Dataset Proximity Mining for Supporting Schema Matching and Data Lake Governance

Ph.D. Dissertation

Ayman Alserafi

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH

Dataset proximity mining for supporting schema matching and data lake governance

Ayman Alserafi

ADVERTIMENT La consulta d'aquesta tesi queda condicionada a l'acceptació de les següents condicions d'ús: La difusió d'aquesta tesi per mitjà del repositori institucional UPCommons (http://upcommons.upc.edu/tesis) i el repositori cooperatiu TDX (http://www.tdx.cat/) ha estat autoritzada pels titulars dels drets de propietat intel·lectual **únicament per a usos privats** emmarcats en activitats d'investigació i docència. No s'autoritza la seva reproducció amb finalitats de lucre ni la seva difusió i posada a disposició des d'un lloc aliè al servei UPCommons o TDX. No s'autoritza la presentació del seu contingut en una finestra o marc aliè a UPCommons (*framing*). Aquesta reserva de drets afecta tant al resum de presentació de la tesi com als seus continguts. En la utilització o cita de parts de la tesi és obligat indicar el nom de la persona autora.

ADVERTENCIA La consulta de esta tesis queda condicionada a la aceptación de las siguientes condiciones de uso: La difusión de esta tesis por medio del repositorio institucional UPCommons (http://upcommons.upc.edu/tesis) y el repositorio cooperativo TDR (http://www.tdx.cat/?locale-attribute=es) ha sido autorizada por los titulares de los derechos de propiedad intelectual **únicamente para usos privados enmarcados** en actividades de investigación y docencia. No se autoriza su reproducción con finalidades de lucro ni su difusión y puesta a disposición desde un sitio ajeno al servicio UPCommons No se autoriza la presentación de su contenido en una ventana o marco ajeno a UPCommons (*framing)*. Esta reserva de derechos afecta tanto al resumen de presentación de la tesis como a sus contenidos. En la utilización o cita de partes de la tesis es obligado indicar el nombre de la persona autora.

WARNING On having consulted this thesis you're accepting the following use conditions: Spreading this thesis by the institutional repository UPCommons (http://upcommons.upc.edu/tesis) and the cooperative repository TDX (http://www.tdx.cat/?locale- attribute=en) has been authorized by the titular of the intellectual property rights **only for private uses** placed in investigation and teaching activities. Reproduction with lucrative aims is not authorized neither its spreading nor availability from a site foreign to the UPCommons service. Introducing its content in a window or frame foreign to the UPCommons service is not authorized (framing). These rights affect to the presentation summary of the thesis as well as to its contents. In the using or citation of parts of the thesis it's obliged to indicate the name of the author.

Dataset Proximity Mining for Supporting Schema Matching and Data Lake Governance

Ph.D. Dissertation

Ayman Alserafi

Supervisors:
Prof. Alberto Abelló
Prof. Oscar Romero
Prof. Toon Calders

Dissertation submitted on November, 2020

A thesis submitted to Barcelona School of Informatics at Universitat Politècnica de Catalunya, BarcelonaTech (UPC) and the Faculty of Engineering at Université Libre De Bruxelles (ULB), in partial fulfilment of the requirements within the scope of the IT4BI-DC programme for the joint Ph.D. degree in computer science. The thesis is not submitted to any other organization at the same time.

Thesis title:	Dataset Proximity Mining for Supporting Schema Matching and Data Lake Governance
Thesis submitted:	November, 2020
PhD Supervisors:	Prof. Alberto Abelló Universitat Politècnica de Catalunya, BarcelonaTech, Spain Prof. Oscar Romero Universitat Politècnica de Catalunya, BarcelonaTech, Spain Prof. Toon Calders Université Libre de Bruxelles, Brussels, Belgium and Universiteit Antwerpen, Antwerp, Belgium
PhD Committee:	Prof. Wolfgang Lehner Technische Universität Dresden, Germany Prof. Patrick Marcel Université de Tours, France Prof. Jose Francisco Aldana Montes Universidad de Málaga, Spain
PhD Series:	Barcelona School of Informatics, Universitat Politècnica de Catalunya, BarcelonaTech
Doctoral Programme:	The Erasmus Mundus Joint Doctorate in Information Technologies for Business Intelligence - Doctoral College (IT4BI-DC)

© Copyright by Ayman Alserafi. Author has obtained the right to include the published and accepted articles in the thesis, with a condition that they are cited, DOI pointers and/or copyright/credits are placed prominently in the references.

Printed in Spain, 2020

Abstract

A task is only as difficult as one perceives it. Everything can be dissected into its smaller simpler parts. If a task seems unattainable, break it into smaller pieces and it will seem simpler and more interesting to achieve or study.

With the huge growth in the amount of data generated by information systems, it is common practice today to store datasets in their raw formats (i.e., without any data preprocessing or transformations) in large-scale data repositories called Data Lakes (DLs). Such repositories store datasets from heterogeneous subject-areas (covering many business topics) and with many different schemata. Therefore, it is a challenge for data scientists using the DL for data analysis to find relevant datasets for their analysis tasks without any support or data governance. The goal is to be able to extract metadata and information about datasets stored in the DL to support the data scientist in finding relevant sources. This shapes the main goal of this thesis, where we explore different techniques of data profiling, holistic schema matching and analysis recommendation to support the data scientist. We propose a novel framework based on supervised machine learning to automatically extract metadata describing datasets, including computation of their similarities and data overlaps using holistic schema matching techniques. We use the extracted relationships between datasets in automatically categorizing them to support the data scientist in finding relevant datasets with intersection between their data. This is done via a novel metadata-driven technique called proximity mining which consumes the extracted metadata via automated data mining algorithms in order to detect related datasets and to propose relevant categories for them. We focus on flat (tabular) datasets organised as rows of data instances and columns of attributes describing the instances. Our proposed framework uses the following four main techniques: (1) Instance-based schema matching for detecting relevant data items between heterogeneous datasets, (2) Dataset level metadata extraction and proximity mining for detecting related datasets, (3) Attribute level metadata extraction and proximity mining for detecting related datasets, and finally, (4) Automatic dataset categorization via supervised k-Nearest-Neighbour (kNN) techniques. We implement our proposed algorithms via a prototype that shows the feasibility of this framework. We apply the prototype in an experiment on a real-world DL scenario to prove the feasibility, effectiveness and efficiency of our approach, whereby we were able to achieve high recall rates and efficiency gains while improving the computational space and time consumption by two orders of magnitude via our proposed early-pruning and pre-filtering techniques in comparison to classical instance-based schema matching techniques. This proves the effectiveness of our proposed automatic methods in the early-pruning and pre-filtering tasks for holistic schema matching and the automatic dataset categorisation, while also demonstrating improvements over human-based data analysis for the same tasks.

Keywords: *Data Lake Governance, Dataset Similarity Mining, Holistic Schema Matching, Metadata Management, Supervised Machine Learning*

Resum

Amb l'enorme creixement de la quantitat de dades generades pels sistemes d'informació, és habitual avui en dia emmagatzemar conjunts de dades en els seus formats bruts (és a dir, sense cap preprocessament de dades ni transformacions) en dipòsits de dades a gran escala anomenats Data Lakes (DL). Aquests dipòsits emmagatzemen conjunts de dades d'àrees temàtiques heterogènies (que abasten molts temes empresarials) i amb molts esquemes diferents. Per tant, és un repte per als científics de dades que utilitzin la DL per a l'anàlisi de dades trobar conjunts de dades rellevants per a les seves tasques d'anàlisi sense cap suport ni govern de dades. L'objectiu és poder extreure metadades i informació sobre conjunts de dades emmagatzemats a la DL per donar suport al científic en trobar fonts rellevants. Aquest és l'objectiu principal d'aquesta tesi, on explorem diferents tècniques de perfilació de dades, concordança d'esquemes holístics i recomanació d'anàlisi per donar suport al científic. Proposem un nou marc basat en l'aprenentatge automatitzat supervisat per extreure automàticament metadades que descriuen conjunts de dades, incloent el càlcul de les seves similituds i coincidències de dades mitjançant tècniques de concordança d'esquemes holístics. Utilitzem les relacions extretes entre conjunts de dades per categoritzar-les automàticament per donar suport al científic del fet de trobar conjunts de dades rellevants amb la intersecció entre les seves dades. Això es fa mitjançant una nova tècnica basada en metadades anomenada mineria de proximitat que consumeix els metadades extrets mitjançant algoritmes automatitzats de mineria de dades per tal de detectar conjunts de dades relacionats i proposar-ne categories rellevants. Ens centrem en conjunts de dades plans (tabulars) organitzats com a files d'instàncies de dades i columnes d'atributs que descriuen les instàncies. El nostre marc proposat utilitza les quatre tècniques principals següents: (1) Esquema de concordança basat en instàncies per detectar ítems rellevants de dades entre conjunts de dades heterogènies, (2) Extracció de metadades de nivell de dades i mineria de proximitat per detectar conjunts de dades relacionats, (3) Extracció de metadades a nivell de atribut i mineria de proximitat per detectar conjunts de dades relacionats i, finalment, (4) Categorització de conjunts de dades automàtica mitjançant tècniques supervisades per k-Nearest-Neighbour

(kNN). Posem en pràctica els nostres algorismes proposats mitjançant un prototip que mostra la viabilitat d'aquest marc. El prototip s'experimenta en un escenari DL real del món per demostrar la viabilitat, l'eficàcia i l'eficiència del nostre enfocament, de manera que hem pogut aconseguir elevades taxes de record i guanys d'eficiència alhora que millorem el consum computacional d'espai i temps mitjançant dues ordres de magnitud mitjançant el nostre es van proposar tècniques de poda anticipada i pre-filtratge en comparació amb tècniques de concordança d'esquemes basades en instàncies clàssiques. Això demostra l'efectivitat dels nostres mètodes automàtics proposats en les tasques de poda inicial i pre-filtratge per a la coincidència d'esquemes holístics i la classificació automàtica del conjunt de dades, tot demostrant també millores en l'anàlisi de dades basades en humans per a les mateixes tasques.

Résumé

Avec l'énorme croissance de la quantité de données générées par les systèmes d'information, il est courant aujourd'hui de stocker des ensembles de données (datasets) dans leurs formats bruts (c'est-à-dire sans prétraitement ni transformation de données) dans des référentiels de données à grande échelle appelés Data Lakes (DL). Ces référentiels stockent des ensembles de données provenant de domaines hétérogènes (couvrant de nombreux sujets commerciaux) et avec de nombreux schémas différents. Par conséquent, il est difficile pour les data-scientists utilisant les DL pour l'analyse des données de trouver des datasets pertinents pour leurs tâches d'analyse sans aucun support ni gouvernance des données. L'objectif est de pouvoir extraire des métadonnées et des informations sur les datasets stockés dans le DL pour aider le data-scientist à trouver des sources pertinentes. Cela constitue l'objectif principal de cette thèse, où nous explorons différentes techniques de profilage de données, de correspondance holistique de schéma et de recommandation d'analyse pour soutenir le data-scientist. Nous proposons une nouvelle approche basée sur l'intelligence artificielle, spécifiquement l'apprentissage automatique supervisé, pour extraire automatiquement les métadonnées décrivant les datasets, calculer automatiquement les similitudes et les chevauchements de données entre ces ensembles en utilisant des techniques de correspondance holistique de schéma. Les relations entre datasets ainsi extraites sont utilisées pour catégoriser automatiquement les datasets, afin d'aider le data-scientist à trouver des datasets pertinents avec intersection entre leurs données. Cela est fait via une nouvelle technique basée sur les métadonnées appelée proximity mining, qui consomme les métadonnées extraites via des algorithmes de data mining automatisés afin de détecter des datasets connexes et de leur proposer des catégories pertinentes. Nous nous concentrons sur des datasets plats (tabulaires) organisés en rangées d'instances de données et en colonnes d'attributs décrivant les instances. L'approche proposée utilise les quatres principales techniques suivantes: (1) Correspondance de schéma basée sur l'instance pour détecter les éléments de données pertinents entre des datasets hétérogènes, (2) Extraction de métadonnées au niveau du dataset et proximity mining pour détecter les datasets connexes, (3) Extraction de métadonnées au niveau des attributs et proximity mining pour détecter des datasets connexes, et enfin, (4) catégorisation automatique des datasets via des techniques supervisées k-Nearest-Neighbour (kNN). Nous implémentons les algorithmes proposés via un prototype qui montre la faisabilité de cette approche. Nous appliquons ce prototype à une scénario DL du monde réel pour prouver la faisabilité, l'efficacité et l'efficience de notre approche, nous permettant d'atteindre des taux de rappel élevés et des gains d'efficacité, tout en diminuant le coût en espace et en temps de deux ordres de grandeur, via nos techniques proposées d'élagage précoce et de pré-filtrage, comparé aux techniques classiques de correspondance de schémas basées sur les instances. Cela prouve l'efficacité des méthodes automatiques proposées dans les tâches d'élagage précoce et de pré-filtrage pour la correspondance de schéma holistique et la cartegorisation automatique des datasets, tout en démontrant des améliorations par rapport à l'analyse de données basée sur l'humain pour les mêmes tâches.

Acknowledgements

We don't work in isolation, but rather in stronger, more efficient and successful groups. [...] Karma does take place, and we reap what we sow! Thus, treat others like you'd like to be treated.

This thesis would not have been possible if it was not for the support of many people who helped me along the journey in many positive ways. Firstly, I would like to thank my thesis supervisors for their huge efforts with my thesis and all our fruitful discussions. I have learned a lot from them in many useful ways. They taught me to achieve professional rigorous research, to keep expanding my knowledge and to never give up along the (rocky!) path till the finish line. I would also like to thank my family for their continuous support during the most difficult times. They deserve all the sincere appreciation in the world. All the grateful appreciation to my sincere friends (you know yourselves!) who helped me adapt with all the new experiences and for always being there when I need them (especially during the low times). Without them I would not have been able to finish this PhD project. I would like to thank all my colleagues from IT4BI-DC, UPC, and ULB who made it an amazing work environment for learning and collaborating together. I learned something useful from each and everyone of this big research group (and academic family). I'm also really thankful for all the assistance by the technical support staff at the ULB and UPC labs. This thesis would not have been possible without the grateful funding and contributions by the European Commission via the research grant that supported me to complete my thesis and my international research stays. I am also thankful for my country, the cradle of ancient civilisations, for making me who I am and for engraving in me all the good values which support me in my life. It taught me the love for all of humanity, and the respect for all our beautiful diversity. Finally, I would like to end this with a dedication by wishing all the prosperity for the nations and peoples of Europe, Africa, my homeland, and the whole world. May God bless them all and their future generations of scientists and engineers. With our continuous collaboration we will achieve a better world and future for everyone and the whole of humanity!

27th of March, 2020

Contents

Abstract **v**

Resum **vii**

Résumé **ix**

Acknowledgements **xi**

List of Figures . . . xv

List of Tables . . . xviii

Abbreviations **xx**

Thesis Details **xxi**

1 Introduction **1**

1 Motivation . . . 1

2 Background and State-of-the-art . . . 2

2.1 Data Lakes and Tabular Datasets . . . 2

2.2 Data Lake Governance . . . 3

3 Techniques and Challenges . . . 10

3.1 Schema Matching . . . 11

3.2 Dataset Similarity Computation . . . 14

3.3 Similarity Models Learning . . . 15

4 Thesis Objectives and Research Questions . . . 18

5 Thesis Overview . . . 20

5.1 Proximity Mining Framework . . . 20

5.2 DL Categorization . . . 23

5.3 Metadata Query Interface . . . 25

6 Thesis Contributions . . . 25

7 Structure of the Thesis . . . 27

7.1 Chapter 2: Instance-level value-based schema matching for mining proximity between datasets . . . 28

7.2 Chapter 3: Dataset-level content metadata based proximity mining . . . 28
7.3 Chapter 4: Attribute-level content metadata based proximity mining for pre-filtering schema matching . . . 28
7.4 Chapter 5: Automatic categorization of datasets using proximity mining . . . 29
7.5 Chapter 6: Prox-mine tool for browsing DLs using proximity mining . . . 29

2 Instance-level value-based schema matching for computing dataset similarity 31

1 Introduction . . . 33
2 Related Work . . . 35
3 Motivational Case-Study . . . 36
4 A Framework for Content Metadata Management . . . 37
5 The CM4DL Prototype . . . 39
5.1 Prototype Architecture . . . 40
5.2 Ontology Alignment Component . . . 41
5.3 Dataset Comparison Algorithm . . . 43
6 Experiments and Results . . . 45
7 Discussion . . . 48
8 Conclusion and Future Work . . . 49

3 Dataset-level content metadata based proximity mining for computing dataset similarity 51

1 Introduction . . . 53
2 Problem Statement . . . 54
3 Related Work . . . 55
4 The DS-Prox Approach . . . 56
4.1 The Meta-Features Distance Measures . . . 57
4.2 The Approach . . . 58
5 Experimental Evaluation . . . 61
5.1 Datasets . . . 61
5.2 Experimental Setup . . . 62
5.3 Results . . . 63
5.4 Discussion . . . 64
6 Conclusion and Future Work . . . 66

4 Attribute-level content metadata based proximity mining for pre-filtering schema matching 68

1 Introduction . . . 70
2 Related Work . . . 72
3 Preliminaries . . . 75

4 Approach: Metadata-based Proximity Mining for Pre-filtering Schema Matching . . . 79
4.1 Proximity Metrics: Meta-features Distances . . . 79
4.2 Supervised Proximity Mining . . . 81
4.3 Pre-filtering Dataset Pairs for Schema Matching . . . 88
5 Experimental Evaluation . . . 88
5.1 Datasets . . . 89
5.2 Evaluation Metrics . . . 92
5.3 Experiment 1: Attribute-level Models . . . 94
5.4 Experiment 2: Dataset-level Models . . . 95
5.5 Experiment 3: Computational Performance Evaluation . 103
5.6 Generalisability . . . 105
6 Conclusion . . . 105

5 Automatic categorization of datasets using proximity mining 107
1 Introduction . . . 109
2 Preliminaries . . . 111
2.1 Proximity Mining: Meta-features Metrics and Models . 113
3 DS-kNN: a Proximity Mining Based k-Nearest-Neighbour Algorithm for Categorizing Datasets . . . 115
4 Experimental Evaluation . . . 117
4.1 Dataset: OpenML DL Ground-truth . . . 117
4.2 Experimental Setup . . . 118
4.3 Results . . . 121
4.4 Validation Experiment . . . 131
5 Related Work . . . 133
6 Conclusion . . . 134

6 Prox-mine tool for browsing DLs using proximity mining 135
1 Introduction . . . 136
2 Data Lake Index . . . 137
3 Similarity Search . . . 137
4 Dataset Categorization . . . 138
5 Dataset Matching . . . 140
5.1 New Dataset Matching . . . 141
6 Proximity Graph . . . 143

7 Conclusions and Future Directions 148
1 Conclusions . . . 149
2 Future Directions . . . 150

Bibliography 152
References . . . 152

List of Figures

1.1 A flat structured dataset consisting of tabular data organised as attributes and instances . . . 3
1.2 DL governance classification and tasks . . . 4
1.3 An example of schema matching and dataset similarity computation . . . 11
1.4 Classification of schema matching techniques . . . 13
1.5 Example of a decision tree model for classification of related dataset pairs . . . 17
1.6 An ensemble of decision trees for classification . . . 18
1.7 Overview of the proposed proxmity mining framework . . . 21
1.8 The proximity mining metadata management process . . . 22
1.9 A proximity graph showing topic-wise groupings of interlinked datasets in the DL with their similarity scores . . . 24
1.10 DL content metadata management and analysis process . . . 25

2.1 The Metadata Management BPMN Process Model . . . 37
2.2 The EXP01 Metadata Exploitation Sub-Process Model . . . 38
2.3 CM4DL System Architecture . . . 40
2.4 Performance analysis of CM4DL in the OpenML experiments . 47

3.1 Similarity relationships between two pairs of datasets . . . 55
3.2 DS-Prox: supervised machine learning . . . 58
3.3 DS-Prox cut-off thresholds tuning . . . 60
3.4 Recall-efficiency plots (left column) and recall-precision plots (right column) for experiments 1,2,3 and 4 in each row . . . 67

4.1 The stratified holistic schema matching approach at different levels of granularity. . . . 71
4.2 The dependencies of components in the metadata-based proximity mining approach for pre-filtering schema matching. . . . 76
4.3 Final output of our approach consisting of similarity relationships between two pairs of datasets. . . . 78
4.4 An overview of the process to ***build*** the supervised ensemble models in our proposed datasets proximity mining approach using previously manually annotated dataset pairs. . . . 82
4.5 Proximity Mining: supervised machine learning for predicting related data objects. . . . 83
4.6 Different normal distributions for assigning weights to ranked attribute linkages. . . . 86

4.7 An overview of the process to ***apply*** the learnt supervised models in our approach for pre-filtering previously unseen dataset pairs independent of the build process. 87
4.8 A 10-fold cross-validation experimental setup consisting of alternating folds in training and test roles. Image adapted from the book: Raschka S (2015) Python Machine Learning. 1st Edition. Packt Publishing, Birmingham, UK. 97
4.9 Classification accuracy from 10-fold cross-validation of dataset pairs pre-filtering models. 98
4.10 Kappa statistic from 10-fold cross-validation of dataset pairs pre-filtering models. 98
4.11 ROC statistic from 10-fold cross-validation of dataset pairs pre-filtering models. 98
4.12 Recall against efficiency gain for the different supervised models.100
4.13 Recall against precision for the different supervised models. . . 100
4.14 Recall against efficiency gain for the different metric types. . . 100
4.15 Recall against precision for the different metric types. 100

5.1 A visualisation of the output from DS-kNN data lake (DL) categorization. A proximity graph shows the datasets as nodes and the proximity scores as edges between nodes. Fig.(a) complete DL and Fig. (b) a zoomed-in view highlighted by the red box in (a) 110
5.2 The data lake categorization scenario using k-NN proximity mining 113
5.3 Performance of DS-kNN using **k=1**, different models, different ground-truths, and different category sizes 122
5.4 Performance of DS-kNN using **k=3**, different models, different ground-truths, and different category sizes 123
5.5 Performance of DS-kNN using **k=5**, different models, different ground-truths, and different category sizes 124
5.6 Performance of DS-kNN using **k=7**, different models, different ground-truths, and different category sizes 125

6.1 The input screen for the similarity search component of Prox-mine138
6.2 The output screen for the similarity search component of Prox-mine 138
6.3 The input screen for the dataset categorization component of Prox-mine 139
6.4 The output screens for the dataset categorization component of Prox-mine. 140
6.5 The input screen for the dataset matching component of Prox-mine 141

6.6 The output screens for the dataset matching component of Prox-mine. 142
6.7 The input screen for the new dataset matching component of Prox-mine 143
6.8 The output screen for the new dataset matching component of Prox-mine 144
6.9 The input screen for the proximity graph component of Prox-mine 145
6.10 An overview of the output proximity graph component of Prox-mine, where (a) gives a zoomed-out view and (b) gives a zoomed-in view 145
6.11 The search and filtration panel of the output proximity graph component of Prox-mine, where (a) gives a view of the category selector in the left-panel and (b) gives the result of applying the filtration step 146
6.12 The selection of a specific dataset node in the proximity graph and the relationships information panel shown on the right side 147

List of Tables

1.1 Current types of metadata tools for data lakes 7

2.1 Description of OpenML datasets 36
2.2 Example Cross-dataset Relationships 43
2.3 Results of Manual Annotation 47

3.1 DS-Prox meta-features 57
3.2 A description of the OpenML samples collected 62
3.3 An example of pairs of datasets from the all-topics sample from OpenML 62
3.4 A description of the experiments conducted 63

4.1 Schema matching techniques state-of-the-art comparison 75
4.2 Schema matching pre-filtering functions 76
4.3 Attribute level content meta-features 80
4.4 Description of the OML01 datasets 90
4.5 Example Cross-dataset Attribute Relationships from OML01 . . 91
4.6 Description of the OML02 datasets 92
4.7 An example of pairs of datasets from the OML02 sample from OpenML 92
4.8 The significance of the Kappa statistic 93
4.9 The significance of the ROC statistic 93
4.10 Performance evaluation of attribute pairs proximity models . . 95
4.11 Spearman rank correlation for the different meta-features. We aggregate minimum (Min.), average (Avg.), maximum (Max.), & standard deviation (Std. Dev.) for different meta-feature types. 99
4.12 The standard deviation of each evaluation measure for 10-fold cross-validation of each dataset pairs pre-filtering model, where $c_d = 0.5$ 101
4.13 The computational performance of our approach vs. the PARIS implementation in terms of time and storage space 104

5.1 A description of the OpenML categorized datasets collected. Datasets are categorized by subject and by entity for the 203 ds sample, or by entity for the 118 ds sample. 119
5.2 The evaluation of DS-kNN for the minimum category size of 1+ with the different model types and ground-truth types. For each setting, we only show here the best performing parameters based on F1-scores. 127

5.3 The evaluation of DS-kNN for the minimum category size of 3+ with the different model types and ground-truth types. For each setting, we only show here the best performing parameters based on F1-scores. 128
5.4 The evaluation of DS-kNN for the minimum category size of 5+ with the different model types and ground-truth types. For each setting, we only show here the best performing parameters based on F1-scores. 129
5.5 The evaluation of DS-kNN for the minimum category size of 8+ with the different model types and ground-truth types. For each setting, we only show here the best performing parameters based on F1-scores. 130
5.6 The evaluation of top performant DS-kNN settings for the minimum category sizes of 1+, 3+ and 8+ with the 118 ds validation sample. 132
5.7 The evaluation of specific DS-kNN settings which met specific criteria with the 203 ds sample. We re-validate them with the 118 ds sample. 132

Abbreviations

Standardisation is the pinnacle of global human development.

BD:	*Big Data*
BI:	*Business Intelligence*
CSV:	*Comma-Separated Values*
DB:	*Database*
DL:	*Data Lake*
DM:	*Data Mining*
DS:	*Dataset*
DW:	*Data Warehousing*
ETL:	*Extract, Transform and Load*
GUI:	*Graphical User Interface*
HTML:	*Hypertext Markup Language*
k-NN:	*k-Nearest-Neighbour*
Prox:	*Proximity*

Thesis Details

Thesis Title:	Dataset Proximity Mining for Supporting Schema Matching and Data Lake Governance
Ph.D. Student:	Ayman Alserafi
Supervisors:	Prof. Alberto Abelló, Universitat Politècnica de Catalunya, BarcelonaTech, Spain (UPC supervisor) Prof. Oscar Romero, Universitat Politècnica de Catalunya, BarcelonaTech, Spain (UPC supervisor) Prof. Toon Calders, Université Libre de Bruxelles, Brussels, Belgium (ULB supervisor)

The main body of this thesis consists of the following papers:

[1] Towards information profiling: data lake content metadata management. Ayman Alserafi, Alberto Abelló, Oscar Romero, Toon Calders. International Conference on Data Mining Workshop (ICDMW) on Data Integration and Applications (2016).

[2] DS-Prox: Dataset Proximity Mining for Governing the Data Lake. Ayman Alserafi, Toon Calders, Alberto Abelló, Oscar Romero. International Conference on Similarity Search and Applications (SISAP) (2017).

[3] Keeping the Data Lake in Form: Proximity Mining for Pre-filtering Schema Matching. Ayman Alserafi, Alberto Abelló, Oscar Romero, Toon Calders. ACM Transactions on Information Systems (TOIS) 38(3): 26 (2020).

[4] Keeping the Data Lake in Form: DS-kNN Datasets Categorization Using Proximity Mining. Ayman Alserafi, Alberto Abelló, Oscar Romero, Toon Calders. International Conference on Model and Data Engineering (MEDI) (2019).

[5] Prox-Mine: a Tool for Governing the Data Lake using Proximity Mining. Ayman Alserafi, Alberto Abelló, Oscar Romero, Toon Calders. Under submission (2020).

This thesis has been submitted for assessment in partial fulfillment of the PhD degree. The thesis is based on the published or under-submission scientific papers which are listed above.

Chapter 1

Introduction

To start with, don't just share a problem, present the solution as well!

1 Motivation

Data are becoming highly abundant in large volumes, different structures, and they are flowing to the enterprise at high velocities which leads to the phenomenon of "Big Data" (BD) [1, 114]. This includes datasets created and loaded near-real-time and in big amounts.

With the huge growth in the amount of data collected, it is more common for data scientists to store raw tabular datasets from multiple sources and sensors into Data Lakes (DLs) [13, 53, 107, 113, 118, 128], which is the new generation of data repositories complementing the Data Warehouse (DW) for business intelligence (BI) and data analytics purposes [80]. DLs support the new era of data analytics where datasets are ingested in large amounts and are required to be analysed just-in-time [82]. For this purpose, they store datasets in their raw formats without any transformation or preprocessing, which allows for the concept of schema-on-read, including "fusing" different sources for analytical purposes on-the-fly during analysis requests [107].

However, it is a challenge for data wranglers [51, 69, 128] to prepare the datasets for analysis. They need to understand their structure and *commonalities* for DL governance purposes [15, 42, 80, 119]. Such repositories need to be effectively governed to gain value from them; they require the application of techniques for extracting information and knowledge to support data analysis and to prevent them from becoming an unusable *data swamp* [4] (a DL repository which is not well governed, does not maintain appropriate data quality measures, and which stores data without associated metadata, decreasing their utility [13]). This involves the organised and automated extraction of metadata describing the structure of data stored [132], which is the main focus of this thesis. The main challenge for DL governance is related to information discovery [95]: identifying related datasets that could be analysed together as well as duplicated information to avoid repeating analysis efforts.

DLs should provide a standard access interface to support all its consumers [54, 91], including analysis by non-technical-savvy users who are interested in analysing these data [13, 30, 92, 128]. Such access should support those data consumers in finding the required data in the large amounts of information stored inside the DL for analytical purposes [132]. Currently, data preprocessing, including the crucial step of information discovery, consumes 70% of time spent in data analytics projects [128], which clearly needs to be decreased. To handle this challenge, this thesis proposes an integrated framework for extracting metadata describing the datasets stored in DLs and relationships between those datasets. Thus, we propose using Data Mining (DM) techniques to effectively and efficiently extract similarity between datasets. Currently, there is a lack of such techniques for finding data patterns and similarity between datasets [2, 40, 84, 99, 105]. This is the focus of this thesis.

2 Background and State-of-the-art

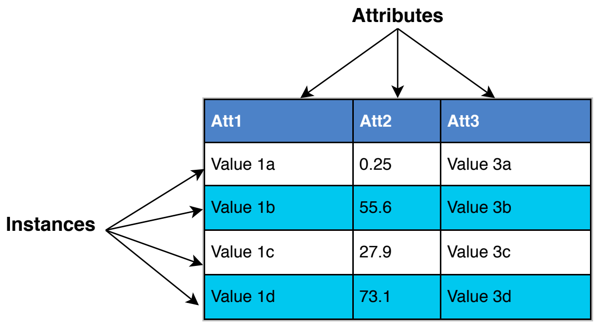
In this section, we introduce the preliminaries and the main concepts related to DLs and their governance. We describe here the state-of-the-art of metadata extraction and management for governing DLs. We present an overview of the approaches used, including the short-comings faced which we aim to solve in the thesis.

2.1 Data Lakes and Tabular Datasets

DLs are large repositories of raw data coming from multiple data sources which cover a wide-range of heterogeneous topics of interest [4, 82, 107]. We focus on DLs having datasets storing data in flat tabular formats as shown in Figure 1.1. These are organised as attributes and instances, such as comma separated values (CSV) files, hypertext markup language (HTML) web tables, spreadsheets, HDFS tables in Hadoop[1], etc. Such datasets are common in DLs today [28, 42, 47].

These datasets have *instances* describing real-world entities, where each is expressed as a set of *attributes* describing the properties of the entity. We

[1]`https://hadoop.apache.org`



Att1	Att2	Att3
Value 1a	0.25	Value 3a
Value 1b	55.6	Value 3b
Value 1c	27.9	Value 3c
Value 1d	73.1	Value 3d

Figure 1.1: A flat structured dataset consisting of tabular data organised as attributes and instances

formally define a dataset D as a set of instances $D = \{I_1, I_2, ...I_n\}$. The dataset has a set of attributes $S = \{A_1, A_2, ...A_m\}$, where each attribute A_i has a fixed type, and every instance has a value of the right type for each attribute. We distinguish between two types of attributes: continuous ***numeric attributes*** with real numbers like 'Att2' in Figure 1.1, and categorical ***nominal attributes*** with discrete values like 'Att1' and 'Att3'.

2.2 Data Lake Governance

DL governance is concerned with management of the data stored appropriately so that they are easy to find, understand, utilise and administer in a timely manner [12, 53, 106, 113, 118]. This makes the DL more accessible (searchable), data-driven, compliant to regulations, etc. It can be seen from two perspectives as seen in the classification in Figure 1.2: (i) Policy administration and (ii) Data management. The first is about enacting procedures, managing people and responsibilities, implementing regulations, etc., while the latter is about the technical implementation of protocols and technologies that help manage the data stored as an asset for supporting users in analysing them. Figure 1.2 shows some of the main tasks required for each type of DL governance (this is only a partial list of the most important tasks mainly based on [106] and our adapted / expanded classification from it).

We describe the different tasks of DL governance as follows:

i. Policy Administration

- **Data ownership and access rights**: the policies of data stewardship related to the procedures and regulations for data sources allowed to be ingested in the DL, the persons responsible for managing specific datasets and their metadata descriptions, and polices regarding how the data should be used (read, write and delete access

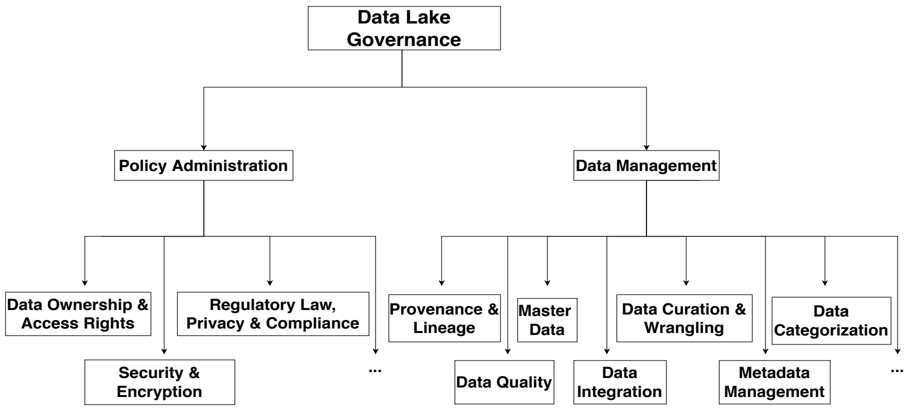


Figure 1.2: DL governance classification and tasks

rights) and who is allowed to access specific data objects.

- **Security and encryption**: the rules put in place to protect the security of the data stored in the DL (e.g., anti-malware and denial-of-service-attack protection), and how the data are encrypted for specific users (management of encryption keys).
- **Regulatory law, privacy and compliance**: the rules and policies for compliance with the regulatory law and for protecting the privacy of data, e.g., with data protection laws like GDPR[2].

ii. Data Management

- **Provenance and lineage**: the collection and management of information about the processes implemented for data collection (data sources), data manipulation, data transformation rules and the sequence of such processes in a traceble pipeline which led to the final data stored in the datasets of the DL (for more details and examples, we refer the reader to [37, 54, 65]).
- **Data quality**: the techniques and tools used to guarantee certain levels of data quality and data integrity according to the rules defined. This includes referential integrity constraints, denial constraints, format and patterns compliance with master data, etc. (see [7, 38, 95]).
- **Master data**: this includes tools and techniques for unifying data records into a single consolidated master data collection which consistently and uniformly describe entities stored in them. This leads to a centralised view of core data (e.g., unified lookup tables

[2]`https://gdpr.eu`

for entities and their identifiers), and is commonly called Master Data Management [92].

- **Data integration**: this involves combining and mapping different datasets together so that they can be used to answer analytical queries. It tackles overall, high-level and generic data requirements. For details see [14, 67, 93]. Other tasks include entity resolution and deduplication [112, 56], and extract, transform and load (ETL) for creating a consolidated global schema or views over the data [63, 68, 97, 129].
- **Data curation and wrangling**: this includes the processes for preparing data for BI and analytics, for structuring the data into machine processable formats and schemata, and also using the metadata describing datasets to identify commonalities between them, commonly called *schema matching* (see Section 3.1). These are commonly tasks targeting a specific analytical goal. The reader can find more details in [125, 128].
- **Metadata Management**: this is a transversal task serving all the other governance tasks. The datasets inside the DL need to be described to include information about what they store (i.e., with semantic definitions and data profiling statistics), how they are stored (e.g., data formats and schema definitions), where are they stored, how they are related to one another, etc. This is all considered metadata, and the management of them in a centralised repository is the main task of concern here. For a detailed discussion of the types of metadata needed for DLs, the reader can check [25, 53, 54, 58, 108, 113, 119, 118, 122, 132]. As this is a core topic for this thesis, we further describe specific metadata which need to be collected and managed later in this section.
- **Data categorization**: we introduce this data management concept for DLs as the task involving the identification of information overlaps between datasets and data catalogues to understand what data are owned in the DL. Such catalogues describe how the datasets are classified into specific subject areas (topics). This is similar to data classification [106] as it also tags datasets with a category from a catalogue, however, for the purpose of this thesis, it is a data-driven and automatic task rather than a manual data-tagging task. This is also closely related to text-document categorization [19, 55], but it categorizes tabular datasets as described in Section 2.1 rather than free-text documents. We describe this further in Sections 2.2 and 5.2. Also see [16] and Chapter 5 for details.

