban Sekhar, and Valentina E. Balas. Academic Press, pp. 515–535. ISBN: 978-0-12-811318-9. DOI: <https://doi.org/10.1016/B978-0-12-811318-9.00027-2>. URL: <https://www.sciencedirect.com/science/article/pii/B9780128113189000272>.
- Goebel, Michael and Le Gruenwald (1999). "A Survey of Data Mining and Knowledge Discovery Software Tools". In: *SIGKDD Explor. Newsl.* 1.1, 20–33. ISSN: 1931-0145. DOI: [10.1145/846170.846172](https://doi.org/10.1145/846170.846172). URL: <https://doi.org/10.1145/846170.846172>.
- Hack, Ulrike (2021). *What's the real story behind the explosive growth of data?* URL: <https://www.red-gate.com/blog/database-development/whats-the-real-story-behind-the-explosive-growth-of-data>.
- Hai Do, Hong (2007). *Schema Matching and Mapping-Based Data Integration: Architecture, Approaches and Evaluation*. Saarbrücken, DEU: VDM Verlag. ISBN: 3865509975.
- Halevy, Alon Y. (2001). "Answering Queries Using Views: A Survey". In: *The VLDB Journal* 10.4, 270–294. ISSN: 1066-8888. DOI: [10.1007/s007780100054](https://doi.org/10.1007/s007780100054). URL: <https://doi.org/10.1007/s007780100054>.
- He, Yeye, Kris Ganjam, and Xu Chu (2015). "Sema-join: joining semantically-related tables using big table corpora". In: *Proceedings of the VLDB Endowment* 8.12, pp. 1358–1369.
- Hidalgo, César A. et al. (2018). "The Principle of Relatedness". In: *Unifying Themes in Complex Systems IX*. Ed. by Alfredo J. Morales et al. Cham: Springer International Publishing, pp. 451–457. ISBN: 978-3-319-96661-8.
- King, Timothy (2019). *80 percent of your data will be unstructured in five years*. URL: <https://solutionsreview.com/data-management/80-percent-of-your-data-will-be-unstructured-in-five-years/>.
- Koksalmis, Emrah and Özgür Kabak (2019). "Deriving decision makers' weights in group decision making: An overview of objective methods". In: *Information Fusion* 49, pp. 146–160. ISSN: 1566-2535. DOI: <https://doi.org/10.1016/j.inffus.2018.11.009>. URL: <https://www.sciencedirect.com/science/article/pii/S1566253518303464>.
- Koutras, Christos et al. (2021). "Valentine: Evaluating Matching Techniques for Dataset Discovery". In: *2021 IEEE 37th International Conference on Data Engineering (ICDE)*, pp. 468–479. DOI: [10.1109/ICDE51399.2021.00047](https://doi.org/10.1109/ICDE51399.2021.00047).
- Krawiec, Krzysztof (2002). "Genetic programming-based construction of features for machine learning and knowledge discovery tasks". In: *Genetic Programming and Evolvable Machines* 3.4, pp. 329–343.
- Lee, Yoonkyong et al. (Jan. 2007). "ETuner: Tuning schema matching software using synthetic scenarios". In: *VLDB J.* 16, pp. 97–122. DOI: [10.1007/s00778-006-0024-z](https://doi.org/10.1007/s00778-006-0024-z).

- Lenzerini, Maurizio (2002). "Data Integration: A Theoretical Perspective". In: *Proceedings of the Twenty-First ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*. PODS '02. Madison, Wisconsin: Association for Computing Machinery, 233–246. ISBN: 1581135076. DOI: [10.1145/543613.543644](https://doi.org/10.1145/543613.543644). URL: <https://doi-org.tudelft.idm.oclc.org/10.1145/543613.543644>.
- Li, Wen-Syan and Chris Clifton (n.d.). "Semantic Integration in Heterogeneous Databases Using Neural Networks". In: *Proceedings of the 20th International Conference on Very Large Data Bases* (). URL: <https://par.nsf.gov/biblio/10077889>.
- (Apr. 2000). "SEMINT: A tool for identifying attribute correspondences in heterogeneous databases using neural networks". In: *Data Knowl. Eng.* 33, pp. 49–84. DOI: [10.1016/S0169-023X\(99\)00044-0](https://doi.org/10.1016/S0169-023X(99)00044-0).