Data categorization is the data-driven task of automatically classifying datasets into pre-defined topics-of-interest (i.e., business domains) based on the overlap between data stored in them and the common information they store (i.e., dataset similarity).

In this thesis, we focus on the data management perspective and when we refer to our proposed approaches for **DL governance we mean metadata management for data categorization** in specific, which we describe further in this section.

Metadata Management for Data Categorization

There is currently a need for DL governance by collecting and managing metadata about ingested datasets [14, 94, 95, 107, 119] to prevent the DL from becoming a data swamp. In addition, proper DL governance should provide an interactive data exploration interface that supports analysing relationships between datasets.

It is important to govern the data ingested into the DL and to be able to describe the data with metadata [95, 119]. The current tools supporting metadata collection in a DL environment (examples given below) usually include automatic and manual techniques for generating metadata. As can be seen in Table 1.1, the tools are either automatically generating metadata about data provenance or manually generating some data content descriptions using user tagging. Examples of such tools which are able to integrate with Hadoop and which were available for us to survey include Apache Atlas[3] (open-source tool), Teradata Loom[4] (free non-commercial use), DataRPM[5] (proprietary tool), Waterline[6] (proprietary tool), Zaloni[7] (proprietary tool), and Podium Data[8] (proprietary tool). Other examples of tools which were not available at hand but which claim to support some metadata management tasks for DLs include: Collibra[9] (proprietary tool), Palantir[10] (proprietary tool), and PoolParty Suite[11] (proprietary tool).

Apache Atlas is mainly for general metadata management and it does not use any DM techniques. It handles mainly the data provenance metadata for

[3]http://atlas.incubator.apache.org
[4]http://loom.teradata.com
[5]http://datarpm.com
[6]http://www.waterlinedata.com/data-catalog-details
[7]https://resources.zaloni.com/webinars/zaloni-bedrock-and-mica-20-minute-demonstration
[8]https://www.podiumdata.com/product/podium-platform
[9]https://www.collibra.com/data-governance
[10]https://www.palantir.com
[11]https://www.poolparty.biz/

Technique	Metadata Type	Description
Automatic	File-level metadata (e.g. filename, creation date, source)	The automatic processing of files ingested in the DL to collect provenance metadata like file source, data creation and update date and time, data ownership, etc.
Manual	User tagging of data	This includes the user tagging the ingested data with business taxonomies. The user tagging is executed using conditional rules defined by the user depending on the type of data source, e.g., tag all data files loaded from the financial module of the enterprise system as “Financial". Another possibility is the user visually exploring the files from each new data source and individually tagging the instances seen by the user.

Table 1.1: Current types of metadata tools for data lakes

the DL rather than data categorization. The Waterline data management tool complements Apache Atlas by adding capabilities for simple data profiling of datasets ingested in the DL (e.g., the top values per field, number of unique values, etc.), which they call “Data Fingerprinting”. They only implement simple data profiling techniques which does not include DM or cross-datasets profiling. Teradata Loom is mainly for data lineage and provenance management of files and data manipulations / ingestion job tasks. It does some simple statistical calculations on the input data using an automatic technology for data profiling called “Activescan”[12], but only at the column level. Finally, DataRPM claims to have an automatic data curation platform (i.e., for data

[12]http://blogs.teradata.com/data-points/overview-of-teradata-loom-technology

discovery and data preparation for analytics, whose details about data curation can be found in [128]), but with little description, no case-study and no published research papers. After investigating this tool it was found that it is more focused on natural language processing, similar to IBM Watson[13]. Although they claim that they can automatically curate data sources in the DL using machine learning[14], there is little evidence in online material or published research about their techniques. This might be because the company developing the tool is a relatively new start-up.

Other tools also include Apache Falcon[15] (designed for the Hortonworks distribution of Hadoop) and the Cloudera Navigator[16] (designed for the Cloudera Hadoop distribution). Both tools handle two of the most popular distributions of Hadoop. Concerning the support provided for handling metadata and data governance, those tools are mainly focused on DL operations metadata like data provenance and lineage, in addition to manually user-generated data catalogues. Both tools are targeting in newer versions the gradual addition of more exploratory content analytics and metadata, however, at the current time such support is very primitive and does not include advanced techniques like schema matching [24], ontology alignment [14, 126] or data mining [99].

The Waterline Data for Hadoop[17] seems to be the closest solution available to our envisioned metadata-based DL governance solution in this thesis. The tool supports exploratory analysis and profiling of data ingested in the DL. However, it is currently lacking in the following aspects described in the research literature:

- Data and schema profiling [98, 115, 123]
- Schema and ontology matching [14, 24, 126]
- Machine learning and DM techniques for finding data patterns and similarity [2, 40, 84, 99, 105]
- Ontology integration (semantic metadata) [2, 3, 94, 132]

Tools implementing some of the above techniques for automatic profiling of datasets and metadata collection exist in the research literature. This includes semantic RDF profiling tools [2], ontology mapping tools like ODIN [94], platforms for BD analysis like Metanome and Stratosphere data profiler [8, 102], metadata visualization tools for BD like [46, 57, 70], and data provenance metadata capture and management like [65]. Other research investigated specific research problems for metadata management like: automatic

[13]http://www.ibm.com/smarterplanet/us/en/ibmwatson/what-is-watson.html
[14]http://www.kdnuggets.com/2014/06/data-lakes-vs-data-warehouses.html
[15]https://falcon.apache.org
[16]https://www.cloudera.com/products/cloudera-navigator.html
[17]http://www.waterlinedata.com/prod

key and dependency detection [98], schema matching and evolution [120], entity resolution and matching [120], schema mining from semi-structured data [10, 23, 105], ontology mining [3], etc. Those independent techniques still need to be directed and integrated in a coherent framework for profiling data in the DL and managing metadata about the content of the files [128]. We mainly propose in this thesis an automated metadata management framework in Section 5.1 that augments many of those techniques in a novel approach we call *proximity mining*, including: data and schema profiling, schema matching (see Section 3.1), and machine learning & DM techniques for similarity computations between datasets (see Section 3.3).

Most of the research literature [98] also demonstrates the shortcomings and gaps of current data profiling techniques for automatic metadata management over the DL. The current shortcomings of the tools in research and industry indicate a lack of automatic ***data categorization*** capabilities which can be summarised in the list below:

1. Efficiently collect cross-dataset metadata like relationships between them using automatic techniques.
2. Automatically categorize datasets into pre-defined topics of interest.
3. Provide better presentation and querying interfaces of metadata discovered from the DL which can make the data more accessible for BI.

As can be seen from the above list and Table 1.1, there is a need to have an automatic process to generate the tagging (commonly called *annotation* [5]) of data content ingested in the DL using more efficient techniques compared to the manual visual scanning of the files. This includes automatic techniques for data profiling and analysis of similarity of data content and relationships between datasets ingested in the DL, which is the main purpose of this thesis.

It is currently a challenge to create an analytical environment where there is an integrated, subject-oriented view of the DL which is similar to that in the data warehouse [64]. This poses the need for annotation of the datasets to facilitate finding the relationships between them [30], which includes collection of metadata about the informational content of the datasets in the DL [58].

We propose to govern the DL by means of cross-dataset metadata collection and management so it does not become a disconnected group of data silos where datasets can not be used together in meaningful analysis. This is done by automating the data cataloguing tasks which include finding related datasets and categorizing datasets into topical domains. Thus, we handle DL governance from metadata management and dataset categorization perspectives, where we aim to support the users in understanding what datasets they own, how they are related to one another, and to support automatic dataset categorization. This should ultimately support data analytics over these data.

We aim to automatically and efficiently collect two types of metadata: content metadata based on data profiling statistics and cross-dataset relationships using DM (see Section 3.3) and schema matching (see Section 3.1). Traditional data profiling and schema extraction involves analysing raw data for extracting metadata about the structural patterns and statistical distribution of datasets [98]. There is currently a need for higher-level profiling, which involves the analysis of data and schemas [58, 116] using DM techniques [23, 87, 105]. This contributes to the computation of cross-dataset relationships to support information discovery and interpretation, in addition to automatic topical categorization of datasets, which are specific research challenges of DL governance we tackle.

Metadata querying and visualization

Metadata collected about the datasets in the DL should be presented to the users using querying and visualisation interfaces. Some examples of such queryable metadata Graphical User Interface (GUI) include: [14] provides a query engine using SPARQL over metadata stored in an RDF repository, [34] provides a dataset search query engine over DLs, [54] provides keyword search over the dataset names and storage location paths, and [33] provides a query language to query the indexes of attributes found in datasets to retrieve relevant ones. The goal is to retrieve datasets related to a specific subject-area and all their datasets found in the enormous amount of data. We consider that related datasets are those matched and found using the indexed metadata stored in the repository. This should support in describing and extracting similar data elements from data repositories, in order to reuse these data, mash-up the data for more plausible business problems and questions, and to be able to handle this data discovery process [125].

We aim to provide solutions to the current shortcomings of such metadata discovery approaches for DLs. This includes the need for generic dataset metadata handling which does not assume specific subject-areas (business domains) or availability of pre-defined metadata like mapping to a global ontology similar to the work in [14, 79, 96], context TF-IDF similarity of words around a web table found on the same webpage like [133], or specific pre-defined rules and patterns for data stored in the attributes like [134].

3 Techniques and Challenges

In order to support the DL governance tasks of automatic metadata collection & management and dataset categorization, we utilise different techniques including schema matching, dataset similarity computation and supervised machine learning. We explain in this section those techniques, their current short-comings and their challenges which we tackle.

3.1 Schema Matching

The datasets in the DL usually cover a wide range of heterogeneous topics making it difficult for the data scientist to identify overlapping attributes. Data wranglers must be able to find those datasets which have related data to be used together in data analysis tasks [82, 89], commonly referred to as *schema matching* [24]. This supports in the automatic collection of metadata about cross-dataset relationships.

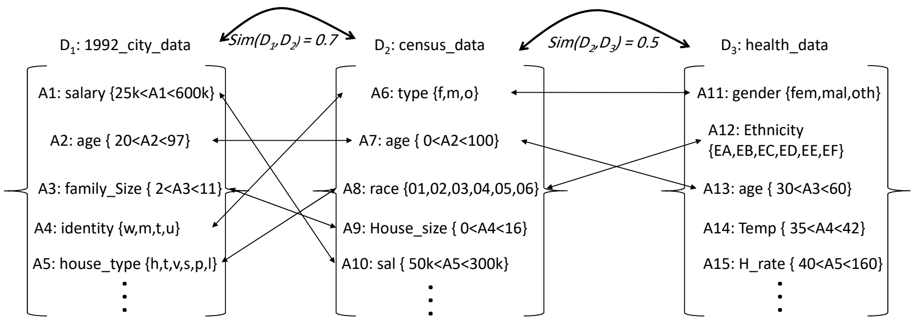


Figure 1.3: An example of schema matching and dataset similarity computation

We show an example for schema matching in Figure 1.3 with three datasets and their attributes (each numbered starting with 'A'), namely "1992_*city_data*", "*census_data*" and "*health_data*", where each dataset has 5 attributes. Attributes can be numerical like 'A1', 'A7', etc. which have the ranges shown. On the other hand, the attributes can be nominal like 'A4', 'A12', etc. where we show the distinct values for each attribute. The goal of schema matching is to be able to find attributes between different schemata (datasets) which store *similar* data. This is shown by the attributes connected by arrows in the figure. Those attributes have similar values (and possibly similar names as well), which should be automatically found using schema matching techniques. Further, we label a similarity score we call '*Sim*' between dataset pairs, which is a real number in the range of $[0, 1]$ describing the overall dataset pair similarity (this is explained in Section 3.2).

Schema matching usually requires Cartesian product comparisons of attributes, entities and their values for calculating their similarity [73], with the aim of finding connection strengths between similar concepts from different pairs of datasets [27, 35, 83, 104]. The large-scale application of such a task to BD repositories is referred to as *holistic schema matching* [18, 89, 104, 110, 133], where the goal is to match multiple datasets together considering them all in the matching task. With thousands of datasets and millions of attributes stored, there is a need for efficient similar attributes search, filtering schema matching comparison tasks, commonly called early-pruning [24], as simple brute-force similarity search of all pairs of attributes from different datasets becomes infeasible [54]. Alternatively, comparisons can be computed faster using parallel computing techniques like the MapReduce framework [133].

It is a challenge for data wranglers using the DL to efficiently process the datasets to detect their common features, as schema matching tasks are generally expensive (involving huge amounts of string comparisons and mathematical calculations) [14, 24, 27, 60, 71]. In this thesis, we propose novel techniques to reduce those comparisons using *pre-filtering* techniques that reduce the number of comparisons. With the rise of DLs, previous research like [43] and [24] recommended using early-pruning steps to facilitate the task using *schema matching pre-filtering*. Here, only datasets detected to be of relevant *similarity* are recommended for further fine-grained schema matching tasks, and dissimilar ones should be filtered out from further comparisons. Such previous research like [24, 43] mentioned the need for early-pruning steps but did not provide detailed solutions for this challenge. We explore this with extensive details in Chapter 4.

Holistic schema matching techniques [110] seek to detect relationships between different schemata stored in a data repository. This can include relationships between instances, which is called instance-based schema matching and which is closely related to entity-resolution [24, 133], and also relationships between attributes (fields storing common information in structured data), which is commonly called attribute-based schema matching. We focus in this thesis on the latter case within the environment of DLs. The goal of using those techniques in the DL is to detect relationships between the attributes inside different datasets. The output patterns from holistic schema matching include multiple observations like: (i) Highly-similar and duplicate pairs of datasets (having similar schemata and data), and (ii) Outlier datasets which have no similarity with any other dataset in the DL (i.e., no similar attributes in the DL). The main challenges concerning this include the efficiency, scalability and handling the variety of topics found.

Current schema matching techniques are mainly based on Jaccard similarity of exact value matches and overlaps between attributes from datasets [33, 96, 101]. Those techniques fall short when values are coded differently between different datasets and with the unavailability of extra semantic mappings to ontologies or word embeddings like [96]. For example, in Figure 1.3 we can see attribute "identity" ('$A4$') with the values "w", "m", "t" and "u" v.s. attribute "type" ('$A6$') with the values "f", "m" and "o". Both contain similar information but with different representations and attribute names which classical schema matching can not detect. This is also the same issue for ('$A8$') and ('$A12$') with different attribute names and value coding. Other challenges for state-of-the-art schema matching include normalised v.s. non-normalised numeric attributes, encrypted values v.s. non-encrypted values, etc. In this thesis, **we develop approximate matching techniques based on statistical**

metadata, which do not necessarily require exact value overlaps between datasets.

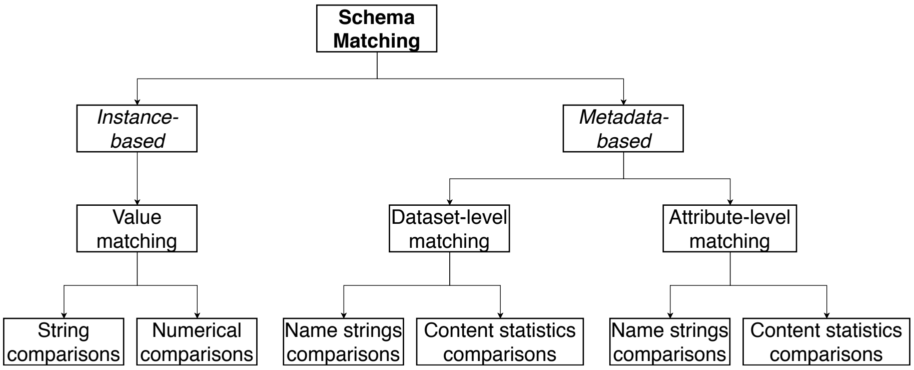


Figure 1.4: Classification of schema matching techniques

Leading from the above discussion, for this thesis, we classify schema matching as shown in Figure 1.4. Here, we can see on the left side the classical schema matching techniques which seek to match values from instances in the datasets. This is done using string comparisons of the values (and names of attributes) or numerical comparisons for numerical attributes. This will have good result with matching attributes like ('$A1$') and ('$A10$') from Figure 1.3 as they have similar overlapping numerical values and similar attribute names.

On the other hand, we propose a new type of schema matching which we call metadata-based schema matching.

Metadata-based schema matching pre-filtering techniques extract metadata collected about the overall profiles of datasets or statistics about attributes for alleviating schema matching comparison tasks. The metadata is used to pre-filter dissimilar pairs of datasets to reduce the number of comparisons from the Cartesian product of value strings matching from instances to a lower number of comparisons over the overall attributes' and datasets' metadata.

For this purpose, we collect descriptive statistics about the attributes and datasets to use them in the comparison task, which includes comparison of name strings (e.g., using the edit distance [86]) and comparison of the statistics collected about the content of the attributes (e.g., we can collect the number of distinct values for attributes and compare this when matching attribute pairs). We can collect such statistics at two levels: the dataset-level (e.g., the average number of distinct values from ***all*** attributes in the dataset) or the attribute-level (e.g., the distinct number of values for each independent attribute, then

aggregating this to the dataset-level by averaging all the matching output for each independent attribute). For example, in Figure 1.3, dataset D_1 has an average of 4.5 distinct values for nominal attributes (3 for 'A4' and 6 for 'A5') while dataset D_2 has an average of 3 distinct values for nominal attributes (2 for 'A6' and 4 for 'A9').

Attribute-level comparisons are obviously at a more detailed level of granularity than the dataset-level. For example, in Figure 1.3, if we consider the distinct values for each attribute to find the similarity between dataset D_2 and D_3 , we will find that they both have matching nominal attributes: 'A6' and 'A11' with 3 distinct values each, and 'A8' with 'A12' with 6 distinct values each. Therefore, if we consider individual attributes they will have 100% matching nominal attributes, considering only 'number of distinct values'. Such metadata-based schema matching is further explained in Chapters 3 and 4, where the former investigates dataset-level matching and the latter investigates attribute-level matching.

Locality Sensitive Hashing

Locality sensitive hashing (LSH) [31, 44] is a family of techniques for grouping together objects of high probability of similarity[18]. Objects assigned to the same block are further compared using more expensive computational techniques [117], like schema matching tasks involving string comparison of values and statistical computations of similarity from data profiles of attributes.

The challenge is to find an adequate LSH algorithm with a relevant similarity function and a list of features to use to compute an approximation of the similarity between objects inside the datasets [28, 33] to support schema matching. For example, in [44] they propose LSH based on the MinHash algorithm for entity resolution in ontologies using term frequencies. We apply similar LSH techniques in this thesis when implementing instance-based value matching in Chapter 2.

3.2 Dataset Similarity Computation

As low-level instance-based schema matching is computationally expensive, there is a need for a filtering strategy that computes similarities of datasets in a cheaper way so that schema matching is conducted *only* in similar cases. This is done by applying a pre-filtering task before schema matching (i.e., early-pruning), where there needs to be dataset similarity computation techniques that can assign an estimation of the overall similarity score between datasets. This can be seen in Figure 1.3, where we give a similarity score between datasets using the connecting arrows between the dataset names. For example,

[18]Contrary to the classical hashing goals for database partitioning, where the goal is to put similar objects into different groups.

datasets D_1 and D_2 have an overall similarity of 0.7 (on a range of $[0, 1]$, i.e., 70% similarity) considering the attributes they store and the overlaps between them.

Most similarity comptuation algorithms utilise attribute names string comparisons when computing dataset similarity [28, 33, 35, 36, 59, 79, 88, 100, 104, 117, 133]. This becomes problematic in real-world scenarios when attributes are not properly named or are named using generically generated codes [27, 96].

The current state-of-the-art mainly includes expensive computations that involve string-matching of values found in attributes. The shortcoming of such techniques is that they can only match exact values of strings and can therefore not produce *approximate similarity computations* which compare the overall similarity of attributes even if the values are coded differently. For example, [28, 33, 96, 101, 117, 133] aim to index and find similar attributes based on the exact values they store, mainly using Jaccard similarity computations like in [28, 96, 101, 117]. Other research like [60, 59, 96] use semantic based dictionary search of synonyms over values to improve the matching process. In [59], they also use extra query log metadata to find related attributes commonly used together in search queries. Extra schema metadata like common primary keys between datasets can also be used as an indicator for dataset similarity [54].

Other techniques like [134] assume that datasets have attributes storing values according to a specific format or template (sequence of codes) which are pre-defined by the data analyst. Values are defined according to convertible values using conversion rules (e.g., measurements in inches and feet). Those techniques are restricted to only matching those attributes which store those specific values defined by the format rules, therefore limiting the generalisability of their application with different domains and heterogeneous topics of interest. We tackle those challenges and shortcomings in our proposed proximity mining approach introduced in the next section and Section 5.1.

3.3 Similarity Models Learning

We propose a novel approach (see Section 5.1) that learns similarity models which are capable of detecting similar and dissimilar datasets to support the approximation of dataset similarity computation and the early-pruning task of holistic schema matching. This is done using *supervised machine learning*.

Supervised machine learning is a group of classical techniques used in artificial intelligence and data mining for automatically learning models from data examples to support future estimation, classification or prediction of *similar cases*. This has been discussed extensively in textbooks like [127] and [32]. The main idea is to be able to make a computer (machine) learn from training examples to classify, predict, or estimate a dependent variable (the output, also called *target* variable) for new examples (commonly called unseen *instances*) based on *similarity* to previous already seen instances with known classification or values for the dependent variable (commonly called *training* instances). This similarity needs to be measured according to a set of *features* that describe information about the instances. In contrast with supervised machine learning, there is also unsupervised machine learning. In such learning problems, the instances do not have a predefined classification or value for a dependent variable, rather, one common goal is to use the features describing such instances to segment them into groups of highly similar instances (commonly called clustering) so we can automatically detect if there is meaningful grouping which can classify the instances into classes (i.e., the goal is to define the classification classes for instances which we do not already know in advance). This type of unsupervised learning does not require any training data (i.e., no pre-advanced annotated examples needed).

Supervised machine learning algorithms create *models* which are capable of getting an input set of features about unseen instances, to apply a group of functions that check the similarity of those instances against previously seen instances based on those features, and to output a classification (if the output is binary or categorical) or estimation (if the output is a continuous number) for the dependent variable. The model created could be a decision tree, a probabilistic statistical model, a mathematical regression, etc.

We give an example of a supervised learning model in Figure 1.5. This model should be the resulting output from running a learning algorithm with some training instances. The figure shows the outcome consisting of a decision tree where the parent (top) node has an input of a previously unseen instance. In this toy example, we use the topic of this thesis, where we consider input descriptions for a pair of datasets, which we call D_y and D_z, in order to decide if they are related to each other or not. The input features could include a percentage of the difference between the attribute names and attribute values from both datasets. Each node in the tree is a decision which takes as input one of those features and decides which branch of the tree to take based on the condition in the node. The leaf nodes (lowest children in the tree visualised as shaded boxes) give the output classification by the decision tree. In this example, the output is binary: '0' for not related and '1' for related datasets. For example, if more than 80% of the attributes have similar names then they are considered to be related (as they are $< 20\%$ different), while if they have more than 50% difference in the attribute names and more than 25% in their values, then they are considered not to be related. Decision trees are one type of supervised models for classification, but other types exist as discussed earlier (e.g., naive Bayes probabilistic approaches, lazy learners[19] like k-nearest neighbour (k-NN), logistic regression for binary classification,

[19]They are called lazy learners because they do not create a model in advance, rather, they check unseen instances against all previously stored examples of classified instances in a database in order to recommend a classification based on the most similar instances found.

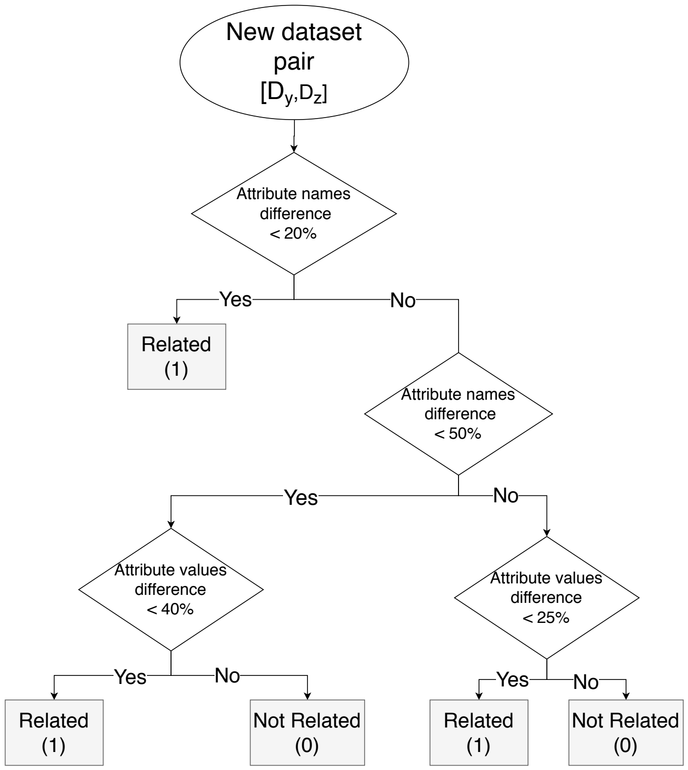

New dataset pair [D_y,D_z]
Attribute names difference < 20%
Yes
No
Related (1)
Attribute names difference < 50%
Yes
No
Attribute values difference < 40%
Attribute values difference < 25%
Yes
No
Related (1)
Not Related (0)
Yes
No
Related (1)
Not Related (0)

Figure 1.5: Example of a decision tree model for classification of related dataset pairs

linear regression for real-value estimation, etc.).

To create a better performing machine learning model capable of more accurately classifying new instances, it is common to create a grouping of independent models from multiple algorithms to create a single model combining them to give a single output. The combination can be a weighted aggregation or combination of the output from the independent models. This is illustrated in Figure 1.6, where we give an example of a decision tree ensemble consisting of multiple decision trees (from 1 to *n*). The output is combined using a weighted function (with weights 'W') to give a single classification as a final output. Common ensemble algorithms include boosting techniques like AdaBoost [49], decision tree ensembles like RandomForest [29], etc.

One of the common pitfalls of supervised machine learning is the dilemma of *overfitting*, where there is a trade-off between highly accurate models and overfitting of those models. Overfitting has a negative connotation as it means that the models are highly matching the exact cases in the training instances, to the extent that even the *noise* in such instances (i.e., instances which are misclassified, erroneous, exceptional, etc.) are incorporated in the models.

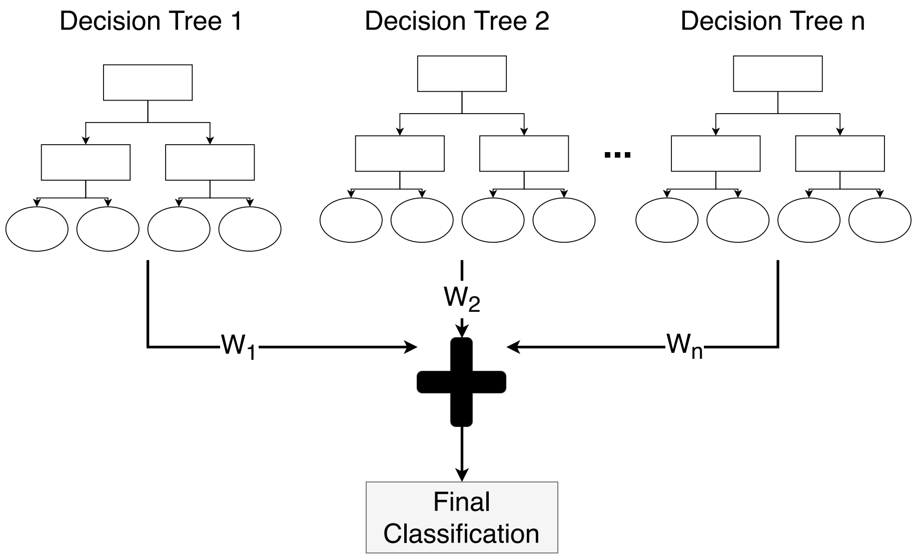


Figure 1.6: An ensemble of decision trees for classification

This makes such overfitted models not generalisable to other instances from another sample not seen in the *training*, leading to lower performance and accuracy when applied to them. This problem is commonly avoided by: doing cross-validation evaluation of models and optimising the performance for all runs of the validation process, removing noise from the input data (for example by data cleansing), removing irrelevant or correlated features used in the training of the models, selecting more relevant features for training, using a bigger data sample, using a supervised learning algorithm which intrinsically leads to less bias and variance like RandomForest ensemble learners, etc.

We aim to develop a similarity technique for dataset pairs based on a set of meta-features we collect to describe them, and then applying supervised machine learning algorithms (including ensemble techniques) to create accurate models which can support the similarity computation task. The main challenge is finding adequate features and a generalisable approach to handle such similarity computations accurately and efficiently. We use supervised machine learning techniques to compute dataset similarity in Chapters 3 and 4.

4 Thesis Objectives and Research Questions

The goal of this thesis is to implement an effective framework that can fill the research gaps presented in the previous discussion. This includes the following objectives:

- **Objective A:** Formulate techniques that can efficiently and effectively compute approximation of dataset similarity.
- **Objective B:** Develop a framework based on approximate dataset similarity techniques that can support DL governance, mainly cross-dataset relationships metadata management, holistic schema matching pre-filtering and automatic dataset categorization.
- **Objective C:** Validate our proposed framework in a real-world DL environment.

For Objective A, we formulate techniques which are able to approximate overall dataset similarity based on metadata we collect from them, including attribute profiles. In order to compute dataset similarity, we utilise supervised machine learning algorithms (e.g., RandomForest ensemble learners [29]). Such algorithms are able to learn from training examples to distinguish between different classification categories, where in this thesis we aim to differentiate between similar and dissimilar dataset pairs based on the overlap between their data. We propose an approach that uses those techniques to effectively and efficiently compute such similarity using the metadata extracted.

Then, for Objective B, we exploit the output dataset similarity computations in specific DL governance tasks. This includes supporting the early-pruning goal for holistic schema matching, in addition to supporting dataset topical classification using automatic categorization techniques. The thesis defines a framework for extracting the schema metadata from flat tabular datasets using classical techniques, in addition to experimentation with novel techniques that can successfully tackle extraction of cross-datasets relationships, which include computation of attribute similarity between different datasets.

Finally, for Objective C, we validate our proposed framework and its utilised techniques on a real-world DL called OpenML[20] in our experiments [131], which has a large amount of datasets intended for analytics from different subject areas. OpenML is a web-based data repository that allows to upload different datasets used in DM experiments. OpenML stores datasets in the ARFF tabular format which consists of diverse raw data loaded without any specific integration schema. This allows us to evaluate our approach in a real-life setting where datasets come from heterogeneous domains.

The above objectives can be translated into specific research questions as follows:

- **Dataset Similarity Computation**
  - **Research Question RQ1:** How can we use instance-based schema matching techniques for computing dataset similarity?

[20] https://www.openml.org

  - **Research Question RQ2:** How can we use schema and data profiles in computing proximity between structured datasets using supervised machine learning techniques?

- **Metadata Exploitation for Governing DLs**

  - **Research Question RQ3:** How can we use proximity scores between datasets to pre-filter schema matching comparisons for early pruning purposes?
  - **Research Question RQ4:** How can we use proximity scores between datasets to automatically and correctly categorize datasets into pre-existing categories of interest?
  - **Research Question RQ5:** How can the content metadata be used to browse the DL and to support the users in finding relevant datasets with overlapping data?

5 Thesis Overview

In this section we present an overview of the thesis, including the proposed proximity mining framework, the DL categorization challenge and how we tackle it, and finally the metadata query interface.

5.1 Proximity Mining Framework

We formulate a framework that caters for the current shortcoming of a formalized metadata management process in the DL, with the goal of ingesting flat tabular datasets, extracting metadata describing them, using the metadata to find relationships, and finally categorising them in order to support governance. This is visualised in Figure 1.7, where we see at the bottom a set of datasets. First, we ingest them into the DL, parsing them to check that they have no errors and extract their schema, which includes description of the attributes found, their names and data types. All metadata extracted is stored in a database (stored as “schema metadata”), where each dataset is annotated with its corresponding schema description.

In the data digestion stage, we profile the datasets to extract distributions and statistics about their attributes, which are stored as “dataset content metadata”. Then, we apply the proposed *proximity mining* techniques to extract relationships between different datasets, including similar schema (and duplicate schema with very high similarity), and we store this as cross-schema relationships.

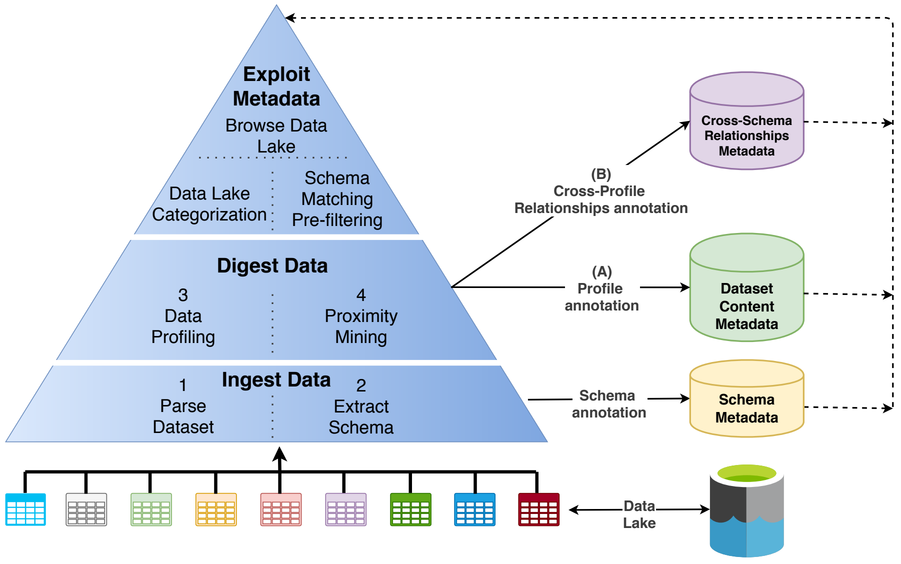


Figure 1.7: Overview of the proposed proxmity mining framework

> We define ***proximity mining*** as a data mining process for the extraction of cross-dataset similarity using metadata-based supervised machine learning models.

Our proposed framework integrates DM techniques in the DL for enabling analytical discovery of cross-dataset content relationships. We introduce in Chapters 3 and 4 a novel supervised machine learning technique for computing cross-dataset similarity, which makes the basis of proximity mining. We mainly adapt the classical techniques of DM to the new environment of the DL and BD. This includes application of DM to discover similar dataset profiles and related attributes between them.

Finally, the extracted metadata are exploited in the top-most stage, where the user (commonly a data wrangler [51, 69, 128]) is able to browse the DL for finding relevant datasets with overlapping data. In addition, they are able to categorize the datasets ingested based on the most similar ones already existing, and applying pre-filtering steps to prune unnecessary comparisons of dataset pairs before running schema matching techniques. We use DM techniques for the pre-filtering task in Chapter 4 and the automatic categorization of datasets in Chapter 5.

DM techniques (especially, supervised machine learning) are utilised in the thesis to extract useful metadata about the content of the datasets and metadata about the relationships between them. This includes data profiles and relevant information for the data scientist. Therefore, this thesis uses the contributions of the DM field and applies it in the data management context to achieve our thesis objectives. Examples of such application of DM include:

- Using supervised machine learning DM techniques to compute proximities between datasets and to detect patterns in their metadata for the schema matching process.
- Using nearest-neighbour techniques with the metadata to automatically find topical categories of datasets in the DL.
- Finding outliers which constitute abnormal data content which is not related to other datasets in the DL.

We model our approach as a 3-stages framework where each higher stage does less data analysis and gives more concise (summarised) information. The lowest *ingestion* stage has to scan the complete raw dataset files to check their correctness and extract their schema, the *digestion* stage only has to scan attributes to extract data profiles and proximity metrics, and finally the highest stage of *exploitation* only scans metadata without going through the raw data inside each dataset.

The proposed framework can be applied in the context of BI inside large business organizations with huge amounts of data or in the context of government administrations requiring a coherent framework of DL governance. Our framework also helps the data consumers understand the information owned in the DL in a quick and easy fashion. As a result, organizations can discover new analytical opportunities for BI and improve the time-to-value of analysing the DL.

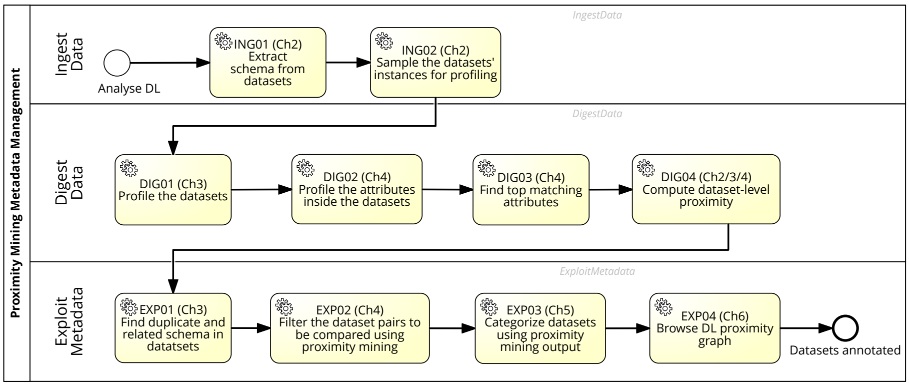

Proximity Mining Metadata Management
Ingest Data
IngestData
Analyse DL
ING01 (Ch2) Extract schema from datasets
ING02 (Ch2) Sample the datasets' instances for profiling
Digest Data
DigestData
DIG01 (Ch3) Profile the datasets
DIG02 (Ch4) Profile the attributes inside the datasets
DIG03 (Ch4) Find top matching attributes
DIG04 (Ch2/3/4) Compute dataset-level proximity
Exploit Metadata
ExploitMetadata
EXP01 (Ch3) Find duplicate and related schema in datatsets
EXP02 (Ch4) Filter the dataset pairs to be compared using proximity mining
EXP03 (Ch5) Categorize datasets using proximity mining output
EXP04 (Ch6) Browse DL proximity graph
Datasets annotated

Figure 1.8: The proximity mining metadata management process

In order to implement the proposed proximity mining framework in Figure 1.7, we instantiate it using prototypical implementations which use the techniques described earlier in this chapter (Section 3). This is visualised in Figure 1.8 as a BPMN process where each activity starts with an identifier for the activity (e.g., 'ING01'), followed by the chapter number(s) where the component is described in parentheses, and a description. The process is organised into the three stages from Figure 1.7. In ingestion, the schema is extracted in ING01 and a random sample of the instances is collected in ING02 for passing it to the next level of digesting the data.

Digestion starts with the dataset content metadata extraction using data profiling techniques. The datasets are profiled in DIG01 based on the sampled instances from the previous activity. In DIG01, we compute high-level summary statistics that describe the whole dataset. In DIG02, we compute finer-grained statistics about each individual attribute inside the datasets.

Then, we compute the cross-schema relationships metadata by matching attributes between different datasets, where their data profiles (and their attributes) are compared in DIG03 in order to find related attributes with similar data profiles and mapping each attribute to its top matching attribute from the other datasets. The output from this activity is fed to the proximity mining models in DIG04 in order to compute overall dataset-level similarity scores.

Once the datasets are profiled and cross-schema relationships are computed, the metadata generated are stored in a repository and are utilised for different tasks and applications. In EXP01, dataset pairs with high proximity scores are marked as related schema if they exceed a specific similarity threshold, or duplicated schema if the dataset pair is highly similar. Based on the output from EXP01, we aim in activity EXP02 to support the early-pruning task for holistic schema matching by pre-filtering dissimilar dataset pairs with a low similarity score from any further comparisons. We also aim in EXP03 to categorize datasets with unknown topics into a recommended topic grouping based on the most similar datasets found in the DL. Finally, in EXP04, all the metadata collected are provided to the DL user in order to browse the datasets in the DL by topics or by finding dataset pairs with highly similar schemata (possible overlapping data). This can be presented as a proximity graph where nodes represent datasets and edges represent the proximity scores between those datasets. We describe how we present the proximity graph and the output of exploitation in the following sub-sections.

5.2 DL Categorization

One of the tasks of DL Governance is effective data categorization. This means finding the subject areas in the datasets and classifying them accordingly. This supports data discovery which includes finding related datasets [10, 95] that can be analysed together, duplicate datasets for data quality checks [56], and outlier datasets [74].

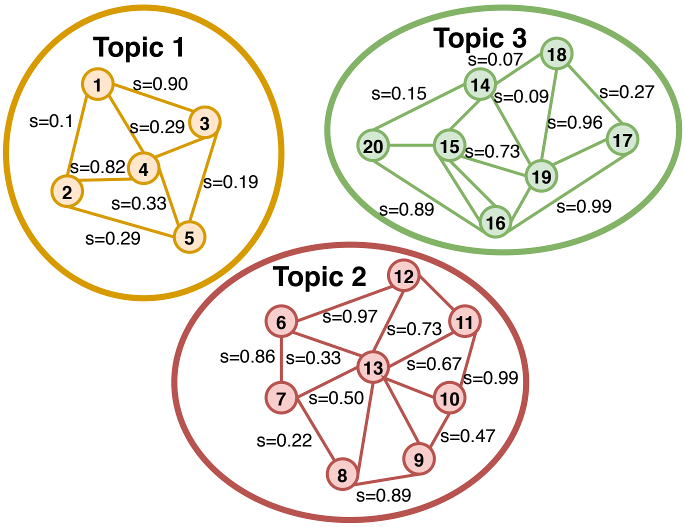


Figure 1.9: A proximity graph showing topic-wise groupings of interlinked datasets in the DL with their similarity scores

We seek to detect topic-wise categories of datasets in the DL describing similar real-life subject areas and business domains. This includes groupings of highly related datasets with similar data components stored inside them, as depicted in the proximity graph in Figure 1.9. Here we show groupings of datasets into topic-wise clusters. Each numbered node is a dataset and the edges connecting them show a similarity score, where 1 means most similar and 0 means most dissimilar. The goal is to be able to maintain such groupings over the DL, where each topical cluster consists of interlinked datasets that are similar to each other.

We tackle this challenge by first extracting metadata describing the content of the datasets in the DL in order to implement this categorization. We use the relationships detected between pairs of datasets to find their proximity (similarities). Then, we use a measure of proximity between datasets using those detected relationships within the categorization algorithm, and finally we allocate datasets to the topics existing in the DL based on their most similar cluster found in the DL or indicating that a dataset is an outlier that does not belong to any such topic if no highly similar datasets are found [10, 105].

We allocate Chapter 5 to discuss the concept of automatic dataset categorization into pre-existing topics-of-interest where we are able to group datasets containing data about the same topic into the same grouping. This is a first step toward fully automatic dataset categorisation for governing diverse DLs.

5.3 Metadata Query Interface

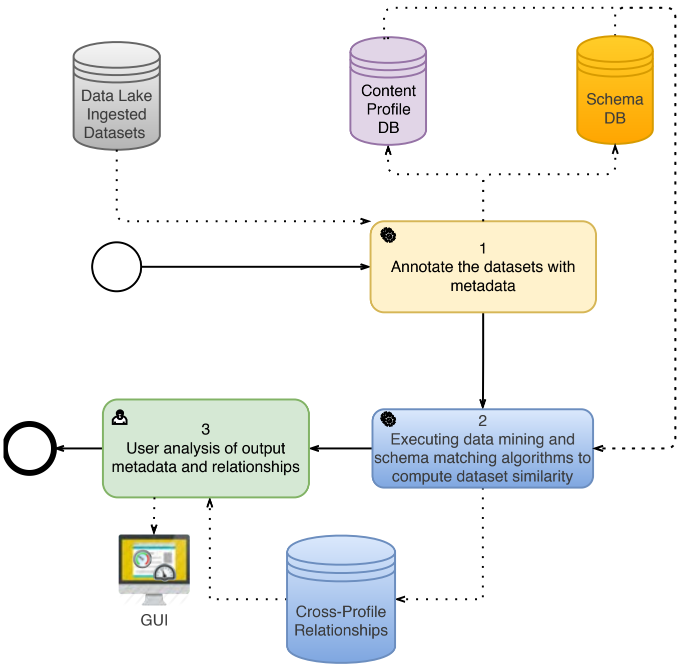

Data Lake Ingested Datasets
Content Profile DB
Schema DB
1
Annotate the datasets with metadata
2
Executing data mining and schema matching algorithms to compute dataset similarity
3
User analysis of output metadata and relationships
GUI
Cross-Profile Relationships

Figure 1.10: DL content metadata management and analysis process

Our ultimate goal is to be able to store all content metadata and cross-dataset relationships extracted within our framework in a common repository which could be queried by data scientists to find relevant data in the DL. This can be seen in the process visualised in Figure 1.10. It is a 3-steps process, where first the datasets ingested in the DL are annotated with their schema metadata and content profiles (output statistics about datasets and their attributes using data profiling). This is stored in metadata DBs. Then, these metadata are consumed by the proximity mining (see Section 5.1) and schema matching techniques (see Section 3.1) in order to compute dataset similarity between pairs and to store relationships between them in a cross-profile relationships repository. Finally, the output metadata and cross-profile relationships are presented to the user in a queryable GUI. We demonstrate this process in Chapter 6.

6 Thesis Contributions

This thesis covers different techniques to collect useful metadata about the content of the DL, which supports in better understanding the datasets stored and their relationships, including automatic algorithms for that collection. The research undertaken is expected to contribute to multiple fields, but mainly to the data management community. The outcome directly contributes to data management by creating new knowledge about managing metadata, which includes techniques to efficiently and effectively gather them to support holistic schema matching, information discovery and information summarisation.

Thus, the thesis formalises the DL metadata management process for governance purposes, experimenting with some schema matching techniques for detecting relationships, and using DM techniques to compute proximity between pairs of datasets. This includes reviewing the different techniques previously utilised in the research literature and adapting/improving those techniques in a holistic process to answer the research questions in Section 4. In addition, we experiment with the data ingestion process of metadata extraction, which allows for the utilisation of schema matching and ontology alignment techniques, like those in [126], to successfully compute similarity and cross-schema relationships between datasets. A novel DM-based approach to compute data profile similarity to find duplicates and relationships is also evaluated for the early-pruning step of holistic schema matching, making some new development and advancement for this topic.

For information discovery, the techniques proposed help discover data profiles and relationships between datasets to support identification of domains within the DL and identification of related datasets. As for information summarisation, the techniques investigated support producing statistical summaries and patterns from the data which efficiently describe their content. Finally, data schemas are extracted to maintain knowledge about dataset structures and how they can be used together to improve data utilisation. In addition, the output metadata are managed to enhance integration and querying for analytical discovery, which helps in the utility of the DL.

We implement and experiment with new proximity mining techniques (we describe them in Section 5.1) and prove their capabilities in accurately and efficiently computing similarities between heterogeneous schemata. We propose generic techniques that do not assume any specific topic or business domains, and improve the performance of the schema matching pre-filtering task, outperforming the state-of-the-art instance-based schema matching techniques. In our proposed techniques, we use metadata-based approximation when matching datasets and their attributes instead of the current state-of-the-art involving exact-value string-matching techniques. This helps in finding related attributes and datasets where the data are coded or named differently between datasets. Thus, we handle the shortcomings of current techniques for dataset similarity computation (see Section 3.2) by developing a proximity mining based approach in Chapters 3 and 4 which achieves the following:

- Extract automatic metadata about the data content stored in datasets at the attribute- and dataset- levels.

- Propose *similarity approximation* techniques which do not depend on exact value matching and name based string-matching only.
- Propose generic approaches which do not assume the existence of any specific attribute value formats, conversion rules, topic domains, or the existence of pre-defined metadata describing the datasets and their attributes like ontology mappings.

This thesis examines multiple directions of research in data management, where the main contributions include answering the RQs in Section 4 as follows:

1. [$RQ1$] Develop an approach for dataset similarity computation using instance-level schema matching between datasets.
2. [$RQ2$] Formulate a supervised machine learning technique to find proximity between heterogeneous datasets covering diverse topics.
3. [$RQ2$] Develop an *approximate* metadata-based schema matching approach where the goal is not to depend solely on syntactic string-based matching but also on data profiles.
4. [$RQ3$] Experiment with techniques for schema matching early-pruning and introduce the concept of *pre-filtering* using proximity mining.
5. [$RQ3, RQ4$] Propose a framework that utilises proximity mining in DL governance, mainly for dataset categorisation and schema matching pre-filtering.
6. [$RQ5$] Develop a GUI to query and use the output from our work to help in discovering the datasets stored in the DL and their relationships.

7 Structure of the Thesis

We aim to conduct the following tasks: (i) Framework conceptualisation and formalisation for the holistic proximity mining and metadata management in the DL, (ii) Prototyping with instance-based techniques for matching datasets, (iii) Implementing efficient and effective proximity mining techniques for early-pruning within holistic schema matching, (iv) Efficient pre-filtering schema matching implementation, and (v) Effective categorization techniques for segmenting the DL.

The thesis is organised as follows:

7.1 Chapter 2: Instance-level value-based schema matching for mining proximity between datasets

This chapter focuses on Objectives A and C and tackles Research Question RQ1. The goal is to apply instance-level value-based schema matching techniques to find related attributes between datasets and to compute overall dataset-level similarity. This includes the implementation of dataset similarity computation based on probabilistic value-based string matching techniques [126]. The computed relationships are aggregated to give overall dataset-level proximity scores between datasets. We experiment with different string matching techniques and different instance sampling sizes, where we test our proposed approach on a sample from a real-world DL to prove its effectiveness. The experiment includes comparison of performance of our proposed automated approach against human-based analysis, where our approach was able to achieve a better performance when compared to humans. The approach was found to be effective as a first step towards fully automated metadata collection and management framework fro DLs. However, some shortcomings were found, including the expensive computations required for the schema matching task which need to improve by using a more efficient approach. This is achieved in Chapters 3 and 4 using higher-level approximate schema matching pre-filtering techniques.

The chapter incorporates the tasks ING01, ING02, DIG03 and DIG04 from Figure 1.8. As this approach handles the raw dataset and does not use any content-based metadata, it does not need to implement tasks DIG01-02.

7.2 Chapter 3: Dataset-level content metadata based proximity mining

This chapter focuses on Objectives A and C and tackles research question RQ2. We introduce the concept of *proximity mining* where we collect high-level, overall dataset content metadata using data profiling techniques, and we use these metadata in supervised machine learning techniques for computing cross-dataset proximity (similarities) and duplicate schemata. We experiment our proposed techniques on a real-world DL, where we are able to detect related and duplicated datasets using high recall rates and efficiency gains. This includes tasks DIG01, DIG04 and EXP01 from Figure 1.8.

7.3 Chapter 4: Attribute-level content metadata based proximity mining for pre-filtering schema matching

This chapter focuses on Objectives A, B, and C and tackles research question RQ3. We aim to improve the proximity mining techniques from Chapter 3 by computing finer-granularity proximity scores between attributes and then computing overall dataset-level proximities by aggregating the individual proximity scores between the top-matching attributes for a dataset pair. We demonstrate an application of our proposed techniques by pre-filtering schema matching comparisons effectively and efficiently. Here, we are able to filter irrelevant dataset pairs with a low proximity score from further detailed value-based schema matching at the instance-level. We compare our approach against the state-of-the-art instance-based matching techniques and prove that our proposed approach is more efficient and effective in detecting related datasets. This supports the goal of early-pruning, where we reduce the amount of comparisons for schema matching. This includes tasks DIG02, DIG03, DIG04 and EXP02 from Figure 1.8.

7.4 Chapter 5: Automatic categorization of datasets using proximity mining

This chapter focuses on Objectives B and C and tackles research question RQ4. The goal is to identify a category of a dataset based on the most similar datasets already existing inside the DL. We devise a k-nearest-neighbour based approach to handle this task, where we find the top-k most similar datasets depending on the extracted cross-dataset relationships metadata and we allocate the most common category among those datasets as the appropriate category. We experiment using a real-world DL sample and find that our approach is highly effective in terms of recall and precision. This includes task EXP03 from Figure 1.8.

7.5 Chapter 6: Prox-mine tool for browsing DLs using proximity mining

This chapter focuses on Objective C and tackles research question RQ5. Here, we integrate the different components and techniques of our approach into a single prototype to demonstrate the value of proximity mining techniques in a real-world setting. We implement a prototype that can handle dataset ingestion, digestion and metadata exploitation for browsing the DL. This includes tasks EXP04 from Figure 1.8, in order to support the DL user in the following tasks:

- Find related and duplicated datasets.
- Identify the category of a new dataset based on the most similar datasets existing in the DL.
- Identify the top-matching attribute candidates between a dataset pair for detailed schema matching.

- Browse the DL based on a *proximity graph* visualisation that shows the relationships between datasets inside the DL using their computed proximity scores.

Chapter 2

Instance-level value-based schema matching for computing dataset similarity

Everything starts as an experimental prototype under development, then it keeps changing, learning, evolving, and adapting over time. Everything that gets measured eventually gets managed. One never stops learning!

This chapter has been published as a paper in the proceedings of the 16th International Conference on Data Mining Workshops (ICDMW) (2016) [15]. The layout of the paper has been revised
DOI: https://doi.org/10.1109/ICDMW.2016.0033

IEEE copyright / credit notice:
© 2016 IEEE. Reprinted with permission from Ayman Alserafi, Alberto Abelló, Oscar Romero, Toon Calders. Towards Information Profiling: Data Lake Content Metadata Management, ICDM workshops, 2016.

Abstract

There is currently a burst of Big Data (BD) processed and stored in huge raw data repositories, commonly called Data Lakes (DL). These BD require new techniques of data integration and schema alignment in order to make the data usable by its consumers and to discover the relationships linking their content. This can be provided by metadata services which discover and describe their content. However, there is currently a lack of a systematic approach for such kind of metadata discovery and management. Thus, we propose a framework for the profiling of informational content stored in the DL, which we call information profiling. The profiles are stored as metadata to support data analysis. We formally define a metadata management process which identifies the key activities required to effectively handle this. We demonstrate the alternative techniques and performance of our process using a prototype implementation handling a real-life case-study from the OpenML DL, which showcases the value and feasibility of our approach.

1 Introduction

There is currently a huge growth in the amount, variety, and velocity of data ingested in analytical data repositories. Such data are commonly called Big Data (BD). Data repositories storing such BD in their original raw-format are commonly called Data Lakes (DL) [128]. DL are characterised by having a large amount of data covering different subjects, which need to be analysed by non-experts in IT commonly called data enthusiasts [91]. To support the data enthusiast in analysing the data in the DL, there must be a data governance process which describes the content using metadata. Such process should describe the informational content of the data ingested using the least intrusive techniques. The metadata can then be exploited by the data enthusiast to discover relationships between datasets, duplicated data, and outliers which have no other datasets related to them.

In this chapter, we investigate the appropriate process and techniques required to manage the metadata about the informational content of the DL. We specifically focus on addressing the challenges of variety and variability of BD ingested in the DL. The metadata discovered supports data consumers in finding the required data in the large amounts of information stored inside the DL for analytical purposes [132]. Currently, information discovery to identify, locate, integrate and reengineer data consumes 70% of time spent in data analytics project [128], which clearly needs to be decreased. To handle this challenge, this chapter proposes (i) a systematic process for the schema annotation of data ingested in the DL and (ii) the systematic extraction, management and exploitation of metadata about the datasets' content and their relationship by means of existing schema matching and ontology alignment techniques [13, 24, 126].

The proposed process allows for the automation of data governance tasks for the DL. To our knowledge, the proposed framework is the first holistic approach which integrates automated techniques for supporting analytical discovery of cross-DL content relationships, which we call *information profiles* as explained below. This should cater for the current shortcoming of a formalized metadata management process to prevent the DL from becoming a *data swamp*; that is a DL that is not well governed and can not maintain appropriate data quality. Data swamps store data without metadata describing them, decreasing their utility [13].

Information Profiling. Traditional schema extraction and data profiling involves analysing raw data for detecting structural patterns and statistical distributions [98]. There is currently a need for higher-level profiling which involves analysing information about the approximate schema & instances relationships between different datasets instead of just single datasets [58], which we specifically define as *Information Profiling*. This involves the analysis of metadata and schema [58, 116] extracted from the raw data using ontology alignment techniques [24, 126]. Such techniques exploit 1. schema metadata and 2. data profile metadata to match different attributes from different datasets, generating the information profile. A schema profile describes the schema of datasets, e.g. how many attributes, their data types, and the names of the attributes [1]. The data profiles considered describe the values of the dataset, i.e. the single-attribute statistics of values [98]. Information profiles, the 3^{rd} type of content metadata, exploits the patterns from data profiles and data schemas [132]. For example, annotating attributes which can be linked based on approximate similarity of data distributions and data types.

Content Metadata. Content metadata is the representation of all types of profiles in the DL. Of our interest is augmenting metadata describing the informational content of datasets as first-class citizens, in order to support exploratory navigation of the DL. This involves representing the schema and profiles of data ingested in semantic-enabled standards like RDF (https://www.w3.org/RDF), which is a recommendation of the W3C for representing metadata. It is important to have metadata in semantically-enabled formats, because it supports information profiling using schema matching and ontology alignment techniques like [24, 126].

Contributions. The main contribution here is an end-to-end content metadata management process which provides *a systematic approach for data governance*. We identify the key tasks and activities for content metadata management in the DL for alignment purposes [76]. We focus on detecting three types of relationships: duplicate datasets, related datasets (i.e. “joinable" data attributes between datasets), and outlier datasets. This includes (i) identification of what content metadata must be collected for detecting relationships between datasets. In addition, (ii) identification of methods to collect such metadata to annotate the datasets. Finally, (iii) we prove the feasibility of our approach using a prototype applied to a real-life case-study. With the challenge of new formats of raw data flowing inside the DL and the high variability of such data, the answer to these challenges is non-trivial. Difficulties here include effective techniques for sampling the data to improve efficiency, applying the right matching techniques and efficiently using them for convergence. We propose a framework catering for those challenges which considers the schema, data, and information profile metadata managed.

For the remainder of the chapter: we review related work in Section 2; we demonstrate our approach using a motivational case-study in Section 3; we propose a framework & process for managing such metadata in Section 4; we showcase a prototype implementing our approach in Section 5; we follow with results from experimenting with the prototype on the DL from the motivational example in Section 6; and we conclude the chapter with a discussion of the metadata management approach and recommendations for future research in Sections 7 and 8.

2 Related Work

There is currently missing a holistic approach of informational content metadata management to support the data enthusiast [128, 91]. The DL also needs to have accompanying metadata to prevent it from becoming a data swamp [13]. Currently, data profiling and annotation is of great importance for research in DL architectures and is currently a hot topic for research [132, 105, 92]. Some techniques and approaches were previously investigated, but are mainly focused on relational content metadata [98, 1], free-text metadata [92], or data provenance metadata [128, 65]. Most of the current research efforts are suggesting the need for a governed metadata management process for integrating different varieties of BD [58, 92, 30]. This is currently handled by manual inspection of the data in the DL which consumes a lot of time and results in a big analytical latency [30]. Our proposed framework handles this metadata using automatic techniques.

Many research efforts are targeting *extraction of schema and content metadata*. Those provide an overview of techniques, algorithms and approaches to extract schemas, matching schemas, and finding patterns in the data content of data files [92, 130]. There is also research to detect **cross-data relationships** which aim at detecting similar data files with similar informational concepts [30, 90].

Ontology alignment and schema matching techniques which are based on finding similarity between data schemas and instances of data can also be utilised to integrate datasets [24]. This can be achieved by extracting the schema and ontology from the data and then applying the matching techniques [130].

The current shortcoming of research about managing metadata in the DL is that the available techniques are not formally defined as a systematic process for data governance, it is still applicable only to relational data warehouses, and does not handle the automatic annotation of informational content of datasets in the DL. We cover this gap by proposing an automatic content metadata management process. The ontology alignment techniques were also classically applied to discover similarity between two large ontologies [126],

Table 2.1: Description of OpenML datasets

Domain	Datasets IDs	Datasets
Vehicles	21,455,967,1092	car,cars,cars,Crash
Business	223,549,841	Stock,strikes,stock
Sports	214	baskball
Health	13,15,37	breast-cancer,breast-w,diabetes
Others	48,50,61,969	tae,tic-tac-toe,Iris,Iris

but have not been sufficiently applied before to duplicate detection, outlier detection and cross-datasets relationships extraction on multiple discrete datasets.

3 Motivational Case-Study

In order to demonstrate the feasibility and value of our systematic approach for content metadata discovery, we implement a prototype called *Content Metadata for Data Lakes (CM4DL)*. This prototype is tested with a real-life example of a DL called OpenML[1]. OpenML is a web-based data repository which allows data scientists to contribute different datasets which can be used in data mining experiments [131]. The OpenML platform supports loading different types of data which are stored in the WEKA[2] format (i.e. ARFF). OpenML stores datasets which represent diverse data domains, and can be considered a DL because it involves raw data loaded without a specific integration schema and which represent diverse subject-areas intended for analytics. A subset of this DL involving 15 datasets categorized into 5 subject-areas were used in our experiments and can be seen in Table 2.1 (it uses the OpenML dataset-ID, which can be used for retrieving the data using the OpenML API[3]. The dataset names from OpenML are given in the last column).

OpenML provides pre-computed data profiles for each dataset as JSON files (retrievable by the API too), which we have parsed in our prototype and used to compare the datasets with each other. This includes the statistical distribution of numerical attributes and the value frequency distribution of nominal attributes [131]. The datasets will be used in our experiments as input datasets. They will be automatically annotated to describe their content (attributes and instances). Each dataset consists of a number of attributes for each instance. Each instance of a dataset shares the same attributes.

[1]http://www.openml.org

[2]http://www.cs.waikato.ac.nz/ml/weka

[3]http://www.openml.org/guide

$$x = [d(d-1)/2] * m^2 \tag{2.1}$$

The number of attributes is 10 per dataset on average. In order to compare all attributes together from those datasets, there needs to be about 10500 comparisons according to Equation 2.1. This approximates the number of comparisons x in terms of d number of datasets and m number of attributes, not comparing a dataset to itself or to other datasets twice. This is very difficult for a human to achieve and will require a huge effort (as will be described in the experiments in Section 6). Therefore, it is important to have an automatic process which is capable of efficiently executing those comparisons and capturing the important informational relationships between the datasets. Challenges which arise include efficiently handling the large varieties of datasets in OpenML. The automated process handling this is described in the next Section.

4 A Framework for Content Metadata Management

In this section, we propose a framework for the automatic management of metadata about the DL. The goal is a cross-datasets relationships aware DL which can be navigated easily. This framework integrates the different schema matching and ontology alignment techniques for the purpose of information profiling. Metadata annotation can be efficient and does not heavily affect processing times of datasets in the DL as shown in related experiments like [92, 65] and in our experiments in Section 6.

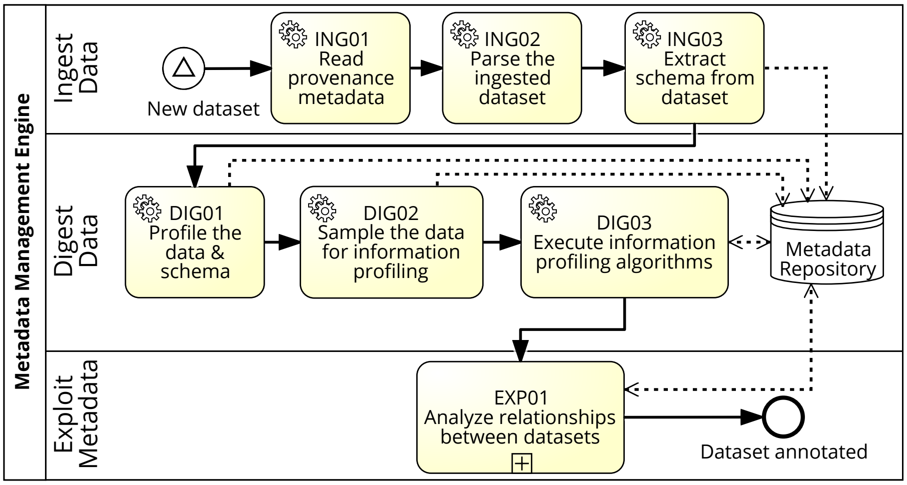

Metadata Management Engine
Ingest Data
Digest Data
Exploit Metadata
New dataset
ING01 Read provenance metadata
ING02 Parse the ingested dataset
ING03 Extract schema from dataset
DIG01 Profile the data & schema
DIG02 Sample the data for information profiling
DIG03 Execute information profiling algorithms
Metadata Repository
EXP01 Analyze relationships between datasets
Dataset annotated

Figure 2.1: The Metadata Management BPMN Process Model

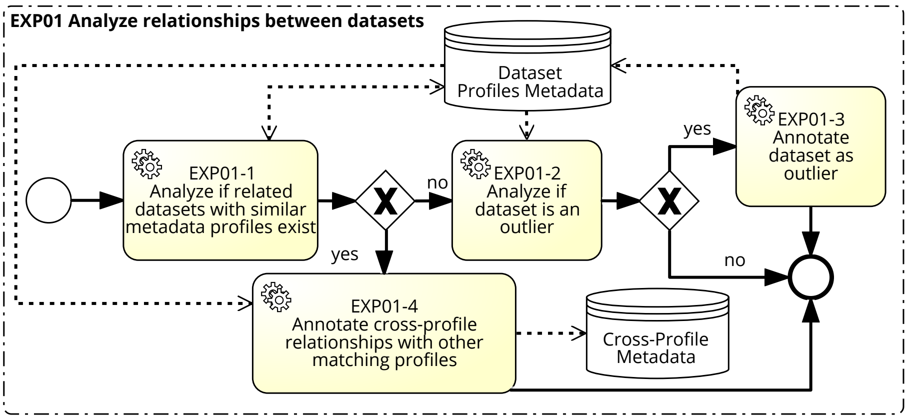

EXP01 Analyze relationships between datasets
Dataset Profiles Metadata
EXP01-1 Analyze if related datasets with similar metadata profiles exist
no
yes
EXP01-2 Analyze if dataset is an outlier
yes
no
EXP01-3 Annotate dataset as outlier
EXP01-4 Annotate cross-profile relationships with other matching profiles
Cross-Profile Metadata

Figure 2.2: The EXP01 Metadata Exploitation Sub-Process Model

The framework involves 3 main phases. The first phase is **data ingestion** which includes discovering the new data by its provenance metadata, parsing the data, extracting the schemas from the data (similar to [130]), and storing the data in the DL with its annotated *schema metadata*. The second phase is **data digestion** which means analysing the data flowing to the DL to discover informational concepts and data features. This phase includes data profiling, schema profiling and ontology alignment to extract information profiles (i.e. extraction of all *content metadata* artefacts). The datasets are annotated with their profiles in the *metadata repository*. The third phase includes **metadata exploitation** like discovering relationships between datasets. This involves information profiling which exploits the *content metadata* from the data digestion process to detect and annotate the relationships of a dataset with other related datasets which can be analysed together [132]. Those are called *cross-dataset relationship metadata*.

The framework is implemented by a structured metadata management process, which can be seen in Figure 2.1. This facilitates systematically collecting and maintaining the metadata throughout the lifetime of the DL. To define the activities for this framework, we present a BPMN process model. Each activity in the BPMN model is described below by the technique along with its computational complexity, and description of what is achieved.

Start & data ingestion. The dataset annotation process starts when a signal arrives to the metadata engine, indicating that a new dataset is uploaded to the DL. In ING01, the dataset is located using its provenance metadata in $O(1)$ time. Then it is parsed in ING02 to verify its structural correctness in $O(n)$ time, where n is number of instances. The dataset is then analysed in activity ING03 to extract and annotate the schema semantics in $O(m)$ time, where m is the number of attributes. This is done using RDF ontology

extraction techniques like in [130]. The generated metadata are stored in a semantic-aware metadata repository (i.e. RDF Triplestore[4]).
Data digestion. The dataset is then digested to extract the content metadata. This starts by DIG01 which creates the data profile and schema profile using simple statistical techniques and profiling algorithms similar to [98]. This is done in $O(n)$ time. The following activity DIG02 samples the data instances to improve the efficiency of the information profiling algorithms in the next activity, which is completed in $O(1)$ time. In DIG03, the dataset and its profiles are compared to other datasets and their profiles using ontology alignment techniques, which requires $O(m^2)$ in worst case scenario [76]. We propose an algorithm to reduce this complexity in Section 5. There should be certain cut-off thresholds of schema similarity (like [92, 130, 90]) and data profile similarity [105] to indicate whether to align two datasets together, in order to decrease the number of comparisons made in this activity. Ontology alignment is used to extract metadata about the relationships with other datasets. The existing alignment techniques we utilize first hash and index the values from the data instances like [124], then use an alignment algorithm like [126] to match the attributes from the datasets. The dataset is analysed to extract its information profile and then goes to the exploitation phase of the framework.
Metadata exploitation. This starts in the EXP01 subprocess which detects relationships with other datasets using the content metadata stored in the *Metadata Repository*. This can be seen in Figure 2.2. This includes EXP01-1 which checks if similar attributes in other datasets exist by comparing the similarity stored between datasets' attributes against a specific threshold. If the similarity of attributes exceeds the threshold then related datasets exists, and the flow follows with EXP01-4 which annotates the cross-profile relationships and stores this as the *Cross-Profile Metadata*. We also discover duplicate datasets in EXP01-4. Those are datasets with the same profiles including the same schema structure, with the same number of attributes, and similar data profile (i.e. overlapping value frequency distributions). Otherwise, if there is no related datasets detected in EXP01-1, the dataset is checked in EXP01-2 to see whether it is an outlier [1]. The dataset is an outlier if it has no matching attributes with other datasets found in the metadata repository, and is annotated as an outlier in EXP01-3.

The remainder of this chapter discusses an instantiation of this process and experimental results from its implementation.

5 The CM4DL Prototype

In order to instantiate the BPMN model in Figures 2.1 & 2.2 and to prove its feasibility, we implement a prototype called *Content Metadata For Data Lakes*

[4]https://www.w3.org/wiki/LargeTripleStores

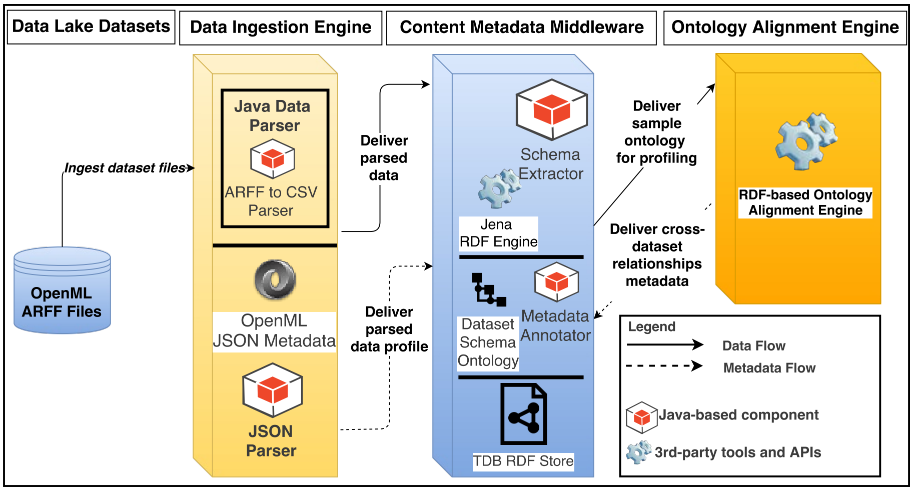


Figure 2.3: CM4DL System Architecture

(CM4DL in short). The prototype consists of multiple components and the system architecture can be seen in Figure 2.3. The prototype is based on a Java implementation. Tools and APIs which are developed by 3rd-parties, but which are utilised are shown using a different symbol as seen in the legend.

5.1 Prototype Architecture

The prototype consists of three main layers, in addition to the DL dataset files. The DL files containing the datasets are first read along with their accompanying JSON metadata from OpenML (using the OpenML Java-based API library). This retrieves the ARFF datasets and JSON metadata objects from the OpenML library and stores it on the local server machine. In the data ingestion engine layer, a Java data parser component is utilised. The data parser reads the ARFF files using the WEKA Java API and converts the datasets to CSV files. This conversion is based on mapping the ARFF attributes to CSV columns. The parser utilises the JSON metadata files provided by the OpenML API which describe each attribute and its data-type. This provides us with pre-computed data-profiles describing the dataset and each attribute in the dataset. For numeric attributes, the metadata includes min, max, mean, and standard deviation. For nominal and string attributes, it provides the full frequency distribution of values.

The next layer is the main component for content metadata management, which is called Content Metadata Middleware. It is responsible for first converting the datasets from OpenML to RDF schemas. This is done using the schema extractor which loads the CSV files into a Jena TDB RDF triplestore.

Those files are sampled for a specific number of instances, and are then parsed using the Jena RDF library for Java[5]. The output leads to a mapping of each ingested dataset with an RDF N-triple ontology representing the schema and its sampled instances. Each dataset is represented as an RDF *class* and each attribute as an RDF *property*. Finally, a metadata annotator uses the generated ontology mappings to discover relationships between datasets consisting of similar attributes. This matching task is done using the ontology alignment engine in the last layer which detects schema- and instance- based relationships between datasets, and returns them to the metadata annotator to be stored in the *Metadata Repository*.

Each component in the system architecture of the prototype implements and automates activities from the BPMN process in Figure. 2.1. The Java Data Parser handles the activities ING01 and ING02. The schema extractor in the middleware layer handles activity ING03. The data and schema profile is ingested in the OpenML JSON metadata and JSON parser which provide the profile metadata for the middleware. The metadata annotator in the middleware is able to exploit this profile metadata and the schema metadata to detect duplicates using profile querying and ontology alignment respectively. This handles activity DIG03. To detect relationships between datasets, EXP01 is implemented in the ontology alignment layer to detect related datasets and their related attributes by analysing the information profiles extracted in DIG03.

5.2 Ontology Alignment Component

In the CM4DL prototype, we utilize the existing ontology alignment engines to facilitate our approach. The field of ontology alignment is very developed and to understand the basic techniques of such tools you can refer to the following references: [24, 126, 124]. In order to select an appropriate tool for our task, we evaluated the research literature for a tool which supports the following:

- **Schema- and instance- based ontology alignment:** the tool needs to analyse both the schema (attribute types and dependencies) and the instances (values for the attribute) to check for similarity. [24] compares such techniques.
- **Indexing and hashing techniques (like the MinHash algorithm [124]):** this is essential to speed-up the comparison of the datasets and to make this more efficient.
- **Different techniques of instance-based similarity:** the tool should implement different techniques for comparing values of instances like different

[5]https://jena.apache.org

string-comparison techniques (e.g., normalised identities [126], shingling-MinHash distances [124], etc.). The different similarity comparison techniques can yield different effectiveness with different types of data. Therefore, it is important to study different comparison techniques for effectiveness and efficiency in our task.

- **Open-source Java API:** the tool must expose an open-source API which is integrable within our developed prototype.

From the short-listed tools, according to the above criteria, we identified COMA++ [24] and PARIS probabilistic ontology alignment [126] as possible candidates. We selected PARIS because of its simplicity in integration with a Java-based API and being cited for having high effectiveness with large-scale ontology alignment when compared against other tools and benchmarks (see [76]). PARIS aligns ontologies [126] by finding RDF *subclasses* which in our case indicates the similarity of datasets, and RDF *subproperties* indicating similarity of attributes in the datasets. Similarity is given as a percentage; higher values means more similar.