- Li, Wenwen (2010). "Automated Data Discovery, Reasoning and Ranking in Support of Building an Intelligent Geospatial Search Engine". AAI3421139. PhD thesis. USA. ISBN: 9781124201436.
- MacMillan, Don (2014). "Data Sharing and Discovery: What Librarians Need to Know". In: *The Journal of Academic Librarianship* 40.5, pp. 541–549. ISSN: 0099-1333. DOI: <https://doi.org/10.1016/j.acalib.2014.06.011>. URL: <https://www.sciencedirect.com/science/article/pii/S0099133314000950>.
- Madhavan, Jayant, Philip Bernstein, and Erhard Rahm (July 2001). "Generic Schema Matching with Cupid". In: *Proc 27th VLDB Conference*.
- Madhavan, Jayant et al. (June 2003). "Corpus-based Schema Matching". In.
- Mehdi, Osama A., Hamidah Ibrahim, and Lilly Suriani Affendey (2012). "Instance based Matching using Regular Expression". In: *Procedia Computer Science* 10. ANT 2012 and MobiWIS 2012, pp. 688–695. ISSN: 1877-0509. DOI: <https://doi.org/10.1016/j.procs.2012.06.088>. URL: <https://www.sciencedirect.com/science/article/pii/S1877050912004450>.
- Milo, Tova and Sagit Zohar (1998). "Using Schema Matching to Simplify Heterogeneous Data Translation". In: *Proceedings of the 24rd International Conference on Very Large Data Bases*. VLDB '98. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 122–133. ISBN: 1558605665.
- Mork, P. et al. (2006). "Integration Workbench: Integrating Schema Integration Tools". In: *22nd International Conference on Data Engineering Workshops (ICDEW'06)*, pp. 3–3. DOI: [10.1109/ICDEW.2006.69](https://doi.org/10.1109/ICDEW.2006.69).
- Nguyen, Hien M, Eric W Cooper, and Katsuari Kamei (2009). "Borderline oversampling for imbalanced data classification". In: *Proceedings: Fifth International Workshop on Computational Intelligence & Applications*. Vol. 2009. 1. IEEE SMC Hiroshima Chapter, pp. 24–29.
- Nguyen, Hoang Vu et al. (2014). "Detecting Correlated Columns in Relational Databases with Mixed Data Types". In: *Proceedings of the 26th International Conference on Scientific and Statistical Database Management*. SSDBM '14. Aalborg, Denmark: Association for Computing Machinery. ISBN: 9781450327220. DOI: [10.1145/2618243.2618251](https://doi.org/10.1145/2618243.2618251). URL: <https://doi.org/10.1145/2618243.2618251>.
- Özsu, M Tamer and Patrick Valduriez (1999). *Principles of distributed database systems*. Vol. 2. Springer.
- Pei, Jin, Jun Hong, and David Bell (2006). "A Novel Clustering-Based Approach to Schema Matching". In: *Advances in Information Systems*. Ed. by Tatyana Yakhno and Erich J. Neuhold. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 60–69. ISBN: 978-3-540-46292-7.
- Peukert, Eric, Sabine Maßmann, and Kathleen König (2010). "Comparing Similarity Combination Methods for Schema Matching". In: *GI Jahrestagung*.

- Rahm, Erhard and Philip Bernstein (Dec. 2001). "A Survey of Approaches to Automatic Schema Matching." In: *VLDB J.* 10, pp. 334–350. DOI: [10.1007/s007780100057](https://doi.org/10.1007/s007780100057).
- Shvaiko, Pavel and Jérôme Euzenat (2005). "A Survey of Schema-Based Matching Approaches". In: *Journal on Data Semantics IV*. Ed. by Stefano Spaccapietra. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 146–171. ISBN: 978-3-540-31447-9.
- Stonebraker, Michael, Ihab F Ilyas, et al. (2018). "Data Integration: The Current Status and the Way Forward." In: *IEEE Data Eng. Bull.* 41.2, pp. 3–9.
- Stonebraker, Michael et al. (2013). "Data Curation at Scale: The Data Tamer System". In: *In CIDR 2013*.
- Sun, Yanmin, Andrew KC Wong, and Mohamed S Kamel (2009). "Classification of imbalanced data: A review". In: *International journal of pattern recognition and artificial intelligence* 23.04, pp. 687–719.
- Sutanta, Edhy et al. (2016). "Survey: Models and Prototypes of Schema Matching." In: *International Journal of Electrical & Computer Engineering (2088-8708)* 6.3.