The ontology alignment tool is capable of reading two ontologies and detecting the degree of similarity between the two ontologies based on the schema and instances in the ontology [126]. The ontology must be defined in N-Triples[6] RDF representation. The metadata annotator component can send any two datasets in N-Triples format and then the tool will return the similarity of classes (i.e. datasets) from both ontologies (coefficient between 0 and 1) along with similarity between both datasets attributes (modelled as RDF properties). The similarity is based on comparing instances (modelled as RDF concepts) using string matching techniques. In PARIS [126], there are two techniques provided, which we have used in our prototype: the identity-based exact match [126] and the shingling-based MinHash approximate matching [124]. For the identity-based approach the attribute values are normalized by removing punctuation marks and converting the characters to lower-case. The normalized text are then compared for exact matches. This works best for numeric attributes with exact values. The shingling-based approach compares n-grams of text (i.e. specific number of character sequences) and is better suited for approximate matching of strings.

Relationships detected in this layer include examples like those in Table 2.2 which are based on the OpenML datasets used in the experiments. The table compares attributes from two datasets by showing their relationship. Each dataset is described by its OpenML ID and dataset name. The attribute name from each dataset is then given. Finally a relationship is listed as either: related, duplicate, or outlier. The relationship *related* is used to identify similar attributes which can be used to “link" the datasets together. The relationships

[6]https://www.w3.org/TR/n-triples

Table 2.2: Example Cross-dataset Relationships

No.	Dataset 1	Dataset 2	Attribute 1	Attribute 2	Relationship
1	37 (diabetes)	214 (baskball)	age	age	related
2	455 (cars)	549 (strikes)	model.year	year	related
3	455 (cars)	967 (cars)	all	all	duplicate
4	455 (cars)	1092 (Crash)	name	model	related
5	455 (cars)	1092 (Crash)	weight	Wt	related
7	50 (tic-tac-toe)	N/A	all	N/A	outlier

are identified by analysing the similarity of the actual value distribution of the attributes as exhibited by the instances of data in the dataset [126]. Related attributes should have an overlapping distribution of values which can be used to link the attributes together. The ontology alignment algorithms should be capable to detect that attributes like those in relationships no. 2, 4, and 5 are related. Although the attributes have different names in the schema, their values are overlapping and hold similar character- or numeric- values. Therefore, it is important to use ontology alignment which is instance-based to detect such relationships.

In addition, when all attributes are related with attributes of another dataset we call this relationship a *duplicate* relation. This means the datasets contain similar informational content in all their attributes. This can be seen for example in Table 2.2 row 3. Detecting duplicates can help in data cleansing and de-duplication by eliminating or merging them to maintain high data quality in the DL with less redundancy. It is based on taking a cut-off threshold of similarity generated by the ontology alignment tool to indicate if datasets are duplicates (e.g. taking 0.8 for similarity of all attributes). Finally, an *outlier* is a dataset which has no related attributes in any of the other datasets in the data lake. For outliers, all attributes of a dataset have no matching attributes in any other dataset.

5.3 Dataset Comparison Algorithm

In order to match the datasets we use Algorithm 1. It automates the information profiling activity DIG03 in Figure 2.1, however, note that the BPMN describes the handling of each separate dataset while the algorithm describes the overall *collective* handling of datasets. The matching algorithm is based on the ontology alignment similarity measure [126] and the average data and schema profile similarity. The profile similarity is calculated as the average of the difference between the normalized profile features from each dataset. The list of profile features used includes: the number of attributes in the dataset, number and percentage of numerical/binary/symbolic attributes, the number of classes for the target variable, the size and percentage of the majority and

```
Algorithm 1: DatasetSimilarityMatching
Input: $DLNTripleFiles, ProfileMetadata, ProfileThreshold, RelationThreshold, DuplicateThreshold$
Output: $Duplicates, Relationships, Outliers$
begin
1    $D \leftarrow (DLNTripleFiles, ProfileMetadata)$
2    $Duplicates, Relationships, Outliers \leftarrow \{\}$
3    foreach $d \in D$ do
4        $P \leftarrow D \setminus \{d\}$
5        foreach $p \in P$ do
6            $psimilarity \leftarrow AvgProfileSimilarity(d, p)$
7            if $psimilarity > ProfileThreshold$ then
8                $Sem \leftarrow parisSimilarity(d, p)$
9                foreach $r \in Sem$ do
10                   if $s(r) > RelationThreshold$ then
11                       $Relationships \leftarrow Relationships \cup \{r\}$
12                   End If
13               if $\forall a_1 \in Attributes(d), \exists a_2 \in Attributes(p) \wedge (d, a1, p, a2, s) \in Sem \wedge s >$
                   $DuplicateThreshold$ then
14                   $Duplicates \leftarrow Duplicates \cup \{(d, p)\}$
15               End If
16       $D \leftarrow D \setminus \{d\}$
17   $D \leftarrow (DLNTripleFiles, ProfileMetadata)$
18   foreach $d \in D$ do
19       if $\nexists (d, a, d_2, a_2, s) \in Relationships$ then
20           $Outliers \leftarrow Outliers \cup \{d\}$
21       End If
22 return $Duplicates, Relationships, Outliers$
```

minority classes of the target variable, the number of instances in the dataset, percentage of instances with missing values, and the dimensionality measure. This subset of features were selected because they are the most frequently occurring in the OpenML JSON metadata.

The algorithm consists of input DL N-Triple files (*DLNTriples*), the JSON metadata features (*ProfileMetadata*), and the thresholds for matching datasets on the basis of profile metadata (*ProfileThreshold*), or thresholds for matching attributes in the ontology alignment tool as related (*RelationThreshold*) or duplicates (*DuplicateThreshold*). The output of the algorithm is 3 sets consisting of discovered relationships. If two datasets d_1 and d_2 are duplicates of each other they are added to the *Duplicates* set as a tuple (d_1, d_2), if they are not related to any other dataset then they are added to the *Outliers set* as a tuple of the dataset identifier (d_x) and if they are related to other datasets then the exact attributes from both datasets a_1 and a_2 with relationships (similarity measure s between 0 and 1) between them are added to the *Relationships set* as a tuple 'r' of (d_1, a_1, d_2, a_2, s).

The algorithm loops (Lines 3-16) on each dataset (and its accompanying profile) and compares it with each of the other datasets (in set 'P' from Line 4) based on the similarity of their data and schema profiles *psimilarity*. If the *psimilarity* is bigger than the assigned threshold in the input then the ontology similarity is computed within the inner-loop of Lines 5-15 which compares each dataset with each of the other datasets not checked before by the algorithm. This filtration If-statement (Line 7) is used to prevent non-necessary expensive comparisons with ontology alignment tools for datasets with disjoint profiles. To prevent any filtration in this step, we can set the *ProfileThreshold* to 0. The *psimilarity* is calculated as the average similarity of all data-profile and schema-profile metadata features in *ProfileMetadata* for both datasets in *AvgProfileSimilarity*(d_1, d_2). In Line 8 we compute the ontology similarity *parisSimilarity* [126] between each attribute of the dataset and attributes of the other datasets not checked by the algorithm before (we guarantee not double checking datasets by removing them from the comparison list of 'D' at the end of the loop in Line 16). The set *Sem* in Line 8 contains relationships tuples 'r'. In Line 13, if all the attributes between both datasets have relationships with similarity exceeding the *DuplicateThreshold*, then we add the datasets to the *Duplicates* set. In Lines 18-21, if the dataset is not related to any other dataset (i.e. does not have any member tuple in the *Relationships* set), then we add it to the *Outliers* set in Line 20. To make the algorithm more efficient we take samples of instances for comparison in the N-Triples of each dataset element of 'D'. The worst-case complexity of the algorithm is given in Equation 2.1.

6 Experiments and Results

In this section, we describe the results of executing the prototype on the OpenML DL. To compare the automated approach and algorithm with the manual approach, we conduct an experiment with OpenML data. Our goal is to test the feasibility and effectiveness of our automated approach as compared to manual human checks. For the sample data of 15 datasets related to different domains as described in Table 2.1, we present these data to 5 human-experts to analyse the relationships (like those listed in Table 2.2) and then compare this to our automated approach. The human participants consisted of postgraduate pharmacists representing data enthusiasts. We have also independently analysed the datasets in 6 hours and have created a gold-standard of relationships, duplicates, and outliers for evaluating the manual and automatic approaches against. Such relationships detection includes analysis of two main types of attributes described below:

- **Numeric attributes:** Those include attributes represented as integers or real numbered values. They have a data profile involving statistical value distributions like mean, min, max, and standard deviations. An example would be the attributes in row no. 5 in Table 2.2 showing the continuous numeric value of the weight of cars in kilograms (e.g., 3000).

- **Nominal and String attributes:** Those include attributes having discrete values of nominal numbers or strings of characters. Their data profile mainly involves frequency distributions of their distinct values. An example would be the attributes in row no. 4 in Table 2.2 showing the name of the car models in the dataset in character strings. For example, the strings "volkswagen_type_3" and "Volkswagen". Although the values are represented in different strings of characters, they still hold the same information about *Volkswagen* cars and should be detected in the experiments as similar values.

For the automated CM4DL implementation, the thresholds used with Algorithm 1 were *0.5 or 0.0* for *ProfileThreshold*, *0.5* for *RelationThreshold* and *0.75* for *DuplicatesThreshold*. All experiments were executed on an i7-5500U Quad-core CPU, 8GB of memory and 64-Bit Windows 7 machine. We examine using the following alternatives:

- **Different sampling sizes:** we execute random sampling on the data instances to speed-up the ontology alignment task. We test using samples of sizes 100, 500, and 700 instances.
- **Different iteration counts until convergence:** In order to align the ontologies, the techniques used are usually iterative in nature and require multiple iterations until convergence [126]. The iterative nature allows for refinement of the matching results [76]. We test different number of iterations until we stop the alignment task. We test using 3,5,7, and 10 iterations.
- **Different similarity detection approaches:** We test two alternative approaches for similarity detection between attributes: the identity-based and the shingling-based matching. We also combine both approaches to detect relationships by running them both on the data and merging the output.
- **Different profile similarity thresholds:** We examine using the average profile similarity between datasets as a filtering technique to eliminate comparisons using ontology alignment. This involves eliminating datasets having a profile similarity below a threshold. We test two thresholds: 0.5 and 0. For the later threshold, it means we do not filter any comparisons.

We compare the overall standard precision, recall and F1 measures [86] for the relationships detected. For the human-experts, their results are summarized in Table 2.3. As can be seen, it takes considerable effort and time to manually compare the datasets. It took on average more than 2 hours and up to 4 hours to annotate the datasets by a human. The precision average is also considerably low at 57.5% (with a min of 20.5% and max of 91.3%). For the recall, it was also at a low average of 61.1%. The overall F1 mean is at 55.6% which shows a need for improvement by automated techniques.

The graphs in Figure 2.4 show the assessment of the F1 measure and computational efficiency (timing) of executing the automated algorithm on

Table 2.3: Results of Manual Annotation

Participant	Time Taken	Precision	Recall	F1
1	0.66 hours	91.3	55.3	68.9
2	4 hours	66.0	92.1	76.9
3	3 hours	20.5	42.1	27.6
4	2.66 hours	28.8	60.5	39.0
5	1.5 hours	80.8	55.3	65.6

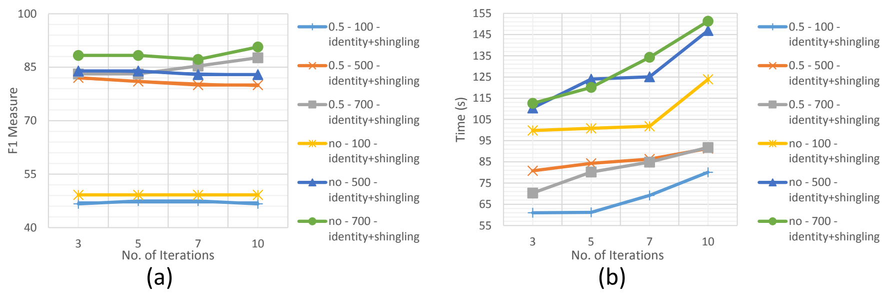


Figure 2.4: Performance analysis of CM4DL in the OpenML experiments

the experiment data. The time durations include the following: loading the datasets along with the JSON metadata to the triplestore, the tasks of the content metadata middleware in parsing the data and converting them into RDF N-Triples, and the ontology alignment execution time between all the datasets in the experimental setting. We test and compare for different number of iterations for the ontology alignment and matching execution, different sampling and different threshold of data profile similarity filtration. Graph (a) shows the F1 for the combined identity-similarity and shingling-similarity instance-matching techniques, and graph (b) shows the corresponding execution time of the Algorithm 1. Each line in the graphs represent the following as indicated in the legend: *ProfileThreshold* for Algorithm 1-sampling size-similarity technique. *ProfileThreshold* was tested at 0.5 and without any limit (as indicated by 'no').

From the graphs in Figure 2.4, it can be seen that the automatic approach yields good F1 scores between 82% and 91% for sample sizes between 500 and 700 instances. Generally, sampling negatively impacts the F1 score of the algorithm, however it has a bigger effect on smaller sample sizes like 100 instances which yielded F1 between 46% and 50%. Filtering the data profiles before comparisons proved to be effective in improving computational times while not severely impacting the F1 score. For a sample of 700 instances, we can still achieve 87% F1 (just 3% downgrading from comparing all datasets) while considerably saving computation time from 151s to 92s. It was noted, as expected, that more iterations of the ontology alignment algorithm yields more time for computations. However, there is no big downgrades from using less iterations, while it can considerably save processing time.

We only demonstrate the results from the combined approach in the graphs of Figure 2.4. However, it must be noted that the identity-similarity matching outperformed the shingling-similarity matching in all experiments. Shingling had an F1 score between 35% and 49% while taking more computation time between 63s and 82s for no filtering and 40s to 50s for filtering the data profiles. On the other hand, identity had an F1 score between 86% and 89% for sample sizes between 500 and 700 instances. For 100 instances samples, the effectiveness deteriorated sharply between 50% and 55%.

7 Discussion

As can be seen in the results, the automated techniques outperforms human subjects in both effectiveness (in terms of F1 measure) and efficiency (in terms of time for computation). By interviewing the human subjects, it was noted that they mainly focus on analysing nominal attributes without delving into numerical attributes analysis. In some subjects, there were almost no relations made within numeric attributes in the whole exercise, although they were instructed to do so. The automated techniques are more adept at comparing numeric features. On the flip side, humans are good at analysing nominal attributes as they can understand the semantic meanings behind them, which is difficult for a machine, e.g., for relationship no. 2 in Table 2.2, it is obvious for humans that a year '1975' is similar to '75' but the automated algorithms considering shingling-sizes of 3 or 4 can not easily detect this relationships (as was experienced in our experiments). Also, the general feedback from human subjects expresses that they feel reluctant when making correlations between data. Such data enthusiasts simply do not want to spend considerable time to take the same systematic approach as a machine.

The results show that sampling can considerably minimize the algorithm's execution duration while not severely impacting the performance of the algorithm. It was observed, as expected, that smaller sample sizes negatively impact effectiveness measured by F1. However, this impact becomes sharp with very small sample sizes only. For slight sampling variations in larger sample sizes, the F1 measure is not severely impacted. The results indicate that higher number of iterations for the alignment algorithms can yield better results, however, a more optimal number of less iterations can be selected without severely impacting the effectiveness of the algorithms. The efficiency gains from less iterations and more sampling can save considerable time. Sampling the data profiles from the datasets before comparison using ontology alignment techniques also saves sufficient computational time while not having considerable negative impact on the F1 score.

Duplicates were overall really well detected in all experiments applying the identity-based matching algorithm. Relationships in numeric attributes were better detected using identity-based matching, and for string attributes they were better detected using shingling-based matching. Combining both matching approaches led to the highest overall F1 score.

Shingling matching techniques generally have a low recall and precision but are good in detecting approximate relationships for string-based attributes. Shingling adds considerable errors especially in numeric attributes which reduces the precision. Identity-based techniques have high recall and precision but miss the cases of quasi-similarity string matching. Identity techniques have a high rate of recall and can easily detect duplicated datasets. It is therefore advisable to use multiple techniques in ontology alignment between datasets to improve the effectiveness in detecting the relationships between them.

The automated end-to-end process saves the huge manual effort required to analyse the datasets and annotating the metadata to the datasets. The automation results in some tens of seconds for metadata extraction and management instead of the multiple hours required by manual human inspection.

8 Conclusion and Future Work

We have presented our content metadata management framework which facilitates alignment in DL. We have demonstrated our approach within the OpenML DL environment. Our experiments shows the feasibility of our automatic approach in detecting relationships between the datasets. The results show that filtering the datasets for comparison, using sampling techniques, and using different ontology matching techniques can improve the efficiency of the approach while still achieving good effectiveness. We have also demonstrated the types of content metadata to collect for: schema, data profiles, and information profiles. This content metadata was used in a structured process to detect relationships between datasets, in order to facilitate the navigation and analysis of the DL.

For the future, we will examine the utilization of different supervised learning techniques to find the optimum similarity thresholds and weightings of the similarity measures to use in our algorithm. We will also investigate how to dynamically select the sample size based on a measure of heterogeneity of the datasets being compared together. We also acknowledge that we can improve the efficiency of the algorithm by creating a 3^{rd} reference integration ontology after each ingestion to decrease the number of comparisons by the algorithm to this single-integrated ontology. To improve the efficiency of our algorithm we are planning to parallelize the computations in a parallel-computing framework like MapReduce.

Acknowledgements. This research has been funded by the European Commission through the Erasmus Mundus Joint Doctorate (IT4BI-DC).

Chapter 3

Dataset-level content metadata based proximity mining for computing dataset similarity

A simpler solution with less knots and more abstraction can be better than a complex entangled monster stuck in the details!

This chapter has been published as a paper in the proceedings of the International Conference on Similarity Search and Applications (SISAP) (2017) [18].
The layout of the paper has been revised
DOI: https://doi.org/10.1007/978-3-319-68474-1_20

Springer copyright / credit notice:
© 2017 Springer, Cham. Reprinted with permission from Ayman Alserafi, Toon Calders, Alberto Abelló, Oscar Romero. DS-Prox: Dataset Proximity Mining for Governing the Data Lake, SISAP, 2017.

Abstract

With the arrival of Data Lakes (DL) there is an increasing need for efficient dataset classification to support data analysis and information retrieval. Our goal is to use meta-features describing datasets to detect whether they are similar. We utilise a novel proximity mining approach to assess the similarity of datasets. The proximity scores are used as an efficient first step, where pairs of datasets with high proximity are selected for further time-consuming schema matching and deduplication. The proposed approach helps in early-pruning unnecessary computations, thus improving the efficiency of similar-schema search. We evaluate our approach in experiments using the OpenML online DL, which shows significant efficiency gains above 25% compared to matching without early-pruning, and recall rates reaching higher than 90% under certain scenarios.

1 Introduction

Data Lakes (DL) [4] are huge data repositories covering a wide range of heterogeneous topics and business domains. Such repositories need to be effectively governed to gain value from them; they require the application of data governance techniques for extracting information and knowledge to support data analysis and to prevent them from becoming an unusable *data swamp* [4]. This involves the organised and automated extraction of metadata describing the structure of information stored [132], which is the main focus of this chapter.

The main challenge for data governance posed by DLs is related to information retrieval: identify related datasets to be analysed together as well as duplicated information to avoid repeating analysis efforts. To handle this challenge it was previously proposed in [15] (see Chapter 2) to utilise schema matching techniques which can identify similarities between attributes of different datasets. Most techniques proposed by the research community [24] are designed for 1-to-1 schema matching applications that do not scale up to large-scale applications like DLs prone to gather thousands of datasets.

To facilitate such holistic schema matching and to deal with the sheer size of the DL, [24] proposed to utilise the strategy of early pruning which limits the number of comparisons of pairs of datasets. We apply this approach in this chapter by proposing a technique which approximates the proximities of pairs of datasets using similarity-comparisons of their meta-features. More specifically, we use a supervised machine learning approach to model topic-wise related classification of datasets. We then utilise this model in assigning proximities between new datasets and those already in the DL, and then predicting whether those pairs should be compared using schema matching (i.e., have related information) or not. We implement this technique in the datasets-proximity (DS-Prox) approach presented in this chapter. Our focus is on early-pruning of unnecessary dataset comparisons prior to applying state-of-the-art schema matching and deduplication (the interested reader is referred to [24, 110] for more details on such techniques).

Our contributions include the following: 1. a novel proximity mining approach for calculating the similarity of datasets (Section 4), 2. applying our new technique to the problem of early-pruning in holistic schema matching and deduplication within different scenarios for maintaining the DL (Sections 2, 3), and finally, 3. testing the proposed proximity mining approach on a real-world DL to demonstrate its effectiveness and efficiency in early-pruning (Section 5).

The chapter is organised as follows: we present a description of the problem in Section 2, we present related work in Section 3, we present our proposed dataset proximity mining approach in Section 4, we experimentally evaluate our approach on a real-world DL in Section 5 and finally we conclude and present future work in Section 6.

2 Problem Statement

Our goal is to automate information profiling, defined in Chapter 2, which aims at efficiently finding relationships between datasets in large heterogeneous repositories of *flat data* (i.e., tabular data like CSV, web tables, spreadsheets, etc.). Those repositories usually include datasets uploaded multiple times with the same data but with different transformed attributes. We focus on two types of attributes: continuous numeric attributes and categorical nominal attributes, and two types of relationships for pairs of datasets $[D_1, D_2]$:

> $Rel(D_1, D_2)$: Related pairs of datasets describe similar real-world objects or concepts from the same domain of interest. These datasets store similar information in (some of) their attributes. Typically, the information contained in such attributes partially overlap. An example would be a pair of datasets describing different human diseases, like one for diabetes patients and another for hypertension patients. The datasets will have similar attributes (partially) overlapping their information like the patient's age, gender, and some common lab tests like blood samples.

> $Dup(D_1, D_2)$: Duplicate pairs of datasets describe the same concepts. They convey the same information in *most* of their attributes, but such information can be stored using differences in data. For example, two attributes can describe the weight of an object but one is normalised between 0 and 1 and the other holds the raw data in kilograms. Both attributes are identified to be representing similar information although their data are not identical.

Examples. We scrutinise the relationship between two pairs of datasets in Figure 3.1. Each dataset has a set of attributes. An arrow links similar attributes between two datasets. For example, attributes 'A1' from D_2 and D_3 are nominal attributes with two unique values, making them similar. A numeric attribute like 'A2' in D_2 holds similar data as attributes 'A3' and 'A4' from D_3, as expressed by the intersecting numeric ranges. In our approach we extract meta-features from the datasets (for this example, the number of distinct values and means respectively) to assess the similarity between attributes of a given pair of datasets. The Rel and Dup properties are then used to express datasets similarities. For example, $Dup(D_1, D_2)$ returns '1' because they have similar information in most attributes (even though 'A5' and 'A3' do not match). Based on these two properties, our proposed approach will indicate whether two datasets are possibly related (e.g., $Rel(D_2, D_3)$ ='1') and should be considered for further scrutinising by schema matching, or if they are possibly duplicated (e.g., $Dup(D_1, D_2)$ ='1') and should be considered for deduplication efforts.

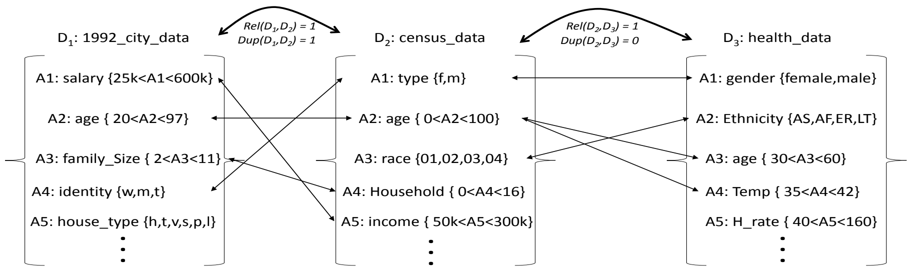


Figure 3.1: Similarity relationships between two pairs of datasets

Scenarios. We aim at governing the DL by maintaining the *Rel* and *Dup* relationships between the datasets it contains. We consider two typical scenarios. In ***scenario (a)***, we want to dredge a data swamp which we don't know any relationships for, thus, for all pairs in the DL we need to find if they are related or duplicated. In ***scenario (b)***, we have an existing DL for which we know all relationships between the datasets. However, given the dynamic nature of DLs new datasets are frequently ingested. Thus, we need to compare this dataset against the datasets already in the DL to find its relationships with them.

3 Related Work

As described in [132], metadata describing the information stored in datasets need to be collected to effectively govern BD repositories. Such metadata are usually automatically collected across multiple datasets using data profiling techniques like schema matching [98], which seeks to identify schematic overlaps between datasets. This involves detecting related objects (instances or attributes) and matching instances between two different schemata [24]. The main line of research in this field is focused around improving the efficiency of matching techniques for two very large schemata. In our research, however, we focus on matching attributes between *multiple large amounts of schemata*, closely related to the field of holistic schema matching [24, 110]. A more restrictive case of schema matching involves *deduplication* [110]; finding highly overlapping instances [39]. Similar to our special requirements for schema matching, we also seek to detect *duplicated schemata* instead of instances. This is when schemata have similar overlapping attributes, not necessarily the same instances.

As described in [24], it is recommended to utilise early-pruning mechanisms for holistic schema matching, which filters out unnecessary matching efforts using less complex techniques. This is commonly done using similarity search techniques which seek to eliminate unnecessary comparisons of datasets [103]. Several techniques for *instance-based matching* were proposed including techniques like clustering [20, 39, 48], hashing [103], and indexing [81, 103]. Alternatively, we propose to focus on *attribute-based* matching across multiple-schemata for governing the DL which needs new and efficient techniques. This field was not sufficiently studied before, with only preliminary results in [125]. We propose a new approach utilising a novel technique of computationally cheaper meta-features proximity comparisons. We seek to prevent unnecessary and expensive schema matching computations in further steps. We propose a machine learning approach for early-pruning that is based on metadata collected from datasets. Such learning techniques were proposed for future research in similarity search [26] where they use a supervised machine learning model based on SVM to find similar strings for deduplication. [26] shows that using machine learning leads to more accurate similarity search from different domains of knowledge.

4 The DS-Prox Approach

We propose a proximity computation based on overall meta-features extracted from the datasets, which we call DS-Prox. We are seeking to have approximate similarity comparisons of pairs of datasets for the early-pruning task. Here we apply cheap computation steps for the overall similarity search, to prevent further expensive detailed analysis of the content of datasets which are estimated to be dissimilar. Similar to our previous work in [15] (see Chapter 2), we seek to profile the datasets ingested in the DL by extracting some *meta-features* describing the overall content and attributes in the datasets. We compute distances between each of the meta-features as proximity metrics. We take a sample of pairs of datasets which are analysed by a data analyst and annotated whether they hold *related* or *duplicate* data by means of the Rel and Dup properties. Rel and Dup are boolean functions retrieving either 1 (similar/duplicate respectively) or 0 (dissimilar/not duplicate). We then use machine learning techniques over the proximity metrics to create two independent models which can classify pairs of datasets according to $Rel(D_1, D_2)$ and $Dup(D_1, D_2)$ respectively. The classification models are used to *score* pairs of datasets with a similarity measure $Sim(D_1, D_2)$. The similarity score 'Sim' is defined independently for each of the relationships $Rel(D_1, D_2)$

and $Dup(D_1, D_2)$ as a number between 0 and 1, where 0 means dissimilar and 1 means most similar: $Sim(D_1, D_2) \in [0, 1]$.

4.1 The Meta-Features Distance Measures

Table 3.1: DS-Prox meta-features

Type	Meta-feature	Description
General	Number of Instances	The number of instances in the dataset
	Number of Attributes	The number of attributes in the dataset
	Dimensionality	The ratio of number of attributes to number of instances
Attributes by Type	Number per Type	The number of attributes per type (Nominal or Numerical)
	Percentage per Type	The percentage of attributes per type (Nominal or Numerical)
Nominal Attributes	Average Number of Values	The average number of distinct values per nominal attribute
	Standard Deviation of Number of Values	The standard deviation in the number of distinct values per nominal attribute
	Minimum/Maximum Number of Values	The minimum and maximum number of distinct values per nominal attribute
Numeric Attributes	Average Numeric Mean	The average of the means of all numeric attributes
	Standard Deviation of the Numeric Mean	The standard deviation of the means of the numeric attributes
	Minimum/Maximum Numeric Mean	The minimum and maximum mean of numeric attributes
Missing Values	Missing Attribute Count	The number of attributes with missing values
	Missing Attribute Percentage	The percentage of attributes with missing values
	Minimum/Maximum Number of Missing Values	The minimum and maximum number of instances with missing values per attribute
	Minimum/Maximum Missing Values Percentage	The minimum and maximum percentage of instances with missing values per attribute
	Mean Number of Missing Values	The mean number of missing values from each attribute
	Mean Percentage of Missing Values	The mean percentage of missing values from each attribute

For each dataset in the DL, we extract meta-features using data profiling techniques. This includes general statistics about the dataset and its attributes as described in Table 3.1. Our purpose for those meta-features is to describe the general structure and content of the datasets for an approximate comparison using our proximity metric and classification models. We compute distances for each meta-feature m_i from Table 3.1 between each pair of datasets $[D_1, D_2]$ using equation 3.1 which gives the relative difference as a number between 0 and 1. Those distances we feed to the supervised machine learning algorithm in our approach.

$$dist_{m_i}(D_1, D_2) = \frac{\max\{m_i(D_1), m_i(D_2)\} - \min\{m_i(D_1), m_i(D_2)\}}{\max\{m_i(D_1), m_i(D_2)\}} \tag{3.1}$$

4.2 The Approach

The approach proposed for early-pruning depends on classical machine learning which is divided into two phases: *Supervised Learning* Phase and *Scoring and Classification* Phase. In the first phase, which can be seen in Figure 3.2, we build a classification model for each of the properties *Rel* and *Dup* using supervised learning techniques. First, for each dataset we extract its meta-features from Table 3.1 which returns its data profile (In Figure 3.2 we see a sample of two meta-features: number of attributes 'nAttr', and number of instances 'nIns'). Then, for each dataset, we generate all pairs with each of the other datasets and compute the distances between their meta-features using Equation 3.1. We also present the pairs of datasets to a human-annotator who manually decides whether they satisfy (assign '1') or not satisfy (assign '0') $Rel(D_1, D_2)$ and $Dup(D_1, D_2)$. Any pair annotated as a match for $Dup(D_1, D_2)$ must also be annotated as a match for $Rel(D_1, D_2)$ (i.e., all duplicate pairs of datasets are also related). We feed both the annotated pairs of datasets with their distances as training examples to a learner which creates two classifiers: M_{rel} and M_{dup}.

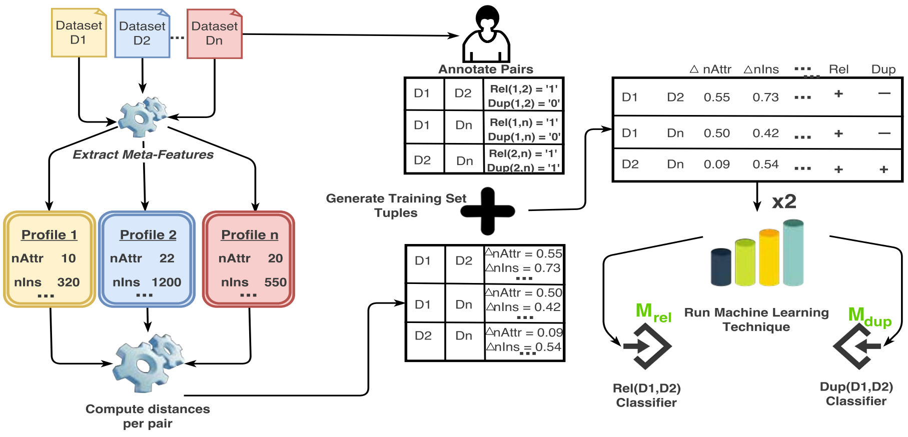

Dataset D1
Dataset D2
Dataset Dn
Annotate Pairs
Extract Meta-Features
Profile 1 nAttr 10 nIns 320
Profile 2 nAttr 22 nIns 1200
Profile n nAttr 20 nIns 550
Compute distances per pair
Generate Training Set Tuples
D1 D2 Rel(1,2) = '1' Dup(1,2) = '0'
D1 Dn Rel(1,n) = '1' Dup(1,n) = '0'
D2 Dn Rel(2,n) = '1' Dup(2,n) = '1'
D1 D2 △nAttr = 0.55 △nIns = 0.73
D1 Dn △nAttr = 0.50 △nIns = 0.42
D2 Dn △nAttr = 0.09 △nIns = 0.54
△ nAttr △nIns ... Rel Dup
D1 D2 0.55 0.73 ... + −
D1 Dn 0.50 0.42 ... + −
D2 Dn 0.09 0.54 ... + +
x2
Run Machine Learning Technique
M_rel
M_dup
Rel(D1,D2) Classifier
Dup(D1,D2) Classifier

Figure 3.2: DS-Prox: supervised machine learning

In the second phase, we apply the classifiers to the scenarios discussed in Section 2, to score each new pair of previously unseen datasets. In ***scenario (a)***, we have a setting where there are two DLs. DL_1 has a group of datasets which have previously known annotations of all their $Rel(D_1, D_2)$ and $Dup(D_1, D_2)$ relationships between all pairs of datasets. On the other hand, DL_2 is without any annotations of such relationships and is therefore a data swamp we would like to dredge. Therefore, we need to learn the models for $Rel(D_1, D_2)$ and $Dup(D_1, D_2)$ from DL_1 and apply them to DL_2 which has different datasets. In ***scenario (b)***, we have an existing DL for which we know all relationships between the datasets. We need to deal with a new dataset as it arrives in this DL. We learn the models from the DL, and we apply them to each new dataset D_i ingested within the same DL. The models should identify all datasets in the DL which are related or duplicate of D_i.

When applying the classifiers, we compute for each pair of datasets the similarity score of $Sim_{rel}(D_1, D_2)$ and $Sim_{dup}(D_1, D_2)$ using the classifiers extracted in the previous phase. The *Sim* score is the positive-class distribution value generated by each classifier. The predicted distribution-value achieved for the 'true' class from each classifier is checked against a minimum threshold to indicate whether the pair of datasets are overall related or duplicates. In our approach, pairs of datasets are evaluated first if they match the $Dup(D_1, D_2)$ relationship (indicating that it also matches $Rel(D_1, D_2)$). If it fails this duplicate test, then we evaluate if the pair still satisfies $Rel(D_1, D_2)$. The output classifiers can classify in the future any new pairs of datasets as either related or duplicate according to two matching approaches: *1-to-1 matching* or *cluster matching*.

1-to-1 matching: all pairs satisfying $Rel(D_1, D_2)$ and $Dup(D_1, D_2)$ need to be selected for further schema matching and deduplication. The calculations are performed under the assumption that each and every pair of matching datasets should be correctly identified using our models.

Cluster-based matching: It is common to use clustering based approaches for the matching process [20, 24, 48]. Groups of datasets with close proximity are segmented into clusters. In our case, the relationship *Rel* can be used to cluster the datasets in the DL, after all relationships are discovered. We therefore relax our requirements for the second phase so that a new dataset should match with *any single* dataset in the same cluster in order to consider it a positive match. Therefore, if a dataset matches one or more dataset(s) from a cluster, we consider all pairs of datasets in this cluster as positively matching pairs (even if the classifier did not indicate a positive match for some of those pairs separately). The rationale behind this approach is that in a real holistic schema matching setting, a new dataset ingested should be compared to *all* the datasets in a cluster it matches to. Clustering can take place after schema matching identifies the relationships between datasets (which is outside the scope of this chapter, but the reader can refer to [20, 48] for such clustering in instance-based matching).

We illustrate our general approach with a toy example in Figure 3.3. Suppose we have two meta-features $nIns$ and $nAttr$ for each dataset. To classify a pair $[(nIns_1, nAttr_1), (nIns_2, nAttr_2)]$ we compute the relative differences. In Figure 3.3(a) we have plotted $(\Delta nIns, \Delta nAttr)$ for all pairs in the training data. '+' indicates a matching pair, '-' a non-matching pair. Based on this data we learn a classifier, for instance a separating hyperplane as shown in Figure 3.3(a) by the red line. Here, for simplification, we show pairs of datasets plotted based on the distances of only two meta-features ($nIns$ and $nAttr$). The actual approach would consider all meta-features in Table 3.1.