- Tu, KeWei and Yong Yu (2005). "CMC: Combining Multiple Schema-Matching Strategies Based on Credibility Prediction". In: *Database Systems for Advanced Applications*. Ed. by Lizhu Zhou, Beng Chin Ooi, and Xiaofeng Meng. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 888–893. ISBN: 978-3-540-32005-0.
- Wang, Jiannan, Guoliang Li, and Jianhua Feng (2014). "Extending String Similarity Join to Tolerant Fuzzy Token Matching". In: *ACM Trans. Database Syst.* 39.1. ISSN: 0362-5915. DOI: [10.1145/2535628](https://doi.org/10.1145/2535628). URL: <https://doi-org.tudelft.idm.oclc.org/10.1145/2535628>.
- Weikum, Gerhard (July 2013). "Data Discovery". In: *Data Science Journal* 12, GRDI26–GRDI31. DOI: [10.2481/dsj.GRDI-005](https://doi.org/10.2481/dsj.GRDI-005).
- wu, Mingfang et al. (Jan. 2019). "Data Discovery Paradigms: User Requirements and Recommendations for Data Repositories". In: *Data Science Journal* 18. DOI: [10.5334/dsj-2019-003](https://doi.org/10.5334/dsj-2019-003).
- Yakout, Mohamed et al. (2012). "InfoGather: Entity Augmentation and Attribute Discovery by Holistic Matching with Web Tables". In: *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*. SIGMOD '12. Scottsdale, Arizona, USA: Association for Computing Machinery, 97–108. ISBN: 9781450312479. DOI: [10.1145/2213836.2213848](https://doi.org/10.1145/2213836.2213848). URL: <https://doi-org.tudelft.idm.oclc.org/10.1145/2213836.2213848>.
- Zhang, Meihui et al. (2011). "Automatic Discovery of Attributes in Relational Databases". In: *Proceedings of the 2011 ACM SIGMOD International Conference on Management of Data*. SIGMOD '11. Athens, Greece: Association for Computing Machinery, 109–120. ISBN: 9781450306614. DOI: [10.1145/1989323.1989336](https://doi.org/10.1145/1989323.1989336). URL: <https://doi.org/10.1145/1989323.1989336>.
- Zhao, Huimin and Sudha Ram (2007). "Combining Schema and Instance Information for Integrating Heterogeneous Data Sources". In: *Data Knowl. Eng.* 61.2, 281–303. ISSN: 0169-023X. DOI: [10.1016/j.datak.2006.06.004](https://doi.org/10.1016/j.datak.2006.06.004). URL: <https://doi-org.tudelft.idm.oclc.org/10.1016/j.datak.2006.06.004>.
- Zhu, Erkang, Yeye He, and Surajit Chaudhuri (2017). "Auto-Join: Joining Tables by Leveraging Transformations". In: *Proc. VLDB Endow.* 10.10, 1034–1045. ISSN: 2150-8097. DOI: [10.14778/3115404.3115409](https://doi.org/10.14778/3115404.3115409). URL: <https://doi-org.tudelft.idm.oclc.org/10.14778/3115404.3115409>.
- Zhu, Erkang et al. (2019). "JOSIE: Overlap Set Similarity Search for Finding Joinable Tables in Data Lakes". In: *Proceedings of the 2019 International Conference on Management of Data*. SIGMOD '19. Amsterdam, Netherlands: Association for Computing Machinery, 847–864. ISBN: 9781450356435. DOI: [10.1145/3299869](https://doi.org/10.1145/3299869).

3300065. URL: <https://doi-org.tudelft.idm.oclc.org/10.1145/3299869.3300065>.
- Ziegler, Patrick and Klaus R. Dittrich (2007). "Data Integration — Problems, Approaches, and Perspectives". In: *Conceptual Modelling in Information Systems Engineering*. Ed. by John Krogstie, Andreas Lothe Opdahl, and Sjaak Brinkkemper. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 39–58. ISBN: 978-3-540-72677-7. DOI: [10.1007/978-3-540-72677-7_3](https://doi.org/10.1007/978-3-540-72677-7_3). URL: https://doi.org/10.1007/978-3-540-72677-7_3.
- Zikopoulos, P. and C. Eaton (2011). *Understanding Big Data: Analytics for Enterprise Class Hadoop and Streaming Data*. McGraw-Hill Education. ISBN: 9780071790543. URL: <https://books.google.nl/books?id=0sJqV1t4UVsC>.