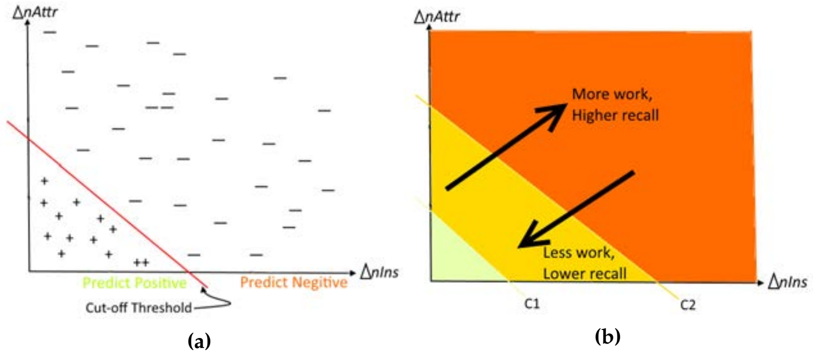


Figure 3.3: DS-Prox cut-off thresholds tuning

Most classification models produce a score instead of a binary output. In the example of the separating hyperplane the obtained distance to the hyperplane can be used as a score. This score can be compared against different cut-off thresholds to decide on the final classification '+' or '-'. The threshold can be chosen to lead to different results, as seen in Figure 3.3(b). If we choose the cut-off threshold 'C1' we restrict the classifier to return less pairs of high proximity (i.e., low distance), leading to lower recall but less work. Alternatively, if we alter the cut-off threshold to 'C2', we relax the classifier to return pairs of lower proximity. This leads to more pairs (i.e., more work) returned by the classifier as positive matches and higher recall of positive cases, but, with more pairs marked incorrectly as matching. Therefore, the cut-off threshold can be tweaked by the data scientist according to practical requirements in order to increase recall at the expense of more work or vice versa. This is the trade-off which we seek to optimise in our experiments when selecting different thresholds. We can use different thresholds 'c_{rel}' and 'c_{dup}' for each of the classifiers evaluated. This means that we consider a positive match if the classifier scores a new pair of datasets with a score greater than the threshold as in Equations (3.2) and (3.3).

$$Rel(D_1, D_2) = \begin{cases} 1, & Sim_{rel}(D_1, D_2) > c_{rel} \\ 0, & \text{otherwise} \end{cases} \tag{3.2}$$

$$Dup(D_1, D_2) = \begin{cases} 1, & Sim_{dup}(D1, D_2) > c_{dup} \\ 0, & \text{otherwise} \end{cases} \tag{3.3}$$

The complexity of our approach is quadratic in the number of datasets, however, it applies the cheapest computational steps for early-pruning (just computing distances in Equation 3.1 and the classifier scoring model on each pair). This way, we save unnecessary expensive schema matching processing in later steps.

5 Experimental Evaluation

We tested an implementation of the DS-Prox approach on OpenML[1], which can be considered an online DL. It consists of different datasets covering heterogeneous topics, each having a name and a description.

5.1 Datasets

The main challenge is to create the ground-truth which we use to evaluate our approach. To achieve this, we created an experimental environment where we extracted the following independent sets of datasets from OpenML:

- **Restricted-topics sample**: First, we extract some datasets by topic using 11 keywords-search over OpenML, e.g., “Disease", “Cars",“Flights", “Sports", etc. This restricted sample consists of 130 datasets and we consider them to be similar if they belong to the same topic.
- **All-topics sample**: This is an independent set of other datasets collected from OpenML. To collect this sample, we scraped the OpenML repository to extract all datasets not included in the restricted-topics sample and having a description of more than 500 characters. Out of the 514 datasets retrieved we selected 213 with descriptive descriptions (i.e., excluding datasets whose descriptions do not allow to interpret its content and to assign a topic).

Therefore, we created two new groups of datasets from OpenML for our experiments, each having its own independent set of datasets without any overlap. Having two independent sets strengthens our results and allows us to generalise our conclusions. A domain expert and one of the authors collaborated to manually label the pairs of datasets with the same topic as duplicated and / or related. The interested reader can download the two annotated datasets from GitHub[2]. The details of each sample is summarised in Table 3.2, which lists the number of datasets, the number of topics, top topics by the number of datasets, and the number of related and duplicated pairs per sample.

[1]http://www.openml.org
[2]https://github.com/AymanUPC/datasets_proximity_openml

Table 3.2: A description of the OpenML samples collected

Sample	Datasets	Topics	Top Topics	$\mathbf{Rel}(\mathbf{D_1}, \mathbf{D_2})$	$\mathbf{Dup}(\mathbf{D_1}, \mathbf{D_2})$
Restricted-topics	130	29	Diseases (45), Health (31), Cars (13), Academic Courses (6), Sports (5)	1205	72
All-topics	213	79	computer software defects (17), citizens census data (12), digit handwriting recognition (12), Diseases (11)	570	128

Table 3.3: An example of pairs of datasets from the all-topics sample from OpenML

No.	DID 1	Dataset 1	DID 2	Dataset 2	Topic	Relationship
1	23	cmc	179	adult	Census Data	related
2	14	mfeat-fourier	1038	gina_agnostic	Digit Handwriting Recognition	related
3	55	hepatitis	171	primary-tumor	Disease	related
4	189	kin8nm	308	puma32H	Robot Motion Sensing	duplicate
5	1514	micro-mass	1515	micro-mass	Mass Spectrometry Data	duplicate

Some of the pairs from the all-topics sample can be seen in Table 3.3. Dataset with ID 23 should match all datasets falling under the topic of 'census data' like dataset 179. Both datasets have data about citizens from a population census. In rows 4 and 5 we can see examples of duplicated datasets, which have highly intersecting data in their attributes. Duplicate pairs in row 4 have the same number of instances, but described with different number of attributes, which are overlapping. The duplicate pairs in row 5 have identical number of attributes, yet, the attributes are transformed using pre-processing techniques and there are different number of instances between both datasets, so in essence the second dataset is a transformed and cleaned version of the first. We aim to detect such kind of scenarios using our DS-Prox approach.

5.2 Experimental Setup

In order to evaluate our approach, we create an experimental setup where we have two sets of datasets for each experiment: 1. Training set and 2. Test set. The training set is used in the supervised learning phase to create the classification models. The classification models are then evaluated using the test set. We use the restricted-topics sample as a training set, and we use both the restricted-topics and the all-topics samples in the scoring phase as test sets to evaluate our approach. We describe how we used those samples to create the training and test sets within our experiments for the two scenarios from Section 2:

- **Scenario (a)** from Section 2: We evaluate our approach by using the restricted-topics sample as the training set and the all-topics sample as the test set. In this case the testing set is an independent collection of datasets. We evaluate both of the 1-to-1 matching and cluster matching approaches for $Rel(D_1, D_2)$. We also evaluate the 1-to-1 matching with $Dup(D_1, D_2)$.

- **Scenario (b)** from Section 2: We evaluate our approach using a leave-one-out (LOO) variant evaluation method and the restricted-topics sample. Here we remove a dataset and all its pairs from the original training set and we use those pairs for evaluation of the output classifiers as a separate test set. We also remove all duplicate pairs of this dataset from the training set to guarantee independence between the training and evaluation environments. We repeat this for every dataset in the input training set. We use the 1-to-1 matching approach in our evaluation.

To execute our experiments, we profile the datasets to extract their meta-features. We use the training set of annotated datasets with the WEKA[3] tool to create the classification models using different supervised techniques: **Bayesian** (Bayesian Network with K2 search, Naïve Bayes) , **Regression** (LogitBoost) , **Support Vector Machines** (Sequential Minimal Optimization) , and **Decision Trees** (Random Forest). We also use **Ensemble Learners** [45]: AdaBoost (with Decision Stump classifier), Classification Via Regression (with M5 Tree classifier), and Random Subspace (with Regression Tree classifier). We tested different techniques because it was suggested by [45] that some individual techniques can outperform the ensemble learners in classification problems. We evaluate the classifiers with 10 different cut-off thresholds for 'c_{rel}' and 'c_{dup}' from Equations (3.2) and (3.3), in order to cover a wide range of values. We benchmark the techniques against the decision table technique [72] which simply assigns the majority class based on matching the features to a table of learned examples.

5.3 Results

Table 3.4: A description of the experiments conducted

Experiment	Graphs in Figure3.4	Matching Approach	Scenario	Relationship
1	row 1: (a) and (b)	1-To-1	(b)	$Rel(D_1, D_2)$
2	row 2: (c) and (d)	1-To-1	(a)	$Rel(D_1, D_2)$
3	row 3: (e) and (f)	Cluster-based	(a)	$Rel(D_1, D_2)$
4	row 4: (g) and (h)	1-To-1	(a)	$Dup(D_1, D_2)$

We evaluate the effectiveness of our approach using the recall, precision, and efficiency-gain measurements, as described in Equations (3.4),(3.5) and (3.6) respectively. Here, TP means true-positives which are the pairs of datasets correctly classified by the classifier. FN are false negatives, FP are false-positives, TN are true-negatives, and N indicates the total number

[3]https://weka.wikispaces.com/Use+WEKA+in+your+Java+code

of possible pairs of datasets. The efficiency gain measures the amount of reduction in work required, in terms of number of pairs of datasets eliminated by the classifier.

$$recall = \frac{TP}{TP + FN} \quad (3.4) \qquad precision = \frac{TP}{TP + FP} \quad (3.5) \qquad efficiency - gain = \frac{TN + FN}{N} \quad (3.6)$$

We change the cut-off thresholds, and we aim to maximize this as much as possible while maintaining the highest recall possible. The effectiveness of our approach is evaluated by recall and precision. By applying our approach with the different scenarios and relationships, we conduct 4 sets of experiments as in Table 3.4. The results are depicted in the graphs in Figure 3.4.

The measures shown in the graphs are all averages from all datasets involved in the test sets for a specific data mining technique and a certain cut-off threshold for the proximity score (darker points have higher cut-off values). The common measure for all graphs, which is the recall plotted on the y-axis, is highlighted by having some of its main values labelled on each graph. Graphs (a) and (b) are the same graphs as (c) and (d) respectively but for the 1-to-1 matching approach applied with the different scenarios. We select for the experiments certain target results, which are minimum expected values for each measure. All area above those values are shaded as follows: **Min. recall**: 0.75 for 1-to-1 matching & 0.9 for cluster matching, **Min. efficiency gain**: 0.33 for 1-to-1 matching & 0.25 for cluster matching, **Min. precision**: 0.25 for all approaches. This means that we were targeting at least 75% recall rate for 1-to-1 matching and 90% recall rate for the cluster based matching (which improves our previous results in [15] described in Chapter 2). We aimed for at least 25% efficiency gains with the cluster matching approach, which exceeds those achieved in [20]. However, we acknowledge that their approach applied to instances-matching within the same dataset, not cross-schema attribute-matching as in our case. For experiment 4 for $Dup(D_1, D_2)$, we aim for a min. recall of 0.9, min. efficiency gain of 0.75, and min. precision of 0.33. In real-world applications, the data scientist can choose different minimum thresholds for each measure according to practical requirements.

5.4 Discussion

General trend. From the results depicted in Figure 3.4, the optimum technique and cut-off threshold is the one in the top-right quadrant of each graph, optimising both measures plotted. The recall-precision and recall-efficiency plots follow the general trend expected which indicate the trade-off between both measures in each plot, yet, more optimised solutions are possible for balancing recall-efficiency, as seen by the classifiers performing in the top-right quadrant. As the cut-off thresholds increase, there is a drop in recall against an increasing efficiency gain. Still, the top mining techniques and thresholds can be used to achieve high efficiency gain and recall. This is discussed for each property below. As the precision rates are generally low, we conclude that our approach can only be used as an early-pruning step, and should be followed by other more expensive and more detailed matching steps. Yet, a good compromise can still achieve high recall and efficiency gains; efficiency gains up to 0.5 for *Rel* and 0.8 for *Dup*. Such efficiency gains can make an important difference for computationally-expensive applications of holistic schema matching in the DL environment.

Rel evaluation. The recall-efficiency plots indicated that it was possible to achieve an optimum technique and threshold in the top-right quadrant, which represent the compromise of not sharply losing recall with higher efficiency gain. For example, from Figure 3.4 (e) for experiment 3, using the AdaBoost technique at a threshold of 0.5 can lead to 0.42 efficiency gains while still maintaining 0.95 recall. If a recall of 1.0 is required, then this can be achieved by the cut-off threshold of 0.3 for the same technique, but only 0.13 efficiency gain is achieved. The data scientist will have to decide if this efficiency gain is sufficient and whether a recall rate of 100% is critical in their application, else, a 0.05 drop in recall should be allowed to achieve much higher efficiency gain using the techniques and thresholds in the top quadrant. For the 1-to-1 matching in Figure 3.4 (c), we can achieve 0.75 recall and 0.35 efficiency gain. There is a drop in recall, as would be expected, because the classifier has more challenges in matching all possible 'related' datasets, while in the cluster matching approach, a single match to a dataset in a cluster acts like a pivot which results in matching all the required related datasets in the same cluster. The cluster-matching approach shows an improved performance over the 1-to-1 matching approach, therefore it is recommended to use DS-Prox with the clustering-based approach.

Dup evaluation. For the results in Figure 3.4 (g) and (h), the top performing techniques were Random subspace and Random Forest at 0.2 cut-off thresholds. This achieved about 0.97 recall and 0.76-0.8 efficiency gain. The baseline method was not able to differentiate at different cut-off thresholds, and had best recall of 0.65, except for the lowest cut-off of 0.1 where it achieved a jump to 0.94 recall. Since the recall was very high for our target efficiency gain using the 1-to-1 approach, the cluster-based approach did not yield any better results.

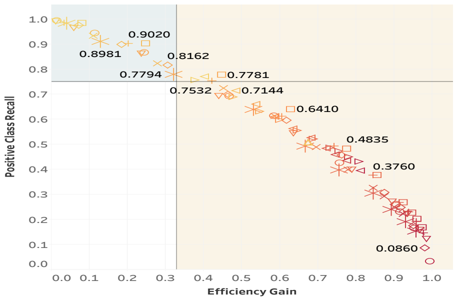
Baseline comparisons. Different techniques can yield better results than the baseline for several of our experiments. There is not one single technique which is best, yet, ensemble learners tend to perform better than their counterparts. However, simple techniques like logistic regression and Naïve Bayes can still have good performance as seen in the graph (e) top-right quadrant. The baseline technique was never in the top-quadrant of graph (e) and many techniques outperformed it. In the 1-to-1 matching in graphs (a) and (c), the baseline classifier was comparable with the other techniques. The top techniques include the iterative optimiser in graph (c) with 0.75 recall and

0.35 efficiency gain. Nearly 1.0 recall was possible using the same classifier at a lower threshold, yet with only 0.09 efficiency gain. For experiment 1, Random Forest and Random Subspace outperformed the baseline with 0.3 cut-off thresholds.
Generalizability. Although our approach is generic and does not apply to a specific domain only, we note that we do not claim that the classifiers for one type of data or of a certain domain will have the same guaranteed effectiveness when applied in another setting. The approach might need to be adjusted and retrained within other settings. Albeit, our results from experiments 2 and 3 show a positive indicator of the possibility to train the model on specific domains, independent of those used in the test set (or real-world setting), and still be effective. We think that this needs further experimentation in the future.

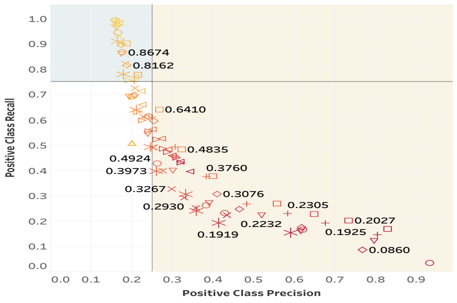
6 Conclusion and Future Work

This chapter presented a novel approach of similarity search within a DL based on a proximity mining technique for early-pruning in holistic dataset schema matching and deduplication applications. The approach uses supervised machine learning techniques based on meta-features describing semi-structured datasets. Experiments on a real-life DL demonstrate the effectiveness in achieving high recall rates and efficiency gains. Proposed techniques support data governance in the DL by identifying relationships between datasets. The drawback of our approach, however, is that it needs some manual effort to annotate training examples for the classifiers. In the future, we will test the generalizability of applying the same classifier to different data sources. We plan to experiment with more detailed meta-features which might lead to improved results. We will also test our approach on other kinds of semi-structured data (like RDF or XML).

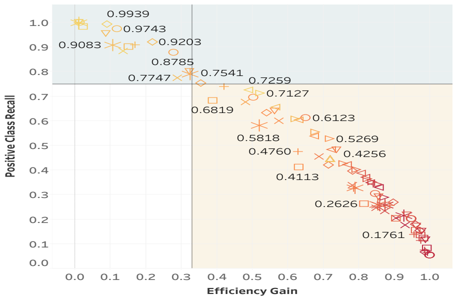
Acknowledgements. This research has been funded by the European Commission through the Erasmus Mundus Joint Doctorate (IT4BI-DC).



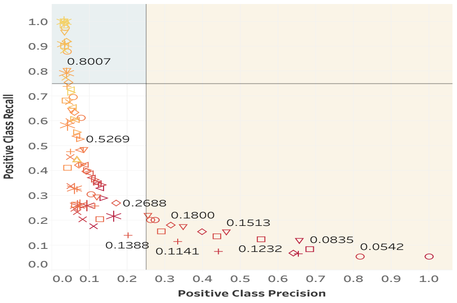
(a)



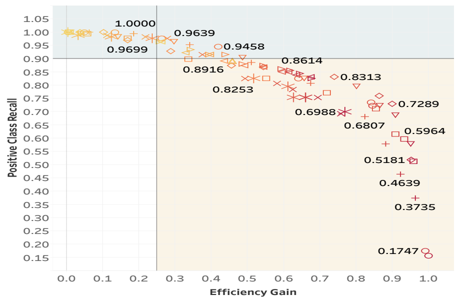
(b)



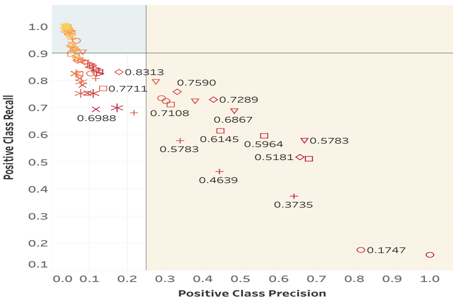
(c)



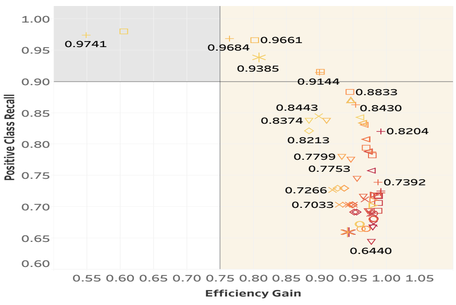
(d)



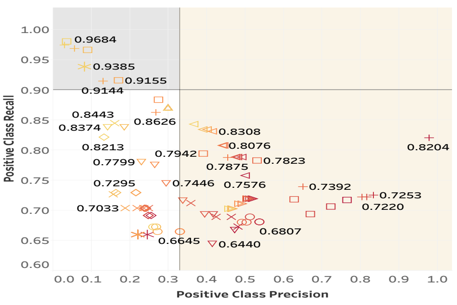
(e)



(f)



(g)



(h)

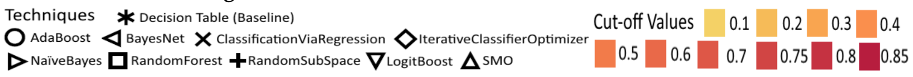


Figure 3.4: Recall-efficiency plots (left column) and recall-precision plots (right column) for experiments 1,2,3 and 4 in each row

Chapter 4

Attribute-level content metadata based proximity mining for pre-filtering schema matching

Everything is continuously changing and to be successful, one must embrace change and swiftly adapt to it. Improvement is a lifetime journey.

This chapter has been published as a paper in the ACM Transactions on Information Systems (TOIS) (2020) [17].
The layout of the paper has been revised
DOI: https://doi.org/10.1145/3388870

ACM copyright / credit notice:
© 2020 ACM. Reprinted with permission from Ayman Alserafi, Alberto Abelló, Oscar Romero, Toon Calders. Keeping the Data Lake in Form: Proximity Mining for Pre-Filtering Schema Matching, TOIS, 2020.

Abstract

Data Lakes (DLs) are large repositories of raw datasets from disparate sources. As more datasets are ingested into a DL, there is an increasing need for efficient techniques to profile them and to detect the relationships among their schemata, commonly known as holistic schema matching. Schema matching detects similarity between the information stored in the datasets to support information discovery and retrieval. Currently, this is computationally expensive with the volume of state-of-the-art DLs. To handle this challenge, we propose a novel early-pruning approach to improve efficiency, where we collect different types of content metadata and schema metadata about the datasets, and then use this metadata in early-pruning steps to pre-filter the schema matching comparisons. This involves computing proximities between datasets based on their metadata, discovering their relationships based on overall proximities and proposing similar dataset pairs for schema matching. We improve the effectiveness of this task by introducing a supervised mining approach for effectively detecting similar datasets which are proposed for further schema matching. We conduct extensive experiments on a real-world DL which proves the success of our approach in effectively detecting similar datasets for schema matching, with recall rates of more than 85% *and efficiency improvements above* 70%*. We empirically show the computational cost saving in space and time by applying our approach in comparison to instance-based schema matching techniques.*

1 Introduction

Today, it is more and more common for data scientists to use Data Lakes (DLs) to store heterogeneous datasets coming from different sources in their raw format [128]. Such data repositories support the new era of data analytics where datasets are ingested in large amounts and are required to be analysed just-in-time [82]. However, it is a challenge for data wranglers [51, 69, 128] preparing the datasets for analysis to understand their structure and *commonalities* for DL governance purposes [15]. They must extract those datasets which have related data to be used together in data analysis tasks [82, 89]. This is commonly referred to as *schema matching*, where the aim is to find connection strengths between similar concepts from different pairs of datasets [27, 35, 83, 104]. When applied on large-scale heterogeneous repositories like DLs, it becomes a case of *holistic schema matching* [18, 89, 104, 110].

> **Holistic schema matching**: The large-scale application of schema matching to BD repositories like DLs, where the goal is to match multiple datasets together considering them all in the matching task.

We focus on DLs having datasets storing data in flat tabular formats. Flat datasets are organised as attributes and instances, such as tabular data, comma separated values (CSV) files, hypertext markup language (HTML) tables, etc. (see Section 3). It is a challenge with such DLs to efficiently process the datasets to detect their common features, as schema matching tasks are generally expensive (involving huge amounts of string comparisons and mathematical calculations) [24, 27, 71]. In this chapter, we propose novel techniques to reduce those comparisons using *pre-filtering* techniques that generate less comparisons. To illustrate this, consider the different types of comparisons in Figure 4.1. Traditionally, schema matching makes many comparisons of data values of instances from different datasets (see the instance-based matching box). With the rise of DLs, previous research [18, 43, 24] recommended using early-pruning steps to facilitate the task using *schema matching pre-filtering*.

> **Schema matching pre-filtering** means that only dataset pairs detected to be of relevant *similarity* are recommended for further fine-grained schema matching tasks, and dissimilar ones are filtered out from further comparisons. This supports the goal of early-pruning of the number of required comparisons in order to improve the efficiency of schema matching in large-scale environments like DLs.

Figure 4.1 reflects this stratified approach that filters by means of extracted metadata at the dataset and attribute level before going for expensive instance-based approaches. For example, consider a DL with 1000 datasets, with 15 attributes and 1000 instances each. Since schema matching techniques generate $\frac{n*(n-1)}{2}$ comparisons, it would result in 499500 comparisons at the dataset level, 113 million at the attribute level and about 500 billion at the instance level. Clearly, fine-grained comparisons do not scale and pre-filtering is necessary.

We propose a pre-filtering approach based on different levels of granularity, at which we collect metadata using data profiling techniques. Our approach collects metadata at two different levels: at the dataset and attribute level. This metadata are then used in a supervised learning model to estimate *proximity* among pairs of datasets. Such proximity is then used for pruning out pairs *less likely* to be related. This is illustrated in Figure 4.1, where our goal is to filter candidate pairs of datasets before conducting computationally intensive instance-based schema matching techniques. The scope of this chapter is the early-pruning at the top two granularity tiers. We refer the interested reader to previous research about classical instance-based schema matching which is outside the scope of this chapter [24, 27, 83, 111, 121].

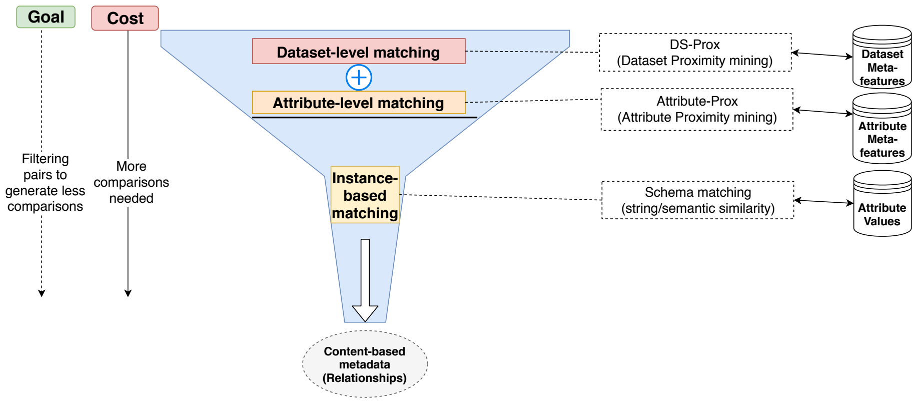

Goal
Cost
Filtering pairs to generate less comparisons
More comparisons needed
Dataset-level matching
Attribute-level matching
Instance-based matching
Content-based metadata (Relationships)
DS-Prox (Dataset Proximity mining)
Attribute-Prox (Attribute Proximity mining)
Schema matching (string/semantic similarity)
Dataset Meta-features
Attribute Meta-features
Attribute Values

Figure 4.1: The stratified holistic schema matching approach at different levels of granularity.

It is a challenge to compute dataset similarities for pre-filtering tasks due to the difficulty of finding adequate similarity metrics and features to use [18]. This chapter is an extension of our previous work in [18] and Chapter 3, and it presents a novel proximity[1] mining approach (see Section 4) between datasets. The similarity functions are based on automatically extracted metadata and can be effectively used for pre-filtering in a stratified approach (see the DS-Prox and Attribute-Prox boxes attached to the matching steps in Figure 4.1).

[1]In this chapter, we use proximity and similarity interchangeably.

To our knowledge, no other approach uses automatically extracted metadata for this purpose.

To show the feasibility of our approach, we assess its performance by using it on a real-world DL. Our approach was able to filter the number of fine-grained comparisons by about 75% while maintaining a recall rate of at least 85% after filtration. Our early-pruning approach also saves computational costs in terms of space and time requirements by at least 2 orders of magnitude compared to instance-based matching.

Contributions. We present an approach for pre-filtering schema matching tasks. We propose techniques for detecting similar schemata based on metadata at different levels of granularity. This supports in early-pruning of the raw-data instance-based schema matching tasks. We present an expanded and cross-validated experiment for the DS-Prox technique from our previous work in [18] (see Chapter 3) and comparisons against combining it with our new proposed attribute-level proximity metrics to find the most appropriate metrics to assign similarities between pairs of datasets. We demonstrate a detailed analysis of the different proximity metrics based on different types of meta-features (name-based and content-based). Our improvements outperform our previous work in terms of effectiveness measures like recall and lift-scores.

The chapter is organised as follows: Section 2 presents the related work, Section 3 introduces the main concepts in our research, Section 4 presents our proximity mining based approach for early-pruning tasks of holistic schema matching, Section 5 presents our experimental evaluation, and finally, we conclude in Section 6.

2 Related Work

State-of-the-art schema matching techniques either use schema-level metadata (mainly names and data types of schema components) [35, 36, 100, 104] or instance-level values of data objects [27, 35, 43, 52, 83, 100, 126]. Some others use a hybrid approach utilising schema metadata and data instances [24, 109]. At the schema-level, these techniques usually use the syntactic similarity of the names of the schema components for the matching task. At the instance-level, values are usually compared using Jaccard similarity of intersecting exact values [89]. This can also be achieved by first matching duplicate instances and finding the correspondences between their schema components [27]. Further, these algorithms can be domain-specific or generic [83].

Schema matching is a computationally intensive task that requires large amounts of comparisons [15, 18, 24] because they typically generate a Cartesian product between the values to be compared. Moreover, other approaches also exploit the semantic or linguistic similarity of values, which requires further computations to translate data values (finding synonyms and hypernyms) or to map them to ontologies [83].

The current focus of the schema matching research community is to implement efficient holistic schema matching that improves performance by reducing the number of actual comparisons to conduct [89]. To handle this challenge, multiple techniques were proposed. For example, several approaches use clustering techniques as a pre-filter of datasets to match [9, 22, 104]. Only datasets falling in the same cluster are matched, or datasets from one cluster are only matched against representative datasets from other clusters. This is similar to the concept of "blocking" for record linkage [124], where items having equal data values in some or all of their attributes are placed in the same bucket for comparison. The work in [124] focuses on matching instances of data (rows in tabular data) rather than attributes in schemata (columns in tabular data). We propose in this chapter a supervised learning technique that can classify dataset pairs (schemata) for a decision whether they are related (should be compared) or not related, rather than unsupervised techniques like blocking and clustering. In addition, we tackle the similar schemata search problem (i.e., schema matching) rather than similar records search (i.e., record linkage or entity resolution).

In [104], they cluster the schemata and attributes of datasets based on TF-IDF similarity scores of their textual descriptions. In [9], they exploit the structural properties of semi-structured XML documents, i.e., data elements embeddings and hierarchies, to cluster the datasets before applying instance-based schema matching. In [22], they use the textual descriptions and keywords of the datasets to cluster them using TF-IDF and WordNet. In this chapter, we do not consider textual descriptions of datasets, which could be misleading or missing, but rely on metadata that can be automatically extracted from any dataset. Metadata describing datasets, their schemata, and the information stored in them can be collected using data profiling techniques [1, 15, 75, 82, 98]. Different types of metadata can also describe the information inside datasets at different levels of granularity, e.g., overall dataset level [18] (see Chapter 3) or attribute descriptions like we propose in this chapter.

The pre-filtering of dataset pairs which are less-likely to have interrelated data before performing schema-matching is called early-pruning [15, 43, 24], and it was implemented in previous research on semi-structured datasets, like XML, by finding the similarity of hierarchical structures between named data objects [9]. Other works have investigated schema matching with semi-structured datasets like XML [83, 100] and JSON [52]. In the web of data, previous research like [41] investigated recommendation of RDF datasets in the semantic web using pre-defined annotations such as the *sameAs* property. In this chapter, we consider flat datasets without such a hierarchy of embedded data objects and without pre-defined semantic linkages.

To facilitate the early-pruning tasks for schema matching, we can apply the same approaches and concepts from collaborative filtering and adapt them to the holistic schema matching problem [6, 62]. The goal is to use profiling information for comparison and recommendation, which was applied to multimedia in [6] and semi-structured documents in [52]. Content-based metadata was also used to predict schema labels [36]. They use minimum and maximum values for numeric attributes, and exact values for nominal attributes, including the format of values. We propose to apply similar techniques but at the dataset granularity level. Accordingly, we adapt such techniques and find the appropriate similarity metrics for tabular datasets.

Another line of research aims at optimising the schema matching process by using computational improvements [43, 109]. This can be done using partitioning techniques that parallelise the schema matching comparison task [109]. Another approach uses efficient string matching comparisons. Such techniques are very useful in the case when schema components are properly named in the datasets. However, such techniques fail when the components are not properly named (e.g., internal conventionalism, like sequential ID numbering of the components). In [43], they introduce intelligent indexing techniques based on value-based signatures.

Schema matching can also be automated using data mining techniques [35, 36, 52]. In [35], they use hybrid name-based and value-based classification models to match dataset attributes to a mediated integration schema. Their approach is focused on one-to-one mediation between two schemata, while our approach targets all datasets in a DL by holistic schema matching using coarser meta-features. In [36], they use content-based meta-features in multi-value classification models to match schema labels of attributes across datasets. Decision trees were also used to profile semi-structured documents before schema matching [52]. In this chapter, we also use data mining classification models, however the goal differs from [35], [36] and [52] as it tackles the early-pruning and pre-filtering task rather than instance-based schema matching.

We summarise the state-of-the-art in Table 4.1. This table gives a comparison of the most relevant techniques discussed with our approach based on the main features discussed in this section. As a result, we can see that we propose an approach based not only on string matching, but also on content metadata matching involving statistics and profiles of data stored in the datasets. Content metadata are matched based on approximate similarities and not just exact value-matches at the instance-level [82, 89]. We focus on proposing novel early-pruning techniques that use supervised learning to pre-filter irrelevant dataset pairs and to detect likely-to-be similar pairs. Finally, the table shows that our technique makes a novel contribution to the schema matching pre-filtering problem that is not achieved by other state-of-the-art techniques.

Table 4.1: Schema matching techniques state-of-the-art comparison

	COMA++ [109]	**PARIS [126]**	**LOD Data Linking [22]**	**XML Semantic-based Matching [71]**	**Ontology Clustering [9]**	**Proximity Mining [this work]**
Type of Data	Tabular, semi-structured, Semantic OWL	RDF	Semantic RDF	Semi-structured XML	Semi-structured, Semantic OWL	Tabular
Instance-based	✓	✓	✓	✓	×	×
Metadata used	Attribute-level schema names	×	Ontology mappings, RDF schema names, Textual descriptions	Attribute-level schema names and structural metadata	Attribute-level structural metadata	Dataset-level content and name, Attribute-level content and name
Data Mining based	×	×	Clustering	×	Clustering	Supervised learning
Approximate Matching	×	×	✓	×	✓	✓

3 Preliminaries

We consider DLs with datasets having tabular schemas that are structured as attributes and instances like Figure 1.1 as described in Section 2.1. We focus on two types of attributes: ***numeric attributes*** (consisting of real numbers) and ***nominal attributes*** (consisting of discrete categorical values). We differentiate between those two types of attributes, similar to previous research like [35, 36, 100], because we collect different profiling metadata for them. The resulting statistics collected are called *content meta-features*, and are as follows:

- **Nominal attributes:** frequency distributions of their distinct values.
- **Numeric attributes:** aggregated statistical value distributions like mean, min, max, and standard deviations.

For pairs of datasets and attributes, we compute the functions in Table 4.2 and describe them in the rest of this section.

We aim at finding the relationship between a pair of datasets $Rel(D_y, D_z)$. We determine such relationship directly using dataset-level meta-features and by computing the relationships between their pairs of attributes (A_i, A_j), being A_i from D_y and A_j from D_z, as could be seen in Figure 4.2. The figure shows an overview of our proposed proximity mining approach.

The goals of the approach is to use efficient and effective techniques to accurately predict $Rel(D_y, D_z)$ for the pre-filtering task. As could be seen in Figure 4.2, this can be done using metadata and similarity collected at the dataset level (right-side) or by using the attribute level $Sim(A_i, A_j)$ to predict

Table 4.2: Schema matching pre-filtering functions

Relationship	Function Type	Output	Object Type	Description
$Rel(A_i, A_j)$	Binary	$\mathbb{Z} \in \{0,1\}$	Attribute pair	Related attributes storing data about the same real-life concept which contain overlapping information in their values. 1 means positively related and 0 means not.
$Sim(A_i, A_j)$	Continuous	$\mathbb{R} \in [0,1]$	Attribute pair	A value $\mathbb{R}$ to measure the attribute similarity in the range [0,1].
$Rel(D_y, D_z)$	Binary	$\mathbb{Z} \in \{0,1\}$	Dataset pair	Related datasets which contain information about the same real-life object. 1 means positively related and 0 means not.
$Sim(D_y, D_z)$	Continuous	$\mathbb{R} \in [0,1]$	Dataset pair	A value $\mathbb{R}$ to measure the dataset similarity in the range [0,1].

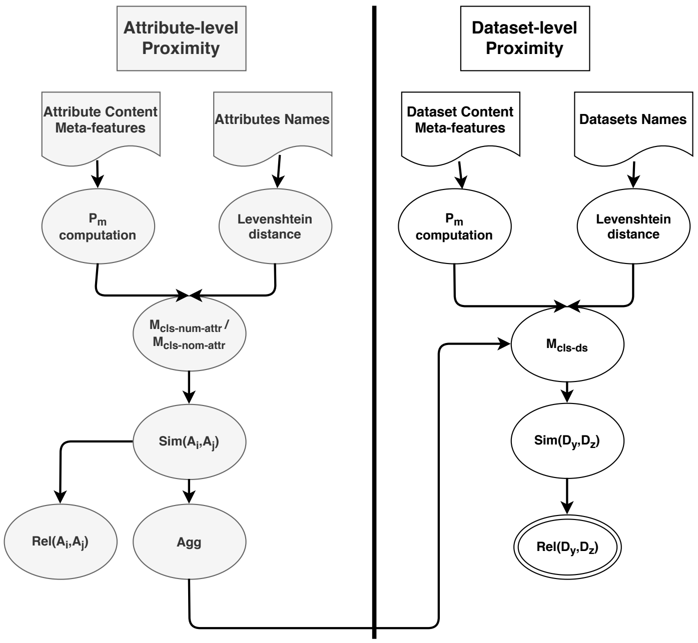


Figure 4.2: The dependencies of components in the metadata-based proximity mining approach for pre-filtering schema matching.

it (left-side). The figure shows the steps required for combining attribute-level similarity with the dataset-level similarity to predict $Rel(D_y, D_z)$. Here, we only use $Sim(A_i, A_j)$ as an auxiliary step that supports us in the main task of predicting $Rel(D_y, D_z)$. This is possible because the attribute metadata are of finer granularity which can be aggregated to a single similarity score at the dataset-level with an aggregation function Agg like averaging $Sim(A_i, A_j)$

scores. When predicting $Rel(D_y, D_z)$, typically the dataset pair will have information contained in some of their attributes which are partially overlapping, satisfying $Rel(A_i, A_j)$, where $\exists A_i \in D_y \wedge A_j \in D_z \implies Rel(A_i, A_j) = 1$. An example would be a pair of datasets describing different human diseases, (e.g., diabetes and hypertension). The datasets will have similar attributes (partially) overlapping their information like the patient's age, gender, and some common lab tests like blood samples.

The *intermediate output* leading to $Rel(A_i, A_j)$ and $Rel(D_y, D_z)$ in our proposed proximity-based approach, seen in Figure 4.2, is a similarity score consisting of a real number in the range of $[0, 1]$, which we indicate using $Sim(A_i, A_j)$ and $Sim(D_y, D_z)$ respectively.

The similarity scores are computed based on proximity models we construct using ensemble supervised learning techniques [127], which we denote as M_{cls-ds} for models handling dataset-level metadata and $M_{cls-num-attr}$ or $M_{cls-nom-attr}$ for models handling attribute-level metadata (depending on the attribute type, numerical or nominal respectively). The models take as input the *distance* between the meta-features describing content of each object pair, whether dataset pair or attribute pair for $Sim(D_y, D_z)$ and $Sim(A_i, A_j)$ respectively, and we call the distance in a specific meta-feature *'m' a proximity metric* which is denoted as $P_m^D(D_y, D_z)$ for dataset pairs or $P_m^A(A_i, A_j)$ for attribute pairs. The names of objects can also be compared using Levenshtein string distance comparison [86] to generate a distance score. The output from the models is a score we compute using the positive class distribution (see Section 4.2).

We convert the intermediate similarity scores to the *final output* consisting of a boolean value for the binary relationships using Equations (4.1) and (4.2). The *sim* score computed for each relationship type is checked against a minimum threshold in the range of $[0, 1]$ to indicate whether the pair involved is overall related '1' or unrelated '0', and therefore whether they should be proposed for expensive schema matching or otherwise filtered out. Using cut-off thresholds of similarity rankings for the collaborative filtering task and schema matching is a common practice [43, 62, 82]. We can use different thresholds 'c_d' and 'c_a' for each of the relationship evaluated at the dataset level and attribute level respectively. This means that we only consider a pair *similar* if their similarity score is greater than the threshold as in Equations (4.1) and (4.2).

$$Rel(D_y, D_z) = \begin{cases} 1, & Sim(D_y, D_z) > c_d \\ 0, & \text{otherwise} \end{cases} \tag{4.1}$$

$$Rel(A_i, A_j) = \begin{cases} 1, & Sim(A_i, A_j) > c_a \\ 0, & \text{otherwise} \end{cases} \tag{4.2}$$

To summarise Figure 4.2, the hierarchy to compute the final output is: *Rel* is based on *Sim* similarity scores, which in turn are based on P_m proximity metrics of meta-features. To convert from P_m to *Sim* we use an ensemble supervised model M_{cls} which takes the P_m proximity metrics as input. The output Sim is compared against a minimum threshold, and those passing the threshold are positive cases for Rel to be considered for further detailed schema matching.

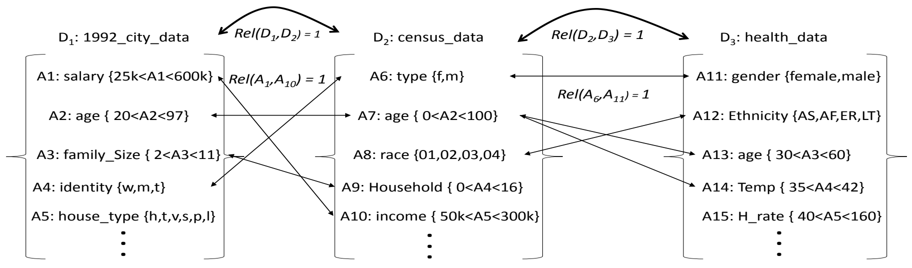


Figure 4.3: Final output of our approach consisting of similarity relationships between two pairs of datasets.

Examples. Consider the relationships between the three datasets in Figure 4.3 which presents the *final output* of our approach. Each dataset has a set of attributes. An arrow links attributes having similar data profiles. We label this as a $Rel(A_i, A_j) = 1$. For example, attributes 'A6' and 'A11' from D_2 and D_3 are nominal attributes with two unique values which we included as a meta-feature called *'number of unique values'*. The proximity metric is the distance (difference) in the meta-feature of number of unique values, which in this case $P_m^A = 0$, because they are identical (i.e., $2 - 2 = 0$), thus making the attribute pair similar (in this case, by their number of distinct values). If we consider this proximity metric of 'number of unique values' alongside other collected content-based meta-features using an *ensemble supervised learning model* M_{cls}, we can compute a $Sim(A_6, A_{11})$ score based on the positive-class distribution (see Section 4.2). This can lead to $Sim(A_6, A_{11}) = 0.95$ and if we use a threshold of $c_a = 0.75$ then the final output for $Rel(A_6, A_{11}) = 1$. A numeric attribute like 'A7' in D_2 holds similar data as attributes 'A13' and 'A14' from D_3, as expressed by the intersecting numeric ranges. For such numeric attributes we can consider a meta-feature like *'mean value'*. On the other hand, attributes 'A1' and 'A7' have different data profiles (different numeric ranges) and therefore are not labelled with an arrow and do not satisfy the $Rel(A_1, A_7)$ relationship, as they will have large differences in their meta-features, leading to high proximity metric and a low similarity scores. In those examples, we collect attribute level meta-features from the datasets (in this case, the number of distinct values for nominal attributes and means for numeric attributes) to assess the similarity between attributes of a given pair of datasets. In our approach, we compute the similarity between attributes $Sim(A_i, A_j)$ using real number proximity metrics in the range of $[0, 1]$ and we use it to predict $Rel(D_y, D_z)$ instead of using the binary output of $Rel(A_i, A_j)$. We should *aggregate* the individual attribute pairs' similarities with an aggregation function *agg* to obtain a single value proximity metric for the overall dataset level similarity. We discuss this in the description of our approach in Section 4.

Furthermore, we extract higher-granularity dataset level meta-features (e.g., 'number of attributes per attribute type') from the datasets for the task of directly computing the $Sim(D_y, D_z)$ similarity relationships. For example, $Rel(D_2, D_3)$ returns '1' in the case we use $c_a = 0.67$ because they have 2 nominal and 3 numeric attributes each, so overall they can have $Sim(D_2, D_3) = 0.7$ passing the minimum threshold. Based on $Rel(D_2, D_3)$ ='1', our approach indicates that these two datasets are possibly related and should be considered for further scrutinising by schema matching.

4 Approach: Metadata-based Proximity Mining for Pre-filtering Schema Matching

Our goal is to effectively apply early-pruning for holistic schema matching in a DL setting. Such pre-filtering is based on novel proximity mining techniques. Those techniques evaluate similarity among pairs of datasets using automatically extracted meta-features and utilising a data-mining ensemble supervised model to select highly similar pairs. We apply this using the stratified approach (Figure 4.1). An overview of the approach is summarised in Figure 4.2, which shows the steps required to compute the schema matching pre-filtering functions from Table 4.2. We explain how we build and apply those models in this section.

In the remaining subsections, we describe the details of our approach as follows: profile the datasets to extract the meta-features and pair-wise proximity metrics in *Subsection 4.1*, use supervised proximity mining to build the ensemble models for $Rel(A_i, A_j)$ and $Rel(D_y, D_z)$ in *Subsection 4.2*, then apply the models on new pairs of attributes and datasets in the DL to compute their $Sim(A_i, A_j)$ and $Sim(D_y, D_z)$ scores in *Subsection 15*, and finally using the $Sim(D_y, D_z)$ to predict $Rel(D_y, D_z)$ for pairs of datasets and applying the pre-filtering step for schema matching in *Subsection 4.3*.

4.1 Proximity Metrics: Meta-features Distances

Our approach gathers metadata at two levels of granularity: at the **I. dataset level** and **II. attribute level**. Further, at each of these levels, we gather **A. content-based** meta-features with profiling statistics and **B. name-based** meta-features with the naming of datasets and their attributes. The name-based techniques are the most commonly used metadata in previous research [52, 111, 121]. We propose other content-based meta-features at the two levels of granularity as follows:

- ***Dataset level (DS-Prox):*** We collect overall meta-features summarising the dataset content: overall statistics concerning all the attributes collectively, the attribute types found and the overall number of instances. The meta-features used are described in our previous work [18] (see Chapter 3 Section 4.1), which include a detailed list of meta-features that proved to be effective in predicting related datasets for schema matching pre-filtering, e.g., number of instances, number of attributes per attribute type, dimensionality, number of missing values, etc.
- ***Attribute level (Attribute-Prox):*** The set of meta-features used for both types of attributes, nominal and numeric, is described in Table 4.3. For each attribute A_i in dataset D, we profile it based on its type by computing the appropriate features.

Table 4.3: Attribute level content meta-features

Attribute Type	Meta-feature	Description
All	distinct_values_cnt	The number of distinct values
All	distinct_values_pct	The percentage of the distinct values from number of instances
All	missing_values_pct	The percentage of missing values from number of instances
Nominal	val_size_avg	The average number of strings in values from the attribute
Nominal	val_size_min	The minimum number of strings in values from the attribute
Nominal	val_size_max	The maximum number of strings in values from the attribute
Nominal	val_size_std	The standard deviation of number of strings in values from the attribute
Nominal	val_pct_median	The median percentage of instances per each value of the attribute
Nominal	val_pct_min	The minimum percentage of instances per each value of the attribute
Nominal	val_pct_max	The maximum percentage of instances per each value of the attribute
Nominal	val_pct_std	The standard deviation of the percentage of instances per each value of the attribute
Numeric	mean	The mean numeric value of the attribute
Numeric	std	The standard deviation of the numeric value of the attribute
Numeric	min_val	The minimum numeric value of the attribute
Numeric	max_val	The maximum numeric value of the attribute
Numeric	range_val	The numeric range of the values of the attribute
Numeric	co_of_var	The numeric coefficient of variance of the attribute

Equation 4.3 shows the proximity metric computed for a pair of attributes (or datasets), denoted as O_i, O_j. Using the meta-features described, we compute the z-score distance for each meta-feature m. The result is a real number, P_m. The z-score is a normalisation where we use the mean 'μ' and standard deviation 'σ' of each meta-feature considering its value from all datasets in the DL. A value of 0 is the most similar, while larger negative or positive number means more different. The z-score is used to standardise the comparisons of attributes in a holistic manner that considers all datasets and attributes in the DL. Most pairs of attributes and dataset will have a value falling in the range of $[-3, 3]$.

$$P_m = zscore\_distance(O_i, O_j) = \left| \frac{m(O_i) - \mu}{\sigma} - \frac{m(O_j) - \mu}{\sigma} \right| \tag{4.3}$$

For the name-based metadata we compute the proximity metric P_m with a Levenshtein string comparison function [86], as in Equation 4.4.

$$P_m = levenshtein\_distance(name(O_i), name(O_j)) \tag{4.4}$$

4.2 Supervised Proximity Mining

After the different proximity metrics of meta-features are generated by profiling the datasets, a *representative sample* (an adequate sample size should be similar to the sample in our experiments in Section 5) of dataset and attribute pairs should be selected by a human annotator and should be labelled whether they satisfy $Rel(D_y, A_z)$ and $Rel(A_i, A_j)$ respectively. The dataset and attribute pairs with their proximity metrics and labels are fed to a supervised learning algorithm to create a proximity scoring model. We propose supervised ensemble models based on different dataset level and attribute level proximity metrics for computing overall similarity between pairs of datasets. The models decide on the number of attributes to consider in order to evaluate a pair of datasets as 'related' by using different aggregation functions agg for the attribute level metrics, giving different weights to a different number of attribute linkages of different similarity ranks. This will be explained in detail in this section.

Our approach builds supervised ensemble models M_{cls-ds} for $Rel(D_y, D_z)$, and $M_{cls-nom-attr}$ & $M_{cls-num-attr}$ for $Rel(A_i, A_j)$ whether the attribute type is nominal or numerical respectively. Model-based learning for pre-filtering has been applied before in the collaborative filtering field [6]. In such scenarios, item pairs are recommended or filtered out using model-based learning algorithms where a learnt model is used to assign the similarities and rankings of item pairs based on previously annotated examples. We give details of how we learn the models and how we use them in our approach in the subsections below.

Building the models from annotated samples

An overview process for building the supervised models in our approach can be seen in Figure 4.4. In the ***build*** phase, we take the pairs of datasets and profile them by computing their dataset level and attribute level meta-features, followed by computing the proximity metrics for those extracted features. We take different dataset pair samples for building the attribute level models and the dataset level models as seen in the split into samples OML01 and OML02 (how to build such samples is given in Section 5.1). The pairs in sample OML01 should be already annotated to indicate whether their attributes match (i.e., $Rel(A_i, A_j)$ for attribute level models) or whether the datasets are relevant for schema matching or not in sample OML02 (i.e., $Rel(D_y, D_z)$ for dataset level models). Initially, we start with the attribute level

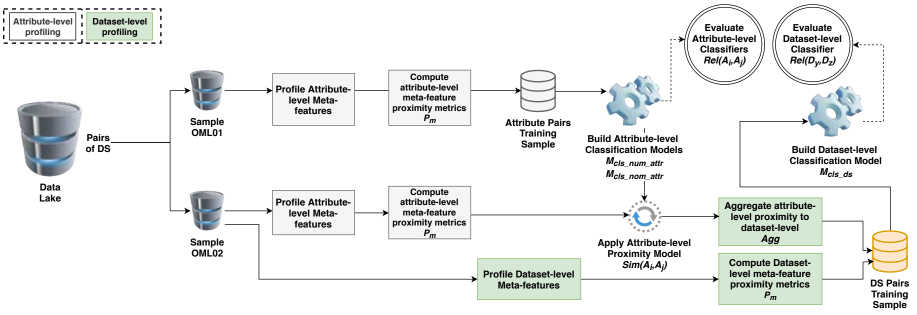

Attribute-level profiling
Dataset-level profiling
Data Lake
Pairs of DS
Sample OML01
Sample OML02
Profile Attribute-level Meta-features
Compute attribute-level meta-feature proximity metrics P_m
Attribute Pairs Training Sample
Build Attribute-level Classification Models $M_{cls\_num\_attr}$ $M_{cls\_nom\_attr}$
Evaluate Attribute-level Classifiers $Rel(A_i,A_j)$
Evaluate Dataset-level Classifier $Rel(D_y,D_z)$
Build Dataset-level Classification Model M_{cls_ds}
Apply Attribute-level Proximity Model $Sim(A_i,A_j)$
Aggregate attribute-level proximity to dataset-level Agg
Profile Dataset-level Meta-features
Compute Dataset-level meta-feature proximity metrics P_m
DS Pairs Training Sample

Figure 4.4: An overview of the process to ***build*** the supervised ensemble models in our proposed datasets proximity mining approach using previously manually annotated dataset pairs.

supervised learning procedure as it is only an auxiliary subcomponent used for the dataset level, where an aggregation step is used to compute dataset level proximities. First, we divide the pairs into training and test sets, we train a supervised learning ensemble model for each attribute type (nominal and numeric types) using the training sample, and we test the performance of the model on the test set (evaluation distinguished by dotted lines and circles in the figure). We conduct this test to guarantee that the models generated are accurate in detecting $Rel(A_i, A_j)$. Similarly, we do the same with the dataset level supervised models which generate $Rel(D_y, D_z)$. We use the dataset level proximity metrics and the attribute level aggregated proximity metrics together to train a supervised model using a training sub-sample of dataset pairs from OML02. Finally, we evaluate the generated dataset level supervised models to guarantee their accuracy in detecting $Rel(D_y, D_z)$.

Supervised learning. To build the models, we use classical supervised learning to create the proximity models. The meta-features are used as input to the models as seen in Figure 4.5, where an object could be an attribute for attribute-level models or a dataset for dataset-level models. First, for each object we extract its meta-features (i.e., 'm1', 'm2', ...). Then, for each object, we generate all pairs with each of the other objects and compute the proximity metrics between their meta-features using either Equation 4.3 for content-based meta-features or Equation 4.4 for the name-based comparison. We then take a sample of pairs of objects which are analysed by a data analyst; a human-annotator who manually decides whether the pairs of objects satisfy (assign '1') or not (assign '0') the *Rel* properties (see Section 3). This can be achieved by simply labelling the objects with their respective subject-areas and those falling under the same one are annotated as positively matching '1', otherwise all others are labelled with '0' (see Section 5.1). We then use supervised learning techniques and 10-fold cross-validation over the proximity metrics to create two types of independent models which can classify pairs of datasets or pairs of attributes according to $Rel(D_y, D_z)$ and $Rel(A_i, A_j)$ respectively. This is the final output consisting of the two auxiliary

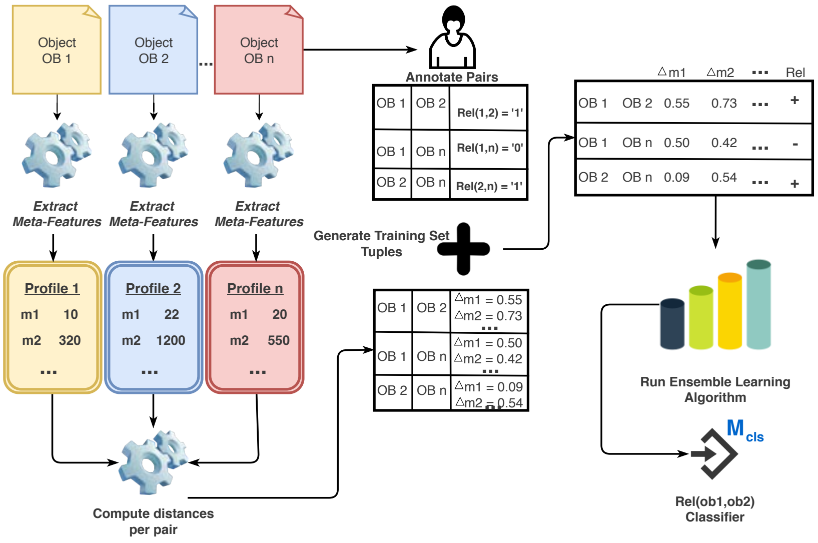


Figure 4.5: Proximity Mining: supervised machine learning for predicting related data objects.

supervised models $M_{cls-nom-attr}$, $M_{cls-num-attr}$ for $Rel(A_i, A_j)$ and the main dataset level model M_{cls-ds} for $Rel(D_y, D_z)$. The positive-class distribution from the generated models is used to *score* new pairs of objects (unseen in the training process) with a similarity score $Sim(D_y, D_z)$ using M_{cls-ds}, and $Sim(A_i, A_j)$ using $M_{cls-nom-attr}$ or $M_{cls-num-attr}$.

We use a random forest ensemble algorithm [29, 127] to train the supervised models in predicting related attribute and dataset pairs as it is one of the most successful supervised learning techniques. The algorithm generates a similarity score based on the positive-class distribution (i.e., the predicted probability of the positive-class based on weighted averages of votes for the positive class from all the sub-models in the ensemble model) to generate a score in $[0, 1]$ for Sim. For example, if Random Forest generates 1000 decision trees, and for a pair of datasets $[D_y, D_z]$ we get 900 trees vote positive for $Rel(D_y, D_z)$ then we get $\frac{900}{1000} = 0.9$ for $Sim(D_y, D_z)$ score.

We feed the supervised learning algorithm the normalised proximity metrics of the meta-features for pairs of datasets $[D_y, D_z]$. For attribute level meta-features, we feed the M_{cls-ds} model with all the different aggregations of the meta-features after computing their normalised proximity metrics (i.e., after applying Equation 4.7, which we describe later in this section).

Attribute-level proximity. To compute the overall proximity of datasets using their attribute level meta-features we use Algorithm 2, which first compares in Lines 5-10 the proximity metrics from the meta-features of each attribute of a specific type against all other attributes of the same type in the other dataset using $M_{cls-nom-attr}$ for nominal attributes and $M_{cls-num-attr}$ for numeric attributes. The algorithm then finds top matching attribute pairs in

Algorithm 2: Attribute Level Top-Similarity Matching

```
Input: Sets of the attribute meta-features of each type $Att_{nominal}$ and $Att_{numeric}$ containing the
       proximity metrics of the meta-features of each attribute in the pair $\{A_i, A_j\}$ for each dataset
       in the pair $\{D_y, D_z\}$, the model $M_{cls-nom-attr}$ for nominal attributes, the model $M_{cls-num-attr}$
       for numeric attributes, an aggregation function $Agg$ for aggregating attribute links to
       compute dataset level proximity
Output: The partially ordered set $SP$ of proximity metrics $P^D_m(D_y, D_z)$ for each pair of $\{D_y, D_z\}$
1  $SP_{dataset} \leftarrow \emptyset$;
2  $SP_{attribute} \leftarrow \emptyset$;
3  $SP_{top\_attribute} \leftarrow \emptyset$;
4  foreach $\{D_y, D_z\} \subset DL$ and $y \neq z$ do
5      foreach $\{A_i, A_j\} \subset Att_{nominal}$ and $A_i \in D_y$ and $A_j \in D_z$ do
6          $Sim(A_i, A_j) = M_{cls-nom-attr}(A_i, A_j)$;
7          $SP_{attribute} \leftarrow SP_{attribute} \cup \{[A_i, A_j, Sim(A_i, A_j)]\}$;
8      foreach $\{A_i, A_j\} \subset Att_{numeric}$ and $A_i \in D_y$ and $A_j \in D_z$ do
9          $Sim(A_i, A_j) = M_{cls-num-attr}(A_i, A_j)$;
10         $SP_{attribute} \leftarrow SP_{attribute} \cup \{[A_i, A_j, Sim(A_i, A_j)]\}$;
       \\Iterate on the set of attribute pairs $SP_{attribute}$ to find top matching pairs
11     while more attribute pairs $\{A_i, A_j\}$ can be picked do
12         Pick pair $\{A_i, A_j\}$ from $SP_{attribute}$ that maximises $Sim(A_i, A_j)$ where $A_i$ and $A_j$ were not
             picked before;
13         $SP_{top\_attribute} \leftarrow SP_{top\_attribute} \cup \{[A_i, A_j, Sim(A_i, A_j)]\}$;
14     $P^D_m(D_y, D_z) \leftarrow Agg(SP_{top\_attribute})$;
15     $SP_{dataset} \leftarrow SP_{dataset} \cup \{[D_y, D_z, P^D_m(D_y, D_z)]\}$;
```

Lines 11-13 where we match each attribute to the most similar attribute in the other dataset using a *greedy approach* [76]. For each pair of datasets, we match each attribute only once (we do not allow many-to-many matching). We rank attribute pairs by proximity top-to-least, then we assign matching pairs on the top of the list where each attribute did not appear in a previous higher ranking pair (i.e., both attributes need to be unmatched by any higher ranking pair in the list, otherwise the algorithm skips to the next pair until all the list of pairs is traversed). Finally, in order to summarise the attribute linkages to predict the overall proximity of the dataset pairs, we compute in Line 14 an *aggregation* of the top-matching attribute linkages found between a pair of datasets using an *function Agg* to convert the multiple $Sim(A_i, A_j)$ scores to a single proximity metric P^D_m for their dataset pair. We use different types of attribute level aggregation functions. Those functions assign different weights 'W' (which is an indicator of relevance, a bigger weight means more relevant) to the attribute links to consider. The different aggregations should have the goal of giving less weight to attribute links which could be considered as *noise*; i.e., those pairs which are too strongly correlated without any meaning (e.g., discrete ID numbers) or those attribute pairs with too low proximity to be significant.

Thus, the top-matching pairs of attributes are sorted by proximity weights and are fed to the aggregation function which allocates a weight between $[0, 1]$ for aggregation in the summation of weights. The total sum of weights should add up to 1.0. The different aggregations we use are as follows:

- ***Minimum:*** we allocate all the weight (i.e., $W = 1.0$) to the single attribute pair link with the minimum similarity, and we consider this as the overall proximity between the dataset pair. Therefore, all top-matching attribute pair links need to have a high similarity score to result into a high proximity for a dataset pair.
- ***Maximum:*** we allocate all the weight (i.e., $W = 1.0$) to the single attribute pair link with the maximum similarity, and we consider this as the overall proximity between the dataset pair. Therefore, only one top-matching attribute pair link needs to have a high similarity score to result into a high proximity for a dataset pair.
- ***Euclidean:*** a Euclidean aggregation of the similarities Sim of all matching pairs of attributes without any weighting as in Equation 4.5. Here we consider all the attribute pair links in the aggregation and we assign equal weights to all the links.

$$P_m^D = \sqrt{\sum_{i=1,j=2}^{n} [Sim(A_i, A_j)]^2} \tag{4.5}$$

- ***Average:*** a standard averaging aggregation of the pairs of attributes without any weighting, where all attribute links are equally weighted in the average.
- ***Weighted function:*** a normal distribution function to assign different proximity weights W for all attribute linkages found, and then summing up all the weighted similarities as the overall proximity as in Equation 4.6.

$$P_m^D = \sum_{i=1,j=2}^{n} [W_i * Sim(A_i, A_j)] \tag{4.6}$$

  This is visualised in Figure 4.6. Here the weight $0.0 \leqslant W \leqslant 1.0$ for each top-matching attribute linkage is assigned based on ordering the linkages top-to-least in terms of their similarity scores, and the weight allocated varies according to a normal distribution. We use different p-parameters (probability of success) of $\{0.1, 0.25, 0.5, 0.75, 0.9\}$, where a parameter of 0.5 leads to a standard normal distribution of weights allocated for the sorted pairs of attributes. A lower parameter value leads to skewness to the left, allocating more weight to highly related pairs, and a higher parameter leads to skewness to the right, allocating higher weights to pairs with lower ranked relationships. This means that with lower p we expect similar datasets to have a few very similar attributes and a higher p value means we expect most of the attributes to be strongly similar.

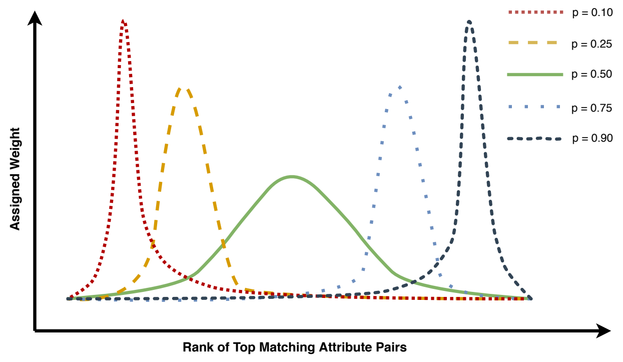


Figure 4.6: Different normal distributions for assigning weights to ranked attribute linkages.

As can be seen from their descriptions, each aggregation leads to a different meaning of similarity based on the number of attribute linkages to consider and which attribute linkages are considered more important (having higher weights assigned). All the dataset proximity metrics generated by the different aggregations listed above are finally normalised using Equation 4.7. The proximity metric P_m^D for two datasets D_y and D_z is computed by multiplying the number of matching attributes found (N), and divided by the minimum number of attributes of both datasets ($Min(|Attr_y|, |Attr_z|)$). This is done to prevent an inaccurate similarity score for two datasets having few very similar attributes of a single type, and many other attributes of different types. For example, if dataset D_1 has 1 nominal attribute and 10 numeric attributes and D_2 just has 8 nominal attributes, then if the single nominal attribute in D_1 is highly similar to a nominal attribute in D_2 (e.g., $Sim(A_1, A_2) = 0.9$) then the overall outcome without normalisation will be a high proximity metric between both datasets although they have many disjoint attribute types. The resulting proximity metric after normalisation for the datasets would be calculated as follows: $P\prime_m^D = 0.9 * \frac{1}{9} = 0.1$, so overall they will have a low proximity compensating for all the unmatched attributes without corresponding types.

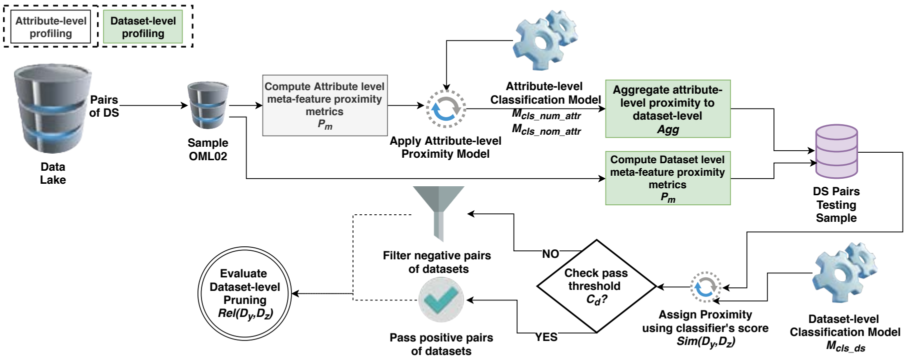
$$P\prime_m^D = P_m^D * \frac{N}{Min(|Attr_y|, |Attr_z|)} \tag{4.7}$$

Applying the models on the DL

In the second phase, after building the ensemble models, we apply them to each new pair of previously unseen datasets to achieve a measure of the similarity score. When applying the models, we compute for each pair of datasets the similarity score of $Sim(D_y, D_z)$ and for each attribute pair $Sim(A_i, A_j)$ using the supervised models extracted in the previous phase. The Sim score is the positive-class distribution value generated by each ensemble model [127]. For the attribute level scoring, we complete the proximity mining task by aggregating the *sim* scores between pairs of datasets (as seen in the last steps of Algorithm 2). To compare dataset pairs, we use Algorithm 3, and the M_{cls-ds} model generated by the previous build phase.

Algorithm 3: Dataset level Matching

```
Input: Dataset-level proximity metrics for each pair of datasets $\{D_y, D_z\}$, the model $M_{cls-ds}$
Output: The set $SP$ of similarity score $[D_y, D_z, Sim(D_y, D_z)]$ for each pair of $\{D_y, D_z\}$
1 $SP \leftarrow \varnothing$;
2 foreach $\{D_y, D_z\} \subset DL$ and $y \neq z$ do
3     $Sim(D_y, D_z) = M_{cls-ds}(D_y, D_z)$;
4     $SP \leftarrow SP \cup \{[D_y, D_z, Sim(D_y, D_z)]\}$;
```



Attribute-level profiling
Dataset-level profiling
Data Lake
Pairs of DS
Sample OML02
Compute Attribute level meta-feature proximity metrics P_m
Apply Attribute-level Proximity Model
Attribute-level Classification Model
$M_{cls\_num\_attr}$
$M_{cls\_nom\_attr}$
Aggregate attribute-level proximity to dataset-level Agg
Compute Dataset level meta-feature proximity metrics P_m
DS Pairs Testing Sample
Dataset-level Classification Model M_{cls_ds}
Assign Proximity using classifier's score $Sim(D_y,D_z)$
Check pass threshold C_d?
NO
YES
Filter negative pairs of datasets
Pass positive pairs of datasets
Evaluate Dataset-level Pruning $Rel(D_y,D_z)$

Figure 4.7: An overview of the process to ***apply*** the learnt supervised models in our approach for pre-filtering previously unseen dataset pairs independent of the build process.

In the ***apply*** phase visualised in Figure 4.7, we take the pairs of datasets from sample OML02 which have not been used in the build phase and we compute the proximity metrics for the dataset level and attribute level meta-features. First, we profile the attribute level meta-features from this new dataset pairs sample. Then, we apply the attribute level supervised models resulting from the previous sub-process to score the attribute pairs similarities from the different dataset pairs. Then, we aggregate the resulting attribute pairs similarities to the dataset level using the aggregation functions. Once we have the dataset level proximity metrics generated from dataset level and attribute level meta-features, we feed them all to the dataset level supervised models from the build phase to unseen testing set pairs, not used in the training of the models, which assigns a proximity score to the pairs. If a pair exceeds a certain proximity threshold, we consider that pair as a positive match to propose for further schema matching, otherwise the pair is considered as a negative match and is pruned out from further schema matching tasks (we evaluate this by pruning effectiveness metrics in Section 5.2). This is described in the next subsection.

4.3 Pre-filtering Dataset Pairs for Schema Matching

For the final step of pre-filtering pairs of datasets before applying detailed instance-based schema matching, we check whether the pairs of datasets are overall related or not, and therefore whether they should be filtered out or proposed for expensive schema matching. We analyses the final $Sim(D_y, D_z)$ score generated by the model M_{cls-ds} for each dataset pair in the DL to decide whether they satisfy the $Rel(D_y, D_z)$ or not. We consider the relationship of each dataset with each of the other datasets existing in the DL. Each dataset must pass the similarity threshold c_d with each individual dataset to be proposed for detailed schema matching (as in Equation 4.1).

If we choose a high cut-off threshold we restrict the supervised model to return less pairs of high proximity, leading to lower recall but also less comparisons, thus helping to reduce the computational time at the expense of possibly missing some misclassified pairs. Alternatively, if we choose a lower cut-off threshold, we relax our model to return pairs of lower proximity. This leads to more pairs (i.e., more work for further schema matching tasks) yielding positive matches and higher recall of positive cases, but, with more pairs marked incorrectly as matching. We propose how to select an appropriate threshold that optimises this trade-off empirically in Section 5.

The complexity of our approach is quadratic in the number of objects (attributes or datasets) compared, and therefore runs in polynomial time, however, it applies the cheapest computational steps for early-pruning (just computing distances in Equations 4.3 and 4.4 and applying the model to score each pair). This way, we save unnecessary expensive schema matching processing per each value instance of the attributes in later steps, reducing the computational workload at the detailed granularity schema matching level by pre-filtering the matching tasks. We demonstrate this empirically in Section 5.5.

5 Experimental Evaluation

In this section, we present the experiments which evaluate our approach by using a prototype implementation. We evaluate the following components of our approach in predicting $Rel(D_y, D_z)$ for pre-filtering schema matching:

- **Proximity metrics:** we evaluate the different individual dataset level and aggregated attribute level meta-features.
- **Supervised models:** we also evaluate the ensemble supervised models, which consume the proximity metrics, in the pre-filtering task.

In addition, we evaluate the sub-components of our approach which include the attribute level models $M_{cls-nom-attr}$ and $M_{cls-num-attr}$ in predicting

$Rel(A_i, A_j)$. We test the attribute level model in experiment 1, the dataset level pruning effectiveness against the ground-truth in experiment 2, and the computational performance in experiment 3.

In experiments 2 and 3, we compare the performance of our proposed proximity mining models against traditional instance-based schema matching techniques we proposed in Chapter 2. Those are the most expensive techniques which compare values from instances in the datasets to for computing schema similarity. We benchmark our results against a naïve averaging of attribute similarity from a prototype called Probabilistic Alignment of Relations, Instances, and Schema (PARIS), which is one of the most cited schema matching tools [126]. PARIS was found to be best performing with large datasets when compared against other tools [76] and does not need collection of extra metadata (see Table 4.1). PARIS does exact value-string matching based on value-frequency inverse functionality [126]. We implement a prototype in [15] (see Chapter 2) which compares pairs of attributes from different datasets using PARIS and generates an overall score for $Sim(D_y, D_z)$ by averaging $Sim(A_i, A_j)$ generated by PARIS from the top-matching attribute-pairs (similar to Algorithm 2, where PARIS replaces the supervised models). It converts tabular datasets to RDF triples, and executes a probabilistic matching algorithm for identifying overlapping instances and attributes. We selected PARIS because of its simplicity and ease of integration with Java-based APIs and its high performance in previous research [76]. We parametrised the prototype with the top performing settings from experiments in [15] (see Chapter 2) sampling 700 instances per dataset, 10 iterations comparisons, with identity and shingling value strings matching. This will be a baseline pre-filtering heuristic approach we shall compare against in the experiments.

The rest of this section describes the datasets used in the experiments, the evaluation metrics used and the different experiments implemented. We present the results from our experiments and discuss their implications.

5.1 Datasets

We use the OpenML DL[2] in our experiments [131], which has more than 20,000 datasets intended for analytics from different subject areas. OpenML is a web-based data repository that allows data scientists to upload different datasets, which can be used in data mining experiments. OpenML stores datasets in the ARFF tabular format which consist of diverse raw data loaded without any specific integration schema. This allows us to evaluate our approach in a real-life setting where datasets come from heterogeneous domains.

We use two subsets of manually annotated datasets from OpenML as our ground-truth (gold standard) for our experiments. Those two subsets

[2]https://www.openml.org

have been generated using two different independent processes, and therefore provide independently generated ground truths that do not overlap. As the research community is lacking appropriate benchmarking gold standards for approximate (non-equijoins) dataset and attribute similarity search [89], we published those datasets online to support in future benchmarking tasks[3]. The experimental datasets are described as follows:

- **OML01 - The attribute level annotated 15 DS:** consists of 15 datasets from different domains as described in Table 4.4. The total number of attributes is 126 (61 nominal and 65 numeric), and the average number of attributes per dataset is 8. There is a total of 3468 pairs of attributes to be matched (1575 nominal pairs and 1892 numeric pairs). All the pairs of attributes in this subset were manually scrutinised by 5 annotators consisting of post-graduates with an average age of 28, where 4 are pharmacists and 1 is a computer scientist. They checked the attributes in the datasets and annotated all the pairs of attributes from different datasets with related data, $Rel(A_i, A_j)$. It took on average 3 hours by each annotator to complete the task. Annotators assign a single value from $\{0, 1\}$, where '1' means a related attribute, and the majority vote is taken for each pair, where the average Kappa coefficient for the inter-rater agreement is 0.59, the maximum is 0.80 and the minimum 0.37. Annotators were given the following to judge if the attribute pair is related: attribute name, OpenML dataset description, top 10 values, and the mean and standard deviation for numeric attributes. We didn't give instructions on how to use the provided information to judge, but we described that "related attributes should store data related to similar real-world properties, e.g., car prices, specific body size measurements like height, etc., and should contain similar data". Examples of the annotations can be seen in Table 4.5. There are only 56 positively matching pairs (19 nominal and 37 numeric). This subset is used in training the attribute level models for computing the similarity between attributes from different datasets and predicting related attributes, $Rel(A_i, A_j)$.

Table 4.4: Description of the OML01 datasets

Domain	Datasets IDs	Datasets
Vehicles	21,455,967,1092	car,cars,cars,Crash
Business	223,549,841	Stock,strikes,stock
Sports	214	baskball
Health	13,15,37	breast-cancer,breast-w,diabetes
Others	48,50,61,969	tae,tic-tac-toe,Iris,Iris

- **OML02 - The dataset level annotated 203 DS:** consists of 203 datasets different from those in the OML01 subset. To collect this sample, we

[3]https://github.com/AymanUPC/all_prox_openml

Table 4.5: Example Cross-dataset Attribute Relationships from OML01

No.	Dataset 1	Dataset 2	Attribute 1	Attribute 2	Relationship
1	37 (diabetes)	214 (baskball)	age	age	related
2	455 (cars)	549 (strikes)	model.year	year	related
3	455 (cars)	967 (cars)	all	all	duplicate
4	455 (cars)	1092 (Crash)	name	model	related
5	455 (cars)	1092 (Crash)	weight	Wt	related

scraped the OpenML repository to extract all datasets not included in the OML01 sample and having a description of more than 500 characters. Out of the 514 datasets retrieved, we selected 203 with meaningful descriptions (i.e., excluding datasets whose descriptions do not allow to interpret the content and to assign a topic). The datasets have a total of 10,971 attributes (2,834 nominal, 8,137 numeric). There are 19,931 pairs of datasets with about 35 million attribute pairs to match. According to Algorithm 2, there are 3.7 million comparisons for nominal attributes (leading to 59,570 top matching pairs) and 31.5 million numeric attribute pairs (leading to 167,882 top matching pairs). We try to prevent the value-based schema matching on all possible pairs of values between datasets, where there are 216,330 values which would lead to 23.4 billion comparisons at the value level. A domain expert with a background in pharmaceutical studies and one of the authors collaborated to manually label the datasets[4]. They used the textual descriptions of the datasets to extract their topics, which is common experimental practice in dataset matching assessment, similar to the experimental setup in [22]. The annotators sat together in the same room and discussed each dataset with its description and decided on its appropriate real-life subject-area (e.g., car engines, computer hardware, etc.). To group similar datasets in the same subject-area grouping, annotators had to discuss and agree together on a single annotation to give to a dataset. This was done by discussing the specific real-world concept which the dataset describes, e.g., "animal profiles", "motion sensing", etc. The annotators were only allowed to scrutinise the textual descriptions of the datasets and did not receive the underlying data stored in their attributes to prevent any bias towards our proposed algorithms. It took the annotators about 15 hours in total to annotate the datasets. Pairs of datasets falling under the same subject-area were positively annotated for $Rel(D_y, D_z)$. The sample consists of 543 positive pairs from the 20,503 total number of pairs. The details of the sample is summarised in Table 4.6, which lists the number of datasets, the number of topics, top topics by the number of datasets, and the number of related pairs. Some of the pairs from the sample can be seen in Table 4.7. We can see, for example, that dataset with ID 23

[4]Those dataset annotations were reviewed by 5 independent judges, and the results of this validation are published online at:
https://github.com/AymanUPC/all_prox_openml/blob/master/OML02/oml02_revalidation_results.pdf

should match all datasets falling under the topic of 'census data' like dataset 179. Both datasets have data about citizens from a population census. In row 4, we can see an example of duplicated datasets having highly intersecting data in their attributes. Duplicate pairs like those in row 4 have the same number of instances, but described with different number of attributes, which are overlapping. We consider all duplicate pairs of datasets as related pairs. We aim to detect and recommend such kind of similar dataset pairs as those in Table 4.7 for schema matching using our proximity mining approach.

Table 4.6: Description of the OML02 datasets

Datasets	Topics	Top Topics	$\mathbf{Rel}(\mathbf{D_y}, \mathbf{D_z})$
203	74	computer software defects (16), health measurements (13), digit handwriting recognition (12), robot motion sensing (11), plant and fungi measurements (9), citizens census data (8), diseases (8)	543

Table 4.7: An example of pairs of datasets from the OML02 sample from OpenML

No.	DID 1	Dataset 1	DID 2	Dataset 2	Topic	Relationship
1	23	cmc	179	adult	Census Data	related
2	14	mfeat-fourier	1038	gina_agnostic	Digit Handwriting Recognition	related
3	55	hepatitis	171	primary-tumor	Disease	related
4	189	kin8nm	308	puma32H	Robot Motion Sensing	duplicate

5.2 Evaluation Metrics

We use different evaluation metrics to assess the effectiveness of our approach. We use the traditional recommendation and information retrieval evaluation metrics similar to other research [62, 86], including precision, recall and ROC measurements. For the supervised models, we use traditional data mining classification effectiveness metrics [127]. We evaluate the computational costs of our approach vs. traditional schema matching for baseline comparison. Those metrics are categorised per the experiment types and granularities:

- Classification effectiveness
  - Granularity: Attribute level $Rel(A_i, A_j)$ and Dataset level $Rel(D_y, D_z)$
  - Models evaluated: $M_{cls-nom-attr}$, $M_{cls-num-attr}$, M_{cls-ds}
  - Classification measures: Classification accuracy, Recall, Precision, ROC, Kappa

- Pre-filtering (pruning) effectiveness
  - Granularity: Dataset level $Rel(D_y, D_z)$
  - Model evaluated: M_{cls-ds}, $PARIS$
  - Retrieval measures: Recall, Precision, Efficiency Gain, Lift Score
- Computational performance
  - Granularity: Attribute level $Rel(A_i, A_j)$ and Dataset level $Rel(D_y, D_z)$
  - Model evaluated: $M_{cls-nom-attr}$, $M_{cls-num-attr}$, M_{cls-ds}, $PARIS$
  - Computational measures: computational processing time (milliseconds), metadata size (megabytes)

For the classification effectiveness measures, the classification accuracy is given in our results as a percentage. The recall and precision rate are also percentages. For the ROC (area under the curve) and Kappa statistic, they are a real value between 0 and 1, where the value significance is evaluated in our results according to Tables 4.8-4.9.

Table 4.8: The significance of the Kappa statistic

Kappa	Significance
<0	Disagreement
0.0 - 0.10	No significance
0.0 - 0.20	Slight
0.21 - 0.40	Fair
0.41 - 0.60	Moderate
0.61 - 0.80	High
0.81 - 1.0	Excellent

Table 4.9: The significance of the ROC statistic

ROC	Significance
<0.5	Disagreement
0.5 - 0.6	No significance
0.6 - 0.7	Slight
0.7 - 0.8	Moderate
0.8 - 0.9	High
0.9 - 1.0	Excellent

For the pruning effectiveness measures, we evaluate our approach using the measurements described in Equations (4.8),(4.9), (4.10) and (4.11). Here, TP means true-positives which are the pairs of datasets correctly classified by the models. FN are false negatives, FP are false-positives, TN are true-negatives, and N indicates the total number of possible pairs of datasets (which is a sum of all pairs TP + FP + TN + FN). The efficiency gain measures the amount of reduction in work required, in terms of number of pairs of datasets eliminated by our models. The lift score measures the capability of the model in filtering out more pairs than randomly removing pairs for the recall rate achieved. A higher amount is better, where a value of 3.0 would mean that the model is capable of retrieving 3 times more positive pairs than the expected amount of positive pairs from a random sample without using the model.

$$recall = \frac{TP}{TP + FN} \quad (4.8)$$

$$precision = \frac{TP}{TP + FP} \quad (4.9)$$

$$efficiency\text{-}gain = \frac{TN + FN}{N} \quad (4.10)$$

$$lift\text{-}score = \frac{recall}{(1.0 - efficiency\text{-}gain)} \tag{4.11}$$

5.3 Experiment 1: Attribute-level Models

Our goal in this experiment is to evaluate the supervised models we build for detecting the relationship between related attributes $Rel(A_i, A_j)$ using attribute level content meta-features as follows:

- **Dataset**: OML01
- **Evaluation metrics**: Classification effectiveness
- **Relationship evaluated**: $Rel(A_i, A_j)$
- **Input**: the attribute level meta-features matching for pairs of attributes.
- **Output**: a supervised model to predict related attributes per type.
- **Goal**: select the most appropriate models for predicting related attributes by evaluating their effectiveness for each type (nominal or numerical).
- **Description**: we take two subsets of attribute pairs and their meta-features, depending on the attribute type: nominal attributes and numeric attributes. The subsets are annotated by a human to decide whether $Rel(A_i, A_j)$ is 1 or 0. We build a proximity model using a supervised learning algorithm.

Experimental Setup

we evaluate the model using the leave-one-out approach (where we exclude one pair from training in each run, and use it to test the output model, therefore having a cross-validation where the number of folds is equal to the number of pairs). As the number of positive pairs to negative pairs are imbalanced, we create a balanced training set for each type, nominal or numeric, which consists of all the positive pairs of attribute matches and an equal number of negative unmatching pairs. To make the training set representative of all the different negative cases, we cluster the negative cases using the Expectation Maximisation (EM) algorithm [127] and we select a representative sample of negative cases from each cluster.

Results

The attribute level models were evaluated for both nominal attribute pairs and numeric attribute pairs. We evaluate the $M_{cls-nom-attr}$ and $M_{cls-num-attr}$

models which assign the $Sim(A_i, A_j)$ for attribute pairs. As could be seen in Table 4.10, we created two supervised models; one for each type of attribute pairs. Both models achieved excellent ROC performance and highly significant results on the Kappa statistic (see Tables 4.8-4.9 results significance). The models had good accuracy, recall, and precision rates. This is important because the dataset pairs pre-filtering step depends on this attribute proximity step, so we have to achieve a good performance at this level to minimise accumulation of errors for the following tasks.

Table 4.10: Performance evaluation of attribute pairs proximity models

Model	ROC	Kappa	Accuracy	Positive Recall	Positive Precision
Nominal	0.957	0.65	82.5%	89.5%	77.3%
Numeric	0.915	0.7	84.8%	89.2%	80.5%

5.4 Experiment 2: Dataset-level Models

In this experiment, we evaluate the effectiveness of the dataset level models in pre-filtering dataset pairs for further schema matching. Our goal is to evaluate how good is our approach in retrieving related datasets $Rel(D_y, D_z)$ and filtering out unrelated datasets from the schema matching process. We evaluate the effectiveness of correctly proposing related datasets for schema matching using the different types of models we describe later in this section. The evaluation is as follows:

- **Dataset**: OML02
- **Evaluation metrics**: Classification effectiveness, Pre-filtering effectiveness
- **Relationship evaluated**: $Rel(D_y, D_z)$
- **Input**: Pairs of datasets with different types of meta-features matching [dataset level names, dataset level content meta-features, attribute level names, attribute level meta-features, all meta-features].
- **Output**: a supervised model M_{cls-ds} to predict related datasets based on proximity mining and PARIS baseline.
- **Goal**: select the best proximity model to predict related datasets and the proximity threshold c_d to use with that model.
- **Description**: we take annotated pairs of datasets and their meta-features' normalised metrics. The pairs are annotated by a human annotator to decide whether $Rel(D_y, D_z)$ is 1 or 0. We build a proximity model using a supervised learning algorithm.

We create different types of dataset pairs ensemble models to score $Sim(D_y, D_z)$ by using different combination of meta-feature types. We create different models depending on the meta-feature type(s) used as input (namely those are dataset content meta-features, dataset name similarity, attribute content meta-features, and attribute name similarity). We can combine the meta-feature types used to build the model or use each type separately to lead to the following model types depending on which meta-features are used:

- **DS-Prox-Content:** uses the dataset level content meta-features, without considering the dataset name distance.
- **DS-Prox-Name:** uses the dataset name Levenshtein distance as the only predictor of dataset pairs similarity.
- **Attribute-Prox-Content:** uses the attribute level content meta-features, not considering the attribute name meta-features.
- **Attribute-Prox-Name:** uses the attribute level name meta-features only, not considering the attribute content meta-features.
- **Name-Prox:** uses dataset level and attribute level name-based meta-features only.
- **Content-Prox:** uses dataset level and attribute level content-based meta-features only.
- **All-Prox:** uses all the dataset level and attribute level meta-features, including both name-based and content-based meta-features.

We differentiate between the meta-feature types in our experiments so we can test if a specific subset of meta-features is better in predicting $Rel(D_y, D_z)$ or whether it is necessary to use all of them together to build an effective proximity mining model for the pre-filtering task. We also investigate if there is a difference in performance with regards to the types of meta-features extracted: classical name-based meta-features vs. the newly proposed content-based meta-features, and whether using both types together in combination leads to better results. We also separate the types so we can distinguish if purely content-based meta-features can be used as an alternative to name-based meta-features, especially in the case when the datasets and their attributes are not properly named. The most comprehensive of all models is the *All-Prox* model which uses all the possible meta-features we collect from the data profiling step. The *DS-Prox-Name* is the most generic of all models as it just considers a single meta-feature at the most abstract-level, therefore it will be used as our baseline for our performance comparisons in the experiments.

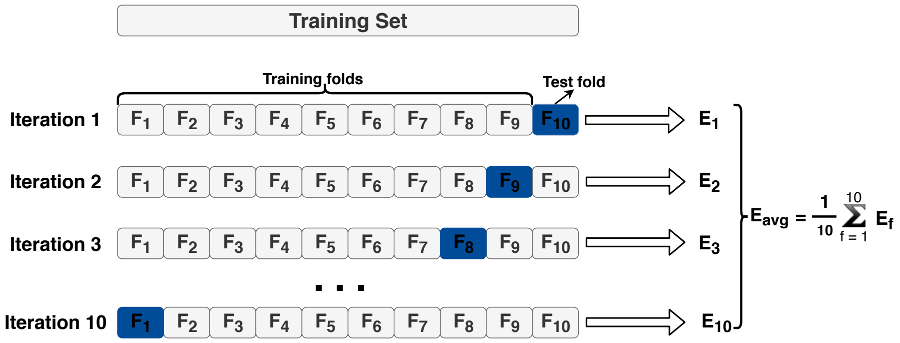

Training Set
Training folds
Test fold
Iteration 1
Iteration 2
Iteration 3
Iteration 10
F_1 F_2 F_3 F_4 F_5 F_6 F_7 F_8 F_9 F_{10}
E_1
E_2
E_3
E_{10}
$E_{avg} = \frac{1}{10}\sum_{f=1}^{10} E_f$

Figure 4.8: A 10-fold cross-validation experimental setup consisting of alternating folds in training and test roles. Image adapted from the book: Raschka S (2015) Python Machine Learning. 1st Edition. Packt Publishing, Birmingham, UK.

Experimental Setup

To evaluate our approach and models, we consider in our experiments a 10-fold cross-validation experimental setup using different subsets of datasets from a real-world DL, as visualised in Figure 4.8. The purpose of a cross-validation setup is to select the best supervised model for the pre-filtering task by evaluating the different models on test-sets separate from the training-sets, which is commonly used in recommendation assessment experiments [6, 62] and schema matching tasks [35]. Such an experimental setup increases the validity and generalisability of our experiments and approach. This is only achieved if the training set is representative of the cases found in the real-life population.

We make sure that the folds do not include intersecting dataset pairs. We create a balanced training set with all the positive cases and an equal number of negative cases similar to experiment 1. As the number of negative cases is much higher in the OML02 subset too, we also follow the clustering of negative cases approach to select a representative sample from each cluster (see Section 5.3). As seen in Figure 4.8, we iterate 10 times using an alternating fold as the test set, and the remaining folds as the training set. We evaluate the models in accurately predicting $Rel(D_y, D_z)$ with 9 different cut-off thresholds in [0.1-0.9] for 'c_d' from Equation (4.1) in order to cover a wide range of values. Finally, we evaluate the performance of each model by averaging the evaluation metrics ('E') from all 10 iterations. We also compute standard deviations in performance between different folds to evaluate the stability and consistency of the models evaluated. For the PARIS baseline implementation, it does not need to train any models, so we simply run it on all pairs of datasets and compare its results to our approach.

Results

Classification effectiveness. First, we created the models for the dataset pairs which assign $Sim(D_y, D_z)$ and check if they satisfy $Rel(D_y, D_z)$ by passing the minimum threshold. We evaluated the classification effectiveness measures for each proximity model after the 10-fold cross-validation. The results are summarised in Figures 4.9-4.11. The figures show a plot of results (from 10 folds) and interquartile ranges of accuracy, kappa statistic and ROC statistic for each model type. The distribution between folds can also be seen to assess the stability of the models. For our comparison, we use the name-based models, which are common in previous research, as our baseline comparison. As could be seen, all models were stable with very close values for the different evaluation metrics, indicating the versatility of our approach. However, still the All-Prox and Attribute-Prox models consistently had slightly better stability (lower deviations) than Name-Prox and other models. It can be seen from the results that the All-Prox and Attribute-Prox models are consistently performing better than the name-based model in terms of accuracy, ROC and Kappa statistic. This indicates that our proposed content-based and name-based combined meta-features models perform best with schema matching pre-filtering.

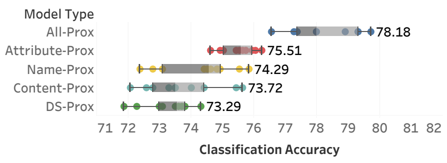


Figure 4.9: Classification accuracy from 10-fold cross-validation of dataset pairs pre-filtering models.

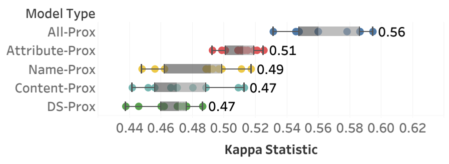


Figure 4.10: Kappa statistic from 10-fold cross-validation of dataset pairs pre-filtering models.

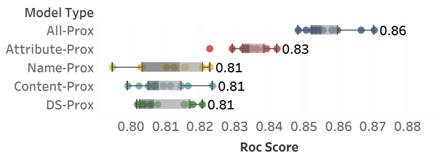


Figure 4.11: ROC statistic from 10-fold cross-validation of dataset pairs pre-filtering models.

The different models used for the schema matching pre-filtering task achieve different results because the meta-features used in the different models are not correlated, therefore contain different information about the datasets leading to the different performance of each model. We evaluated the Spearman rank correlation [62] between the different types of meta-features, which is presented in Table 4.11. The Spearman rank correlation ranks the dataset

Table 4.11: Spearman rank correlation for the different meta-features. We aggregate minimum (Min.), average (Avg.), maximum (Max.), & standard deviation (Std. Dev.) for different meta-feature types.

Type 1	Type 2	Min. Correlation	Avg. Correlation	Max. Correlation	Std. Dev. Correlation
Attribute Name	Attribute Content	-0.12	-0.01	0.19	0.04
Attribute Name	Dataset Content	0.02	0.04	0.10	0.02
Attribute Name	Dataset Name	0.06	0.07	0.13	0.02
Dataset Content	Attribute Content	-0.01	0.09	0.15	0.04
Dataset Content	Dataset Name	-0.02	0.00	0.02	0.02
Dataset Name	Attribute Content	0.00	0.01	0.04	0.01

pairs according to the proximity metrics of the meta-features. If the dataset pairs have the same identical rankings between two different meta-features then we get a perfect correlation. If the rankings produced in descending order by the two proximity metrics are different (e.g., a dataset pair can be ranked in the 100th position by one meta-feature and in the 9th position by the other, which have a difference of 81 ranks) then we get a lower correlation, with completely uncorrelated meta-features. We evaluated the average, standard deviation, minimum, and maximum of the correlation between the meta-features falling under the different types of meta-features. Recall that each type will have multiple meta-features (see Section 4.1), like attribute content will include all the meta-features in Table 4.3 with all their different proximity metrics according to the aggregations described in Section 4.2. We calculate the correlation between each individual meta-feature pair and we calculate aggregates per type. As can be seen in Table 4.11, all the correlation values are low.

Pre-filtering effectiveness. We compare the effectiveness of our approach against the baseline implementation of PARIS. We change the cut-off thresholds for Equation 4.1, and we aim to maximize the efficiency-gain while maintaining the highest recall for all candidate dataset pairs satisfying $Rel(D_y, D_z)$. The effectiveness is also evaluated by lift scores. The results from our approach and from the *'baseline'* PARIS prototype are presented in Figures 4.12-4.15. Figures 4.12-4.13 show the results for the different supervised models and PARIS, and Figures 4.14-4.15 show the results of the same evaluation metrics but for the individual meta-features in our approach, where we use the individual proximity metrics of the meta-features directly as an indicator of $Rel(D_y, D_z)$ without using any supervised learning models. We evaluate the dataset level meta-features from Section 4.1. The graphs show the average performance for all the individual metrics per specific type. We use a different minimum threshold with each proximity model or meta-feature in Figures 4.12-4.15 leading to the different plotted results per model or meta-feature. The aim of comparing both models and individual metrics is to be able to detect if the proposed supervised proximity models perform any better than simply using single independent metrics for the pre-filtering task.

For each evaluation of the models or the individual metrics, we evaluate the efficiency gain against recall first. We set a minimum target recall of 80%

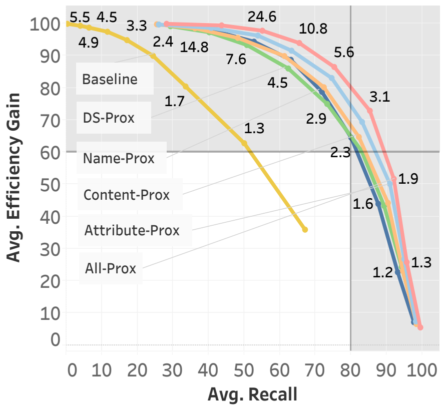


Figure 4.12: Recall against efficiency gain for the different supervised models.

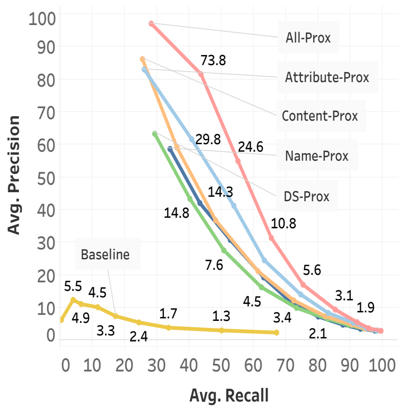


Figure 4.13: Recall against precision for the different supervised models.

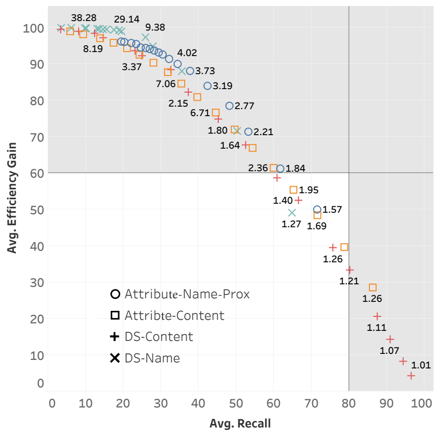


Figure 4.14: Recall against efficiency gain for the different metric types.

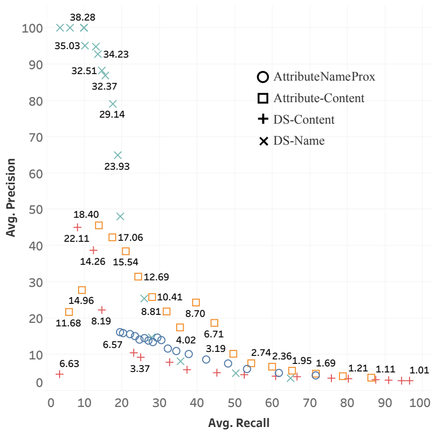


Figure 4.15: Recall against precision for the different metric types.

and a minimum target efficiency gain of 60% (i.e., filtering out at least 60% of the pairs of datasets while still proposing 80% of the true positive pairs), which are the grey shaded areas in the graphs. The minimum thresholds can be selected differently according to the requirements of the data analyst. Good performing models or proximity metrics are those that fall in this shaded area. The numbers annotated to some of the points in the graphs indicate the lift score (higher values are better). Similarly, we compare the precision against the recall in the second graphs for each evaluation (models or metrics). We also annotate some selected lift scores for some points in the graph.

When comparing our new proposed proximity models with the proximity model (DS-Prox) from our previous work [18] (described in Chapter 3), it can be seen that our Attribute-Prox model and the All-Prox model perform consistently better. This is expected because we are collecting finer granularity metadata to describe the datasets which makes it easier in the supervised learning task to differentiate between positive pairs and negative pairs. Although our new proposed techniques outperform our previous work in the DS-Prox model in terms of recall rates and lift scores, it comes at the price of a more computationally expensive algorithm (Algorithm 2). The complexity of the dataset level Algorithm 3 is $O([n * (n-1)]/2)$ while the complexity of the attribute level Algorithm 2 is $O([n * (n-1) * a^2]/2)$ where 'n' is the number of datasets and 'a' is the number of attributes in each dataset (we can use the average number of attributes per dataset as an approximation for 'a' when estimating the number of computations required).

If we would compare the content meta-features only model (Content-Prox) with the name meta-features only model (Name-Prox), we would see that both models perform equally the same in the pre-filtering task, although combining them in the All-prox model leads to the best results capturing the similarity of difficult pairs that can not be retrieved by any single type individually. Therefore, it is possible to solely depend on content-based proximity models as a replacement of name-based proximity models to achieve similar results. This will be important in the cases of DLs which are not well maintained and do not have properly named datasets and attributes. We investigate in detail the performance of the All-Prox proximity model based on its true positives, false positives and false negative pairs in the Appendix[5], where we present the exact cases, we discuss the reasons of discrepancies and we give a comparative analysis of the underlying proximity metrics which led to those cases.

For each of the pruning effectiveness evaluation metrics listed above, we compute the average and standard deviation of the measure between the different folds of evaluation for our approach. The average is plotted in the graphs in Figures 4.12-4.15, and the standard deviations of each model for the threshold 0.5 (we chose the mean threshold) are given in Table 4.12. The standard deviation indicates the stability of our proposed metrics and models with different subsets of datasets. We aim for a low standard deviation to prove the high adaptability of our approach.

Table 4.12: The standard deviation of each evaluation measure for 10-fold cross-validation of each dataset pairs pre-filtering model, where $c_d = 0.5$

Proximity	SD Recall	SD Efficiency Gain	SD Precision	SD Lift Score
All-Prox	5.4	0.61	0.58	0.32
Attribute-Prox	6.5	1.0	0.7	0.24
Content-Prox	6.8	1.0	0.38	0.35
DS-Prox	7.86	0.85	0.29	0.28
Name-Prox	6.3	1.1	0.47	0.24

[5]The appendix could be found online at https://aymanupc.github.io/all_prox_openml

Dataset Pairs Pre-filtering Meta-features

First, we assess if the supervised learning models perform better than a simpler approach based on the sub-components they are dependant on, which are the individual meta-features used in the models. The supervised models use multiple features in combination to score the similarity of pairs of datasets. Here, we assess the individual features as a baseline to compare against, and whether simply using a proximity metric of an individual meta-feature without any models can lead to any good result. We aggregated an average for the pruning evaluation metrics per each type of meta-feature. The results comparing recall against efficiency gain is given in Figure 4.14. In our experiments, no single meta-feature was able to individually predict related pairs of datasets to achieve optimum recall and efficiency gain, as can be seen by the lack of any plotted result in the top-right box. As seen in Figure 4.15, the pre-filtering task using the meta-features can not have a precision better than 10% for the higher recall rates.

We note here that the different types of meta-features are able to model different information about the datasets and their attributes as seen by the low correlations in Table 4.11. That is the main reason we used the combination of different types of meta-features in our proximity models which are able to combine the meta-features to give better results.

Dataset Pairs Pre-filtering Efficiency Gain Vs. Recall

We also evaluated the different supervised proximity models by testing their pre-filtering performance with different proximity thresholds. As can be seen in Figure 4.12, all of the proximity models were able to optimise recall and efficiency gain to achieve results in the top-right shaded area, compared to the baseline PARIS implementation that was not successful. This shows the value of approximate proximity matching and the supervised models in our approach compared to exact instance-based string matching in the baseline. The best performing models were the All-Prox and Attribute-Prox models which achieved better results than DS-Prox from our previous work [18] (described in Chapter 3) and better results than Name-Prox which are more common in other previous research. This means that combining both name-based meta-features and content-based meta-feature in a supervised model achieves best results in the schema matching pre-filtering task. For example, a good result can be achieved using the All-Prox model (combining all meta-feature types) with a threshold of 0.4 which achieves a recall rate of 85%, an efficiency gain of 73% and a lift score of 3.14. This means that the model is able to effectively propose most of the pairs of datasets for schema matching with the least effort possible (only proposing 27% of pairs for comparison), while achieving this with a performance that is three times better than naive random selection of dataset pairs for schema matching (as expressed by the lift score of 3.14 achieved by the All-Prox model).

Dataset Pairs Pre-filtering Precision Vs. Recall

As seen in Figure 4.13, the precision of the proximity models improved the performance of the schema matching pre-filtering as seen by the higher precision rates compared to the individual meta-features in Figure 4.15. By combining the meta-features in a supervised model we were able to achieve higher precision rates with the same recall rates, for example, a precision of 17% with a recall rate of 75% using the All-Prox model. This is better than the best achievable precision with the individual meta-features, which can achieve a precision of 4% with the same recall rate for the attribute level meta-feature type. However, we acknowledge that the precision rates are low for all types of models and meta-features. We can therefore conclude that our proposed proximity mining approach can only be used as an initial schema matching pre-filter which is able to prune unnecessary schema matching comparisons from further steps. Our approach can not be used for the final schema matching task because it will produce false positives. Therefore, dataset pairs should be further scrutinised with more comparisons to assess their schema similarity (as seen in Figure 4.1 bottom instance-based matching layer). Such comparisons use instance-based matching similar to our previous work [15] (described in Chapter 2).

5.5 Experiment 3: Computational Performance Evaluation

In this experiment, we evaluate the computational performance in terms of time and storage space consumption as follows:

- **Dataset**: OML02
- **Evaluation metrics**: Computational performance
- **Relationship evaluated**: all
- **Input**: All the pairs of datasets from OML02, a model (M_{cls-ds}) and the $PARIS$ schema matching prototype.
- **Output**: attribute-level and dataset-level metadata.
- **Goal**: test the comparable computational costs of running the different components of our proximity mining approach vs. traditional instance-based schema matching techniques. We show the value of pre-filtering by means of computational costs saving.
- **Description**: we take all the annotated pairs from OML02 and we do a complete run which collects the required meta-features and metrics, and

Table 4.13: The computational performance of our approach vs. the PARIS implementation in terms of time and storage space

Task	Timing	Average Time	Storage Space
Dataset Profiling	263,019ms (4:23 minutes)	1,295ms per dataset	31.25MB
Numeric Attribute Matching	1,184,000ms (19:44 minutes)	0.04ms per attribute pair	In memory
Nominal Attribute Matching	160,000ms (2:40 minutes)	0.04ms per attribute pair	In memory
Numeric Attribute Top Matching	3,250,000ms (54:10 minutes)	0.1ms per attribute pair 208ms per dataset pair (15,576 dataset pairs)	7MB
Nominal Attribute Top Matching	313,000ms (5:13 minutes)	0.08ms per attribute pair 19ms per dataset pair (16,290 dataset pairs)	2.33MB
Dataset-level All Aggregations of Attribute Similarities	500,000ms (8:20 minutes)	25ms per dataset pair (19,931 dataset pairs)	35MB
Dataset-level Name Matching	202ms (0 minutes)	0.01ms per dataset pair (20,503 pairs)	Part of Top Matching metadata
Dataset-level Content Matching	5,100ms (5.1 seconds)	0.25ms per dataset pair (20,503 pairs)	3.25MB
Attribute-level Name Matching, top pairs computation, and aggregation	1,018,663ms (16:58 minutes)	0.03ms per attribute pair (35,283,824 attribute pair) 51ms per dataset pair (19,931 dataset pair)	12.5MB
Apply the proximity models on the dataset pairs to score their similarities	1,665ms (1.66 seconds)	0.08ms per dataset pair (20,503 dataset pair)	8.5MB
PARIS Alignment Implementation	743,077,431ms (12,384:37 minutes)	36,241ms per dataset pair (0:36 minutes per dataset pair)	15,450MB (15.1GB)

we run the algorithms to compute $Sim(D_y, D_z)$. We measure the amount of time and storage space it takes to process the pairs.

We ran the experiments for our approach using a computer running on Linux Debian, 8GB main memory, a dual-core Intel i7 processor running at 2.4GHz and 4MB cache, Java v8 for the implementation of our algorithms, and Postgres database v9.5.12 for the metadata storage and management. For the PARIS baseline implementation, we used a server with more resources as recommended by the developers. The server runs on Linux Debian, Java v8, 24GB of memory and a quad-core processor at 2.4 GhZ and 4MB cache. We present the results below.

Results

We compare the computational performance by evaluating the amount of time and storage space for running our approach and the PARIS-based implementation with the DL sample OML02. The results can be seen in Table 4.13. We list the tasks from our approach and compare to the baseline in the last row.

We compute the time for each task, the average time it takes, and the storage space used. For the attribute matching, we keep the output in memory and do not materialise it. We only materialise top-matching attribute pairs.

Based on the results in Table 4.13, our approach needs a total of 112 minutes and 100MB storage space for the OML02 DL sample datasets of a total size of 2.1GB (i.e., 5% metadata space overhead). This is at least 2 orders of magnitude less than the time and space consumption of the baseline PARIS implementation. The most expensive steps in our approach were those for the numeric matching tasks as they were much greater in amount than nominal attributes. Still, our approach is more efficient in terms of computational performance and pre-filtering effectiveness as shown by our results.

5.6 Generalisability

In our experiments, we have used the OpenML DL to create a 10-fold cross-validation experimental setup. OpenML stores datasets representing heterogeneous subject-areas. Thus, we expect our proposed techniques to achieve similar results with different heterogeneous DLs. We tested our approach with different heterogeneous DL subsets covering randomly selected subject-areas in each cross-validation fold. This further improves the generalisability of our results as the results achieved proved to be stable between the different cross-validation folds. Therefore, our approach is recommended in the early-pruning and schema matching pre-filtering task in a DL environment with heterogeneous subject-areas. Under different settings, the data scientist should first test the performance of our approach on a test sample and then select the best performing cut-off thresholds accordingly. It is also crucial that the training samples selected for creating the supervised models are representative of the specific DL setting they are used for. We also note, that although our experiments were done over binary approximation for the $Rel(D_y, D_z)$ function in the ground truth due to the difficulty to find a ground truth with a similarity continuum, still our approach can be useful in dataset pairs ranking problems using the $Sim(D_y, D_z)$ continuous function.

6 Conclusion

We have presented in this chapter a novel approach for pre-filtering schema matching using metadata-based proximity mining algorithms. The approach is able to detect related dataset pairs containing similar data by analysing their metadata and using a supervised learning model to compute their proximity score. Those pairs exceeding a minimum threshold are proposed for more detailed, more expensive schema matching at the value-based granularity-level. Our approach was found to be highly effective in this early-pruning task, whereby dissimilar datasets were effectively filtered out and datasets with similar data were effectively detected in a real-life DL setting. Our approach achieves high lift scores and efficiency gain in the pre-filtering task, while maintaining a high recall rate. For future research, we will investigate the different techniques to improve the scalability of our approach by improving attribute level matching selectivity. We also want to investigate the possibility of detailed semantic schema matching at the attribute level. We will also investigate our proximity mining approach in effectively clustering the datasets into meaningful groupings of similarity.

Acknowledgements. This research has been partially funded by the European Commission through the Erasmus Mundus Joint Doctorate (IT4BI-DC).

Chapter 5

Automatic categorization of datasets using proximity mining

Birds that are alike attract each other, flock together, and group up.

— Old proverb

This chapter is an extended version of the work published as a paper in the proceedings of the 9th International Conference on Model and Data Engineering (MEDI) (2019) [16].
The layout of the paper has been revised and the content has been extended
DOI: https://doi.org/10.1007/978-3-030-32065-2_3

Springer copyright / credit notice:
© 2019 Springer, Cham. Reprinted with permission from Ayman Alserafi, Alberto Abelló, Oscar Romero, Toon Calders. Keeping the Data Lake in Form: DS-kNN Datasets Categorization Using Proximity Mining, MEDI, 2019.

Abstract

Given the recent growth in the number of datasets stored in data repositories, there has been a trend of using Data Lakes (DLs) to store such data. DLs store datasets in their raw formats without any transformation or preprocessing, with accessibility available using schema-on-read. This makes it difficult for analysts to find datasets that can be crossed and that belong to the same topic. To support them in this DL governance challenge, we propose in this chapter an automatic algorithm for categorizing datasets in the DL into pre-defined topic-wise categories of interest. We utilise a k-NN approach for this task which uses a proximity score for computing similarities of datasets based on metadata we collect. We test our algorithm on a real-life DL with a known ground-truth categorization. Our approach is successful in detecting the correct categories for datasets and outliers with a precision of more than 90% and recall rates exceeding 90% in specific settings.

1 Introduction

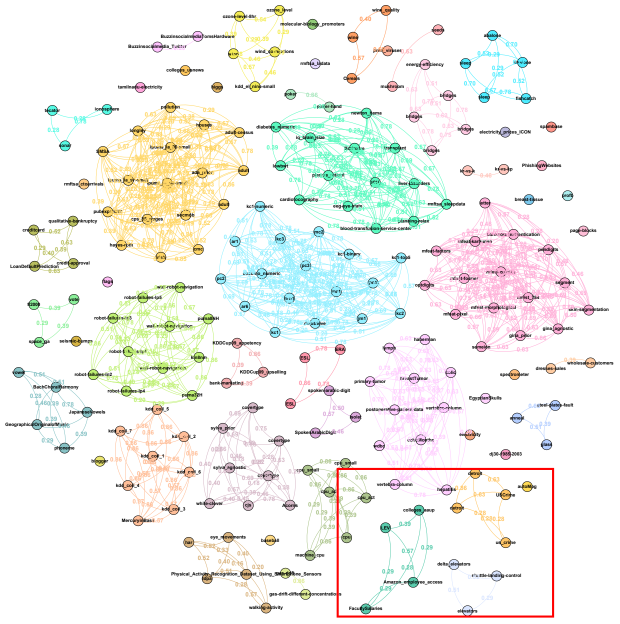
Today, a lot of data is generated covering different heterogeneous topics and domains. Those data are frequently stored as tabular datasets which describe different entities (in the rows) with information about them stored as attributes (in the columns). A collection of such raw datasets which are stored in their original schema without preprocessing or transformations is called a Data Lake (DL) [128]. Over its lifetime, a DL becomes very diverse and can cover different topics, making it difficult to find and retrieve relevant datasets for analysis. Therefore, it is a challenge for the users to govern the DL by detecting the groupings and underlying structures of similar datasets covering relevant topics for analytics [15, 18, 85].

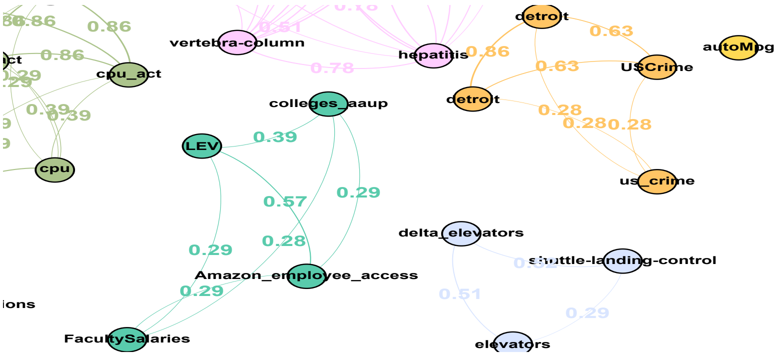
Automatic dataset categorization is concerned with the classification of datasets ingested in the data lake into topic-wise groupings based on their most-similar counterparts already existing in the DL. We use a lazy supervised machine learning approach.

To tackle this challenge, we propose an automated algorithm called DS-kNN to detect such groupings using k-nearest-neighbour (k-NN). The algorithm relies on collecting relevant metadata about the datasets when they are ingested, then we compute proximity models of dataset similarities based on supervised machine learning, and apply those models on new datasets to compute their similarity scores with datasets stored in the DL. Once we computed the similarities, we apply a k-NN algorithm to categorize the ingested datasets into the groupings already present in the DL or to classify them as outliers. An example of the expected results can be seen in Figure 5.1. Here, we visualise the DL as a *proximity graph* having datasets as nodes and edges connecting the nodes showing the similarity scores ($\mathbb{R} \in [0, 1]$) computed using the proximity model. Datasets in the same category (cluster) are shown in the same colour. We only show edges between datasets in the same category. In Figure 5.1 (a) we show an example of a complete DL proximity graph and in (b) we zoom-in on the specific part highlighted with a box for showing more details.

This chapter is an extended version based on our work in [16]. The main contributions of this chapter are:

1. We propose a kNN-based proximity mining algorithm for finding the correct categories for datasets based on existing categories in the DL. The improvement includes a new technique for finding top-k closest neighbours to propose a category for a dataset, where we search for the category having the highest aggregate total of similarity scores from the


(a)



(b)

Figure 5.1: A visualisation of the output from DS-kNN data lake (DL) categorization. A proximity graph shows the datasets as nodes and the proximity scores as edges between nodes. Fig.(a) complete DL and Fig. (b) a zoomed-in view highlighted by the red box in (a)

top-k nearest neighbours instead of the category with the majority of datasets in top-k.

2. We use different proximity mining models from Chapters 3 and 4 for assigning the similarity scores between dataset pairs including models which consider dataset-names together with dataset content, models which consider an ensemble of multiple dataset content meta-features, and models which use different aggregate proximity metrics from attribute comparisons. In addition, we also test models which combine all of these together.
3. We evaluate the algorithm in an expanded and improved experiment in a real-world setting to prove its effectiveness in assigning correct categories to new datasets ingested in the DL. This includes testing different k-values in full experimentation, treating all outliers (a dataset forming a category all by itself) as a single 'Outlier' category to prevent the macro-averaging evaluation metrics from being skewed towards outliers, and testing different category sizes against outliers as well (in addition to non-outlier datasets). This helps in evaluating if the algorithm is capable of equally finding the correct categories as well as detecting outliers without any matching categories in the DL.
4. We experimentally test the effect of different DL settings on the performance of our approach using different size categories and different annotated ground-truths. This includes a new validation experiment of the top performing DS-kNN parameters using a new ground-truth sample of datasets we collect.

In the rest of this chapter, we define the DL and the scenario we consider in Section 2, we present the DS-kNN algorithm in Section 3, then we test the algorithm on a real-life DL and we experiment with our algorithm in Section 4, we present related work in Section 5, and we conclude in Section 6.

2 Preliminaries

We consider a DL consisting of tabular datasets. Those are large heterogeneous repositories of *flat structured data* (i.e., CSV, web tables, spreadsheets, etc.). Such datasets are structured as groups of *instances* describing real-world entities, where each instance is expressed as a set of *attributes* describing the properties of the entity. We formally define a dataset D as a set of instances $D = \{I_1, I_2, ...I_n\}$. The dataset has a set of attributes $S = \{A_1, A_2, ...A_m\}$, where each attribute A_i has a fixed type, and every instance has a value of the right type for each attribute. We focus on two types of attributes: continuous

numeric attributes with real numbers and categorical ***nominal attributes*** with discrete values.

For each dataset, we collect different statistics about their content which we call *content meta-features*:

- **Nominal attributes:** their data profile mainly involves frequency distributions of their distinct values.
- **Numeric attributes:** their data profile mainly involves aggregated statistics like mean, min, max, and standard deviations.

We compute similarity scores between pairs of datasets $[D_a, D_b]$, as follows:

- $Sim(D_a, D_b)$: an estimation ($\mathbb{R} \in [0, 1]$) of the similarity based on the comparison of the *content meta-features* we collect about the datasets and their attributes. Typically, the information contained in highly similar datasets would overlap. An example would be a pair of datasets having similar numeric values and distribution of values, or nominal attributes having the same number of values. Alternatively, it could be based on *name string-similarity* between datasets and their respective attributes.

Scenario. We aim at governing the DL by incrementally maintaining the clusters of datasets defined for them. We consider the scenario where we initially have an existing DL for which we know all clusters of datasets based on their categories. However, given the dynamic nature of DLs, new datasets are frequently ingested. Thus, we need to compare these new datasets against the datasets already in the DL to find their similarity with them, and then to find their most appropriate category based on the similar datasets found in the DL, or to assign them to a separate category as an outlier.

This shapes the main problem for this chapter: given a collection of datasets in a DL and a newly ingested dataset, find all pairs of highly similar datasets, and based on their categories, assign a new category for the new dataset, or if no highly similar datasets are found then indicate that the dataset is an outlier. To compute the similarity between the datasets, we use a proximity model, which we call $M_{DS-Prox}$. We discuss how we create this model in Subsection 2.1.

The scenario discussed is visualized in Figure 5.2. Consider that there is a DL having a group of datasets (white circles) which have annotations of all their $Sim(D_a, D_b)$ relationships between pairs (as seen by the lines linking the datasets). Groups of datasets with linkages are segmented into categories (seen by the encompassing black circles). Those categories are the groupings of the subject-areas or domains-of-knowledge we have in the DL. We need to automatically use this DL and its known annotations to create a model $M_{DS-Prox}$ which can automatically annotate relationships of a new dataset D_i with $Sim(D_i, D_b)$ similarity scores. Therefore, we want to learn a model from

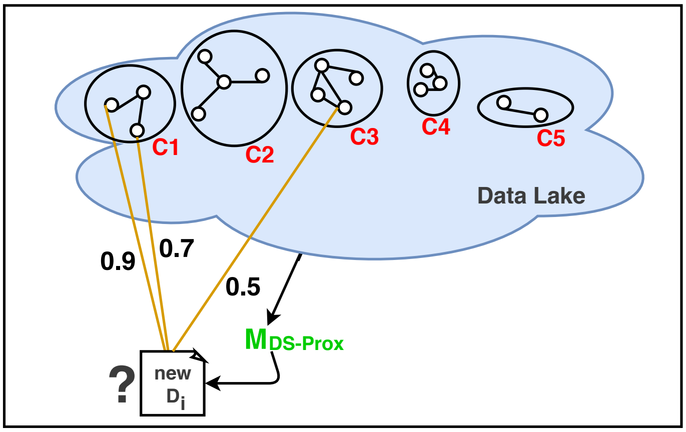


Figure 5.2: The data lake categorization scenario using k-NN proximity mining

the DL and apply it to estimate the similarity between a new dataset and all other datasets already in the DL, in order to find the top-k nearest neighbours.

Based on the similarity scores from the nearest neighbours we assign a category to D_i. The highlighted edges between the new dataset D_i and some nodes in the DL are those having the highest similarity scores computed by the model (in this case, we give an arbitrary example where we use top-3 nearest-neighbours). The numbers labelled on those edges are the computed $Sim(D_i, D_b)$ scores. In our proposed approach, each of those top nearest neighbours passing a *similarity threshold* (i.e., minimum similarity score) suggests its category as the correct one for D_i, and the category with the highest total $Sim(D_i, D_b)$ similarity scores should be proposed to the user, or if no such similar datasets are found (i.e., no dataset has a similarity score above the threshold) then D_i is marked as an outlier without any relevant category found. In the case of tied categories among the proposed ones from the top-k similar datasets, then all of them are proposed to D_i. In the example in Figure 5.2, category 'C1' would be proposed as the final category as it has an aggregated sum of similarity scores of $(0.9 + 0.7 = 1.6)$, as compared to only 0.5 for category 'C3'.

To learn the $M_{DS-Prox}$ model, we use supervised machine learning as described in Subsection 2.1.

2.1 Proximity Mining: Meta-features Metrics and Models

For all the datasets in the DL, we collect two metadata types: **A.Content-based** and **B. Name-based** meta-features. The name-based techniques are the most commonly used metadata in previous research [52, 85, 111, 121]. In our DS-kNN approach, we also use our proposed content-based meta-features complementing the name-based metadata when computing similarity scores (see Chapters 3 and 4). Such content meta-features include data profiling statistics about the content of the datasets. Thus, we use two types of metadata for similarity computations:

- *Name-based metadata:* the naming of datasets and their attributes.
- *Content-based metadata:* profiling statistics about the data stored in the datasets. The collected meta-features (described in Table 4.3) include statistics concerning all attributes collectively, the attribute types found and the overall number of instances, in addition to attribue-level aggregated meta-features in Table 3.1. Those form a concise list of meta-features that have been proved in our work [18] (see Chapter 3 Section 4.1) to be effective in predicting related datasets with similar schemata and stored information. Our purpose for those meta-features is to describe the general structure and content of the datasets for an approximate comparison using our proximity mining classification models.

For content meta-features, we construct the $M_{DS-Prox}$ model using the proximity mining approach from Chapters 3 and 4. First, we compute all proximity metrics for each meta-feature m_i between each pair of datasets $[D_a, D_b]$. We compute this for all dataset pairs $[D_a, D_b]$ in the training sample.

For the attribute-names, we compute the proximity metrics between the attribute pairs from the different datasets and then compute an aggregation that generates a single value for dataset pairs $[D_a, D_b]$ using: minimum distance, maximum distance, and average. Note that to compute distances from name-based metadata (instead of content-based meta-features), we use the Levenshtein distance as a standard string comparison metric [86].

Once we have the metadata collected and their distances computed, we feed them to a supervised machine learning algorithm to produce a classification model which identifies those dataset pairs in the same assigned category. This creates the proximity mining model to compute similarity scores. For this initial training sample of datasets we have in the DL, a data analyst should have incrementally assigned a category cluster to each dataset based on their topics. The target variable for those classifiers is a binary value whether the datasets in the pair belong to the same category or not.

We use the two top performing ensemble learning algorithms from chapter 3 to learn the model, which are the boosting machine learning algorithms AdaBoost by [49] and LogitBoost by [50]. We also use the top performing Random Forest [29] 'All-Prox' model from chapter 4 which we call here 'All-Ensemble' model. Those algorithms were compared in our previous work to other algorithms and were found to be the best in finding related schemata. The positive-class distribution produced by the ensemble model is used as the similarity score $Sim(D_a, D_b)$ [127].

Finally, we apply the learnt $M_{DS-Prox}$ model on pairs of one new ingested dataset and each existing datasets in the DL to generate the similarity scores. We compare the score against a minimum threshold like in Equation 5.1. Only pairs passing the threshold are considered as candidate top-k nearest neighbours to a dataset in our DS-kNN algorithm (we discuss this in detail in Section 3).

$$Top(D_a, D_b) = \begin{cases} 1, & Sim(D_a, D_b) > c_{rel} \\ 0, & \text{otherwise} \end{cases} \tag{5.1}$$

3 DS-kNN: a Proximity Mining Based k-Nearest-Neighbour Algorithm for Categorizing Datasets

Algorithm 4: DS-kNN Categorization of a dataset ingested in a Data Lake

```
Input: A new ingested dataset $D_a$, each existing dataset $D_b$ in the data lake $DL$, Dataset-level
       meta-features distance metrics $MF$ for each pair of datasets $\{D_a, D_b\}$, the category $Cat_{D_b}$ for
       each existing dataset in the $DL$, the classification model $M_{ds-prox}$, algorithmic parameters:
       the number $k$ of nearest neighbours, and the similarity score threshold $c_{rel}$
Output: The set $SP$ of the ingested dataset and its similarity scores $Sim(D_a, D_b)$ and category $Cat_{D_b}$
        for each pair $\{D_a, D_b\}$ passing $c_{rel}$, the set $SP\text{-}Top$ of top matching $k$ datasets and their
        categories, the assigned category for the new dataset $Cat_{D_a}$
1  $SP \leftarrow \varnothing$;
2  $SP\text{-}Top \leftarrow \varnothing$;
3  foreach $\{D_a, D_b\} \subset DL$ and $a \neq b$ do
4      $[D_a, D_b, Sim(D_a, D_b)] = M_{ds-prox}(MF_{\{D_a, D_b\}})$;
5      if $Sim(D_a, D_b) > c_{rel}$ then
6          $SP \leftarrow SP \cup \{[D_a, D_b, Sim(D_a, D_b), Cat_{D_b}]\}$;
7  $SP\text{-}Top$ = Top-k_Nearest_Neighbours($SP$, k); \\Retrieve the subset of the highest ranking k-pairs
     by similarity score
8  $Cat_{D_a} = Top\text{-}category(SP\text{-}Top)$; \\Get category with highest sum of similarity scores from Top-k
9  if $(Cat_{D_a} = NULL)$ then
10     $Cat_{D_a} =' Outlier'$;
```

We propose an algorithm for computing the categories of an ingested dataset as described in the scenario in Section 2. After learning the classification model $M_{DS-Prox}$, we apply the classifier to each new pair $[D_a, D_b]$ where D_a is any new ingested dataset and D_b is each of the existing datasets in the DL, in order to obtain the similarity score $Sim(D_a, D_b)$ with all datasets in the DL. Then, we apply k-NN in our proposed DS-kNN Algorithm 4 to compute the category for the new dataset. k-NN was already successful in similar categorization problems, like free-text document categorization [55], image categorization [78], etc. We also select k-NN as our supervised classification technique rather than other classification methods like Naïve Bayes and decision trees previously used with other applications like text document classification [19], as we preferred a generic lazy learning approach that does not need to be trained a priori for a specific list of pre-defined categories. Rather, k-NN facilitates categorization of datasets without retraining when the DL is dynamically growing with continuous addition of datasets belonging to new categories. Other techniques will require retraining whenever a new category is added to the DL.

First, our algorithm applies in Lines 3-6 the $M_{DS-Prox}$ model on all the dataset pairs for the new dataset D_a to compute their similarity scores, and those passing the minimum threshold are stored in the set SP. To improve efficiency, a heap data structure could be used to store the datasets with their similarity scores for quick search and retrieval of top-k nearest-neighbours. The next step in Line 7 involves finding those top-k nearest-neighbours which are existing datasets in the DL with the highest similarity scores to D_a. Finally, we assign a category to the new dataset in Line 8 based on the category with the highest aggregated sum of similarity scores from the top-k nearest neighbours. If no top-k nearest neighbours that satisfy the given minimal similarity are found in Lines 9-10 then the dataset is marked as an 'outlier' with no proposed category.

The algorithm has the following parameters as input:

- **The number of neighbours (k):** the top-k number of nearest neighbours which our algorithm uses to predict the new category for an ingested dataset.
- **The proximity model ($M_{ds-prox}$):** this is the proximity mining model created using our approach described in Section 2. We use different models depending on the metadata, i.e., content-based, dataset-name based or attribute-name based.
- **The similarity threshold (c_{rel}):** the minimum allowed similarity score to consider a dataset pair as candidate nearest neighbour. This threshold helps in detecting outliers as it eliminates noise by preventing any dataset with a low similarity score from being proposed in the top-k nearest neighbours and proposing its category incorrectly.

Different similarity thresholds lead to a different performance of the algorithm. A higher threshold prevents suggesting irrelevant categories by eliminating dataset pairs with a similarity score below the threshold, therefore helping in improving the capability of the algorithm in detecting outliers (without any suggested categories in the DL). On the other hand, a lower threshold allows more dataset pairs with a lower similarity score to suggest their categories therefore preventing the algorithm from missing any relevant category for a new dataset. As a consequence of this trade-off, we test multiple threshold values in our experiments to discover the best one to use. We also note that generally a lower value for the parameter k makes the algorithm more susceptible to outliers and overfitting, while a higher value leads to underfitting. An optimum value that also tackles this trade-off needs to be empirically discovered too.

Different proximity models $M_{DS-Prox}$ are expected to lead to a different performance of the algorithm as different meta-feature comparisons between datasets will always lead to different similarity scores between them, depending on what is being compared. For example, if we only consider attribute names in the model, then datasets with common values stored in their attributes yet having different naming will lead to a low similarity score, and vice versa. Combining both proximity metrics together might lead to a better estimation of similarity. Therefore, we need to discover which proximity models are more effective with the DS-kNN algorithm and are more accurate in dataset categorization.

The complexity of our algorithm is quadratic in the number of objects compared (attributes or datasets), and therefore runs in polynomial time. For each dataset, we first run some data profiling to analyse the schema and to collect the meta-features in Table 3.1, which run in linear time in the number of datasets. The comparison of the dataset pairs using $M_{ds-prox}$ is the step leading to the overall quadratic complexity of the algorithm, however, we note that this step is very cheap to compute as it involves computing the proximity metrics and feeding them to the scoring model $M_{ds-prox}$ which on average takes less than 0.01 millisecond to apply on a single dataset pair in the experiments (on a single quad-core server running on Linux and 8GB of RAM).

4 Experimental Evaluation

We test our proposed categorization algorithm on a real-life DL. We describe the datasets used, the experimental setup and our results with a detailed discussion of the performance of DS-kNN.

4.1 Dataset: OpenML DL Ground-truth

We created a ground-truth based on manual annotations of two samples of datasets from a real-life DL called OpenML[1]. It consists of different datasets covering heterogeneous topics, each having a name and a description. An example of the categories found in the ground-truth are summarised in Figure 5.1.

Firstly, we collect a set of 118 datasets from OpenML by topic using 11 keywords-search over the dataset descriptions, e.g., “Disease”, “Cars”,“Flights”,

[1]http://www.openml.org

"Sports", etc. A domain expert and one of the authors collaborated to manually label the datasets with their topic. They assigned a topic to each dataset as its category based on the specific entity described by the instances inside it (i.e., the item or object being described in the rows of the tabular data). For example, if each instance describes some health measurements related to patients, they assigned the dataset to a "patient health measurements" category, etc. The annotators examined the textual descriptions of the datasets which include information about how the data was collected and an English description of the content stored in the datasets. We use this sample of annotated datasets in Section 4.4 as the validation set used to evaluate the top performing DS-kNN parameters on them.

We also collected another independent sample of datasets from OpenML to use them in the training of the proximity models and parametric trials of DS-kNN in Section 4.3. We scraped OpenML to extract another set of datasets not included in the above sample and that have a textual description of more than 500 characters. The descriptions helped the manual annotators in deciding on the assigned topic for each dataset. Out of the 514 datasets retrieved, we selected 203 with meaningful descriptions (i.e., excluding datasets whose descriptions do not allow to interpret its content and to assign a topic). Both of the annotators collaborated to manually label the datasets with their topic. The datasets were labelled by both their *broad* **subject** (e.g., '*social demographics*') and the more *specific* **entity** described by the instances stored in them (e.g., '*citizens census data*')[2].

Table 5.1 shows the number of datasets per category assigned based on topic grouping type and each sample. We only show the top 10 categories found by size for each grouping. The total number of categories is also given and the number of categories bigger than a specific size (i.e., with at least this number of members, for example 8+ means a category with at least 8 member datasets), and the number of outliers (datasets with their own specific category without any other members). As can be seen in the table, the datasets in the DL we use in the experiments cover heterogeneous topics and different category sizes.

4.2 Experimental Setup

Our goal is to test the performance of the DS-kNN algorithm in correctly proposing the category to datasets. We compare the performance of the DS-Prox content-based models when applied in DS-kNN against the baseline models of dataset-names and attribute-names, which are the commonly used metadata in previous work [52, 85, 111, 121] (see Section 5). We implement DS-kNN based on those different models in Java using a PostgreSQL database

[2]The interested reader can download the two annotated datasets from GitHub at https://github.com/AymanUPC/ds-knn

Table 5.1: A description of the OpenML categorized datasets collected. Datasets are categorized by subject and by entity for the 203 ds sample, or by entity for the 118 ds sample.

Sample	Category Type	No. of Categories	Categories by Type	Categories by Size	Outliers
203 ds	Subject	53	Computer Software (17), Social Demographics (17), Image Recognition (16), Health (14), Robot (11), Disease (11), Natural Water (8), Ecology (8), Computer Hardware (6), Motion Sensing (5)	8+ members (8), 5+ members (14), 3+ members (25)	21
203 ds	Entity	77	Computer Software defects (16), Citizens Census Data (12), Digit Handwriting Recognition (12), Diseases (11), Robot Motion Measurements (11), Health Measurements (10), Chemical Contamination (8), Plantation Measurements (8), CPU Performance Data (6), Animal Profile (5)	8+ members (8), 5+ members (12), 3+ members (21)	47
118 ds	Entity	35	Diseases (46), Plants (13), Cars (10), Citizen Census Data (4), University Faculty Evaluations (4), Stock Prices (3), Employee HR Records (2), Pollution Measures (2), Food Nutritional Facts (2), Baseball Player (2)	8+ members (3), 5+ members (3), 3+ members (6)	18

as its backend for storing the metadata, and we feed it with the datasets from the OpenML DL. We test the algorithm using different values for $k \in \{1,3,5,7\}$. To test the generalizability and adaptability of DS-kNN under different DL settings, we also conduct trials with the algorithm under the following different settings which affect the ground-truth used in the training and testing of $M_{ds-prox}$:

- **Different category sizes:** we test DS-kNN with all the datasets (including outliers where category size is just 1 dataset) and with categories that at least contain the following number of members (3,5,8). We test different sizes of categories to check if the algorithm is affected by them. We test all the datasets in the sample from different category sizes (including those in smaller categories), but we only consider top-k nearest neighbours from the categories of the sizes specified by this parameter. This way, we also consider outliers if they are not assigned to any category in the required sizes.

- **Different ground-truth types:** We test the algorithm with (1) the broad subject-based categories from the 203 ds sample (Section 4.3), (2) the more detailed entity-based categories from the 203 ds sample (Section

4.3), and (3) we validate the top performing models on the 118 ds entity-based sample (Section 4.4).

For each DL setting, we compare the performance of the DS-kNN algorithm using the following input parameters:

- **k:** the number of top ranking nearest neighbours considered for computing the assigned category for a dataset. We use the following values: 1,3,5,7.
- **Different models ($M_{ds-prox}$):** we test the different models generated by different metadata, which are (1) Dataset Name, (2) Attribute Name (minimum, maximum and average aggregations) and (3) DS-Prox Content. We also test ensemble models which combine all those different types of meta-features in 'All-Prox'.
- **Different similarity thresholds:** we use different thresholds for $Sim(D_a, D_b)$ including (0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9).

For the ensemble models, we train a Random Forest based supervised learning model [29, 127] using the same setup in our previous work [18, 17] (see Chapters 3 and 4). The difference is the input which consist of all the distance metrics generated for all the types of meta-features. We also train the models on a subset based on the type of meta-features used, which leads to the following set of models:

- Combined all-ensemble ('All-Prox'): uses all the meta-features together including dataset name distance, attribute name distances (minimum, maximum and average), the dataset content meta-features from Table 3.1, and the attribute content meta-features from Table 4.3. This is the most comprehensive model.
- Name-based: uses only the name based meta-features from the dataset name and the attribute names.
- DS-Prox-and-Dataset-Name: uses both the dataset-name metadata and the DS-Prox content-based meta-features (see Table 3.1) to train a single ensemble model.

We test the different combinations of the above parameters and settings which resulted in the execution of a total of 2880 independent trials for the different settings (category sizes and ground-truth types) and algorithmic parameters (k, $M_{ds-prox}$, similarity threshold). We utilise a leave-one-out experimental setup to test our categorization algorithm in each trial, as seen by the example in Figure 5.2, so for each experimental trial we train the model under the same settings and with the same parameters 203 times, where for each run we keep a single dataset out from the training of the model and treat it as the new test dataset. We apply the proximity model on the test dataset with all dataset pairs found in the DL, and we run our algorithm to compute its allocated category or to mark it as an outlier. Similarly, for the 118 datasets validation sample, we test the top performing models 118 times. We apply Algorithm 4 on the test dataset and we find the top categories it should be allocated to. The goal is to maximise the number of correctly assigned categories based on top-k nearest neighbours.

To evaluate the effectiveness, we consider our algorithm as an example of a multi-class classification problem. We evaluate whether each dataset gets assigned the correct category based on *top-k nearest neighbours*. We compute the number of correctly annotated categories and outliers by measuring recall, precision and F1-scores which are commonly used for evaluation in similar settings [9, 11, 85]. We compute the F1 score as the harmonic mean of the recall and the precision [86]. The evaluation metrics are described in Equations (5.2),(5.3) and (5.4) respectively. Here, TP means true-positives which are the datasets correctly classified to their category. FN are false negatives, and FP are false positives. We compute the evaluation metrics per category and average the final scores from all categories to achieve macro-averaging scores [86]. For example, consider we have in the ground-truth two categories $C1$ and $C2$ consisting of 10 datasets each. $C1$ had 9 TPs and 1 FP (i.e. a dataset from a different category incorrectly assigned to it by DS-kNN) while $C2$ had 8 TPs and 2 FPs, therefore they will have a precision of 0.9 and 0.8 respectively. Therefore, the macro-precision will be $\frac{0.8+0.9}{2} = 0.85$.

$$recall = \frac{TP}{TP + FN} \tag{5.2}$$

$$precision = \frac{TP}{TP + FP} \tag{5.3}$$

$$F1\text{–}score = 2 \times \frac{(Recall \times Precision)}{Recall + Precision} \tag{5.4}$$

4.3 Results

We present the precision-recall curves from our experiments for $k = 1$ in Figure 5.3, $k = 3$ in Figure 5.4, $k = 5$ in Figure 5.5 and $k = 7$ in Figure 5.6. Each graph plots the macro-averaging performance resulting from leave-one-out cross-validation of a specific model for a specific ground-truth type from the 203 ds sample and different category sizes (which are labelled above the chart). For all our results we use percentages for the performance metrics. Here, we plot recall against precision for each of the different model types used in DS-kNN and the different minimum category sizes we use in the experiment. The numbers annotated on the points indicate the similarity

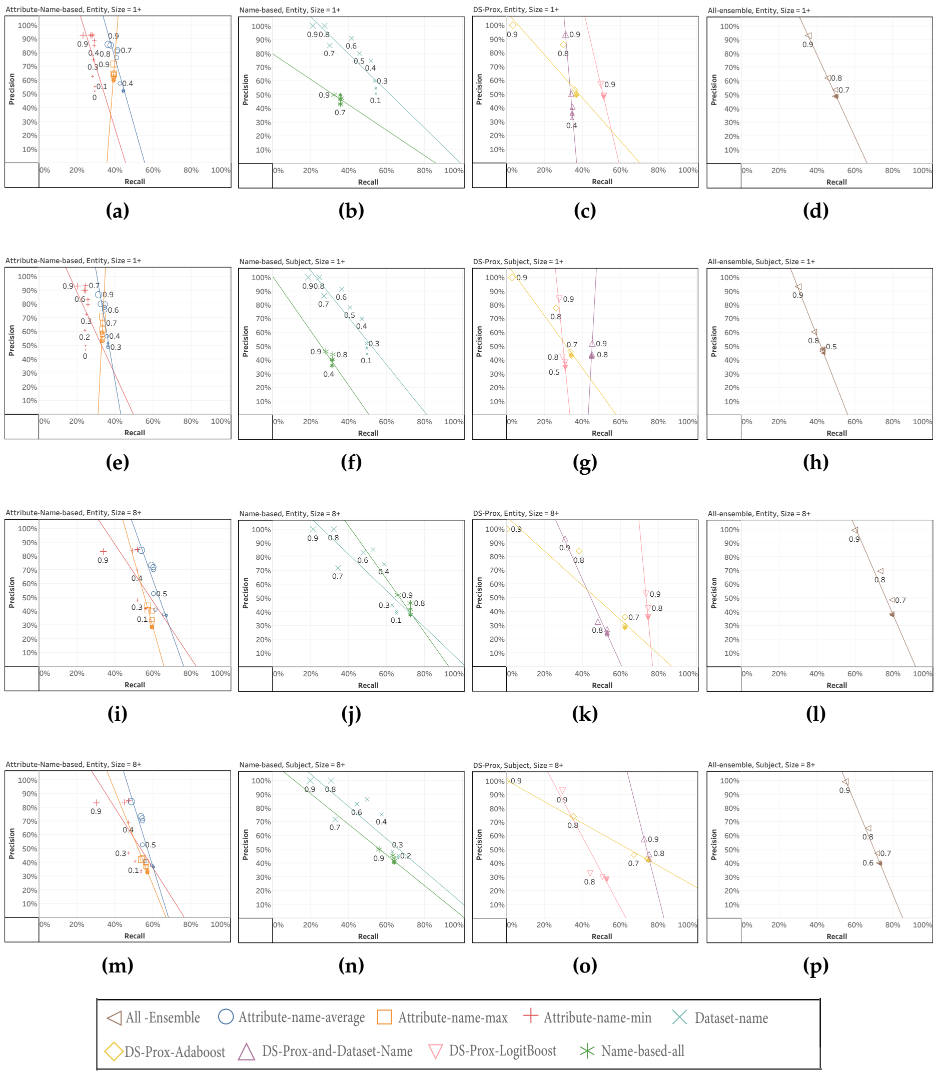


Figure 5.3: Performance of DS-kNN using **k=1**, different models, different ground-truths, and different category sizes

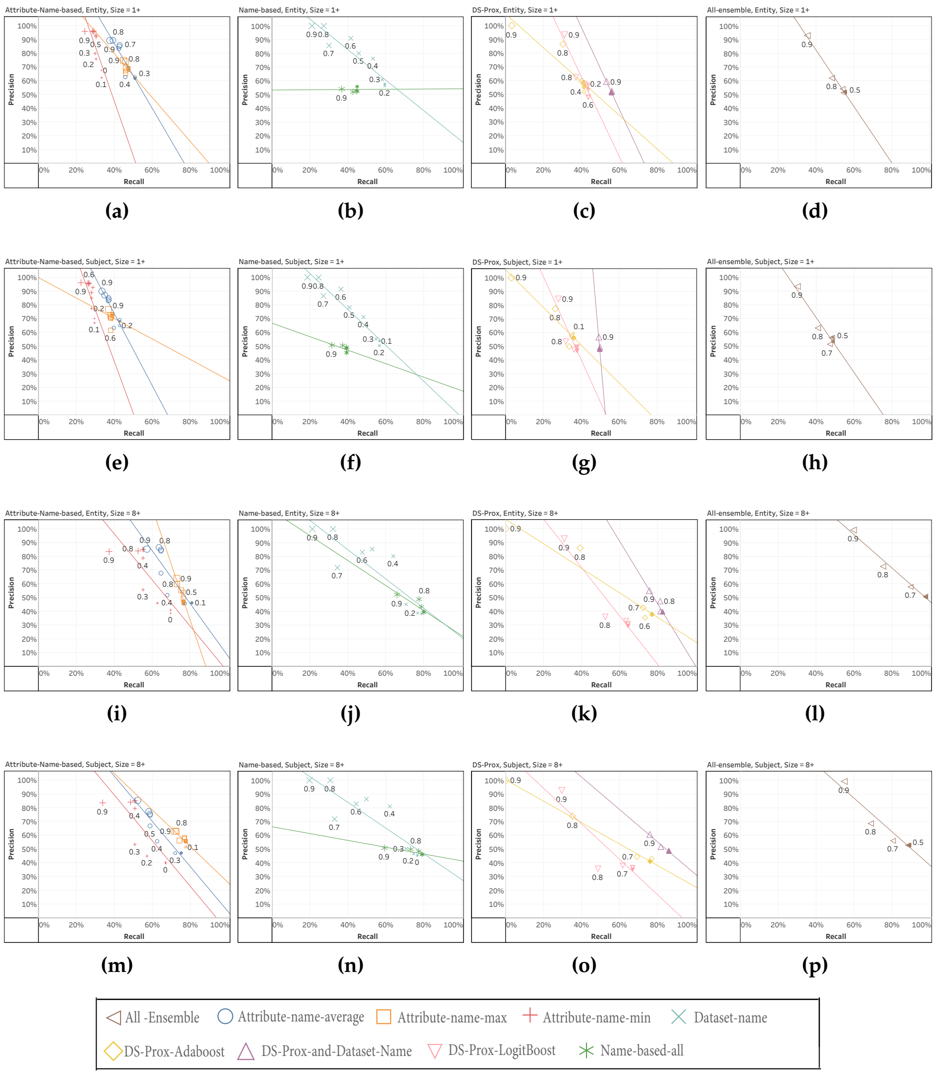


Figure 5.4: Performance of DS-kNN using **k=3**, different models, different ground-truths, and different category sizes

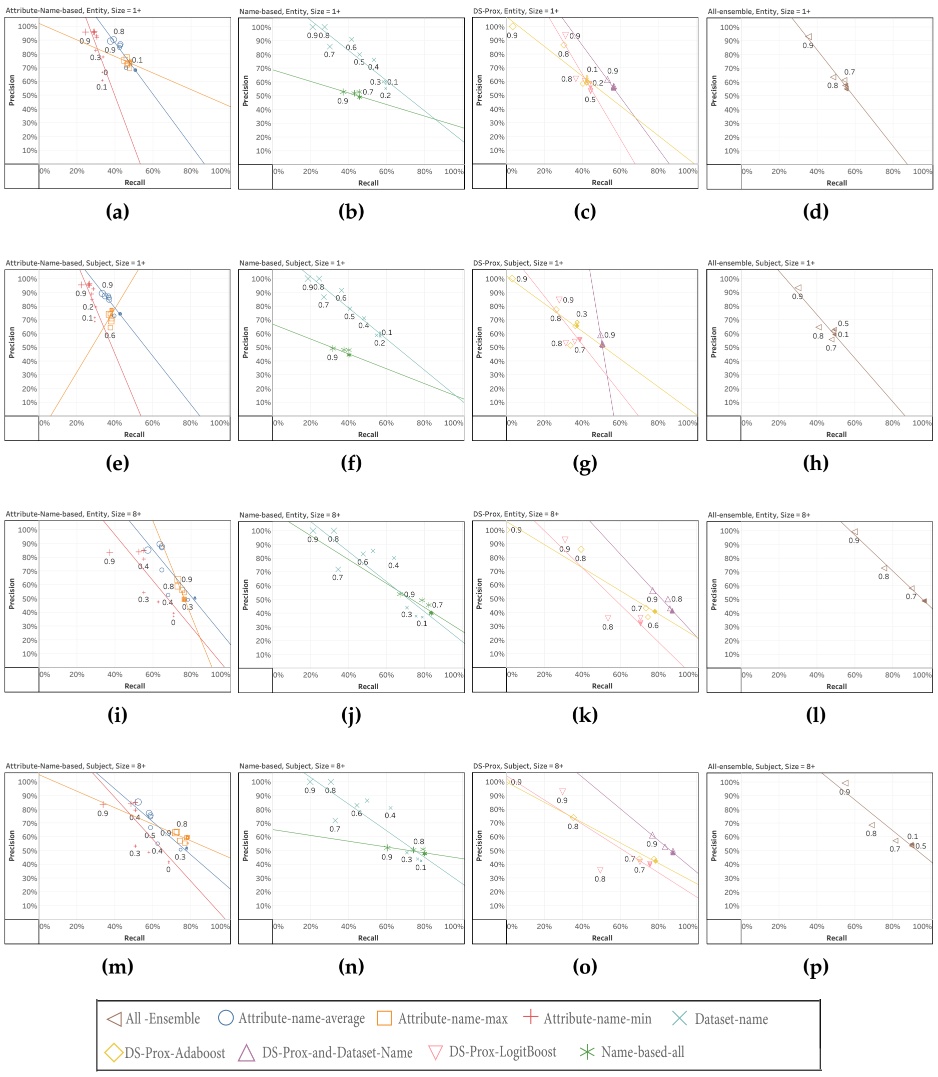


Figure 5.5: Performance of DS-kNN using **k=5**, different models, different ground-truths, and different category sizes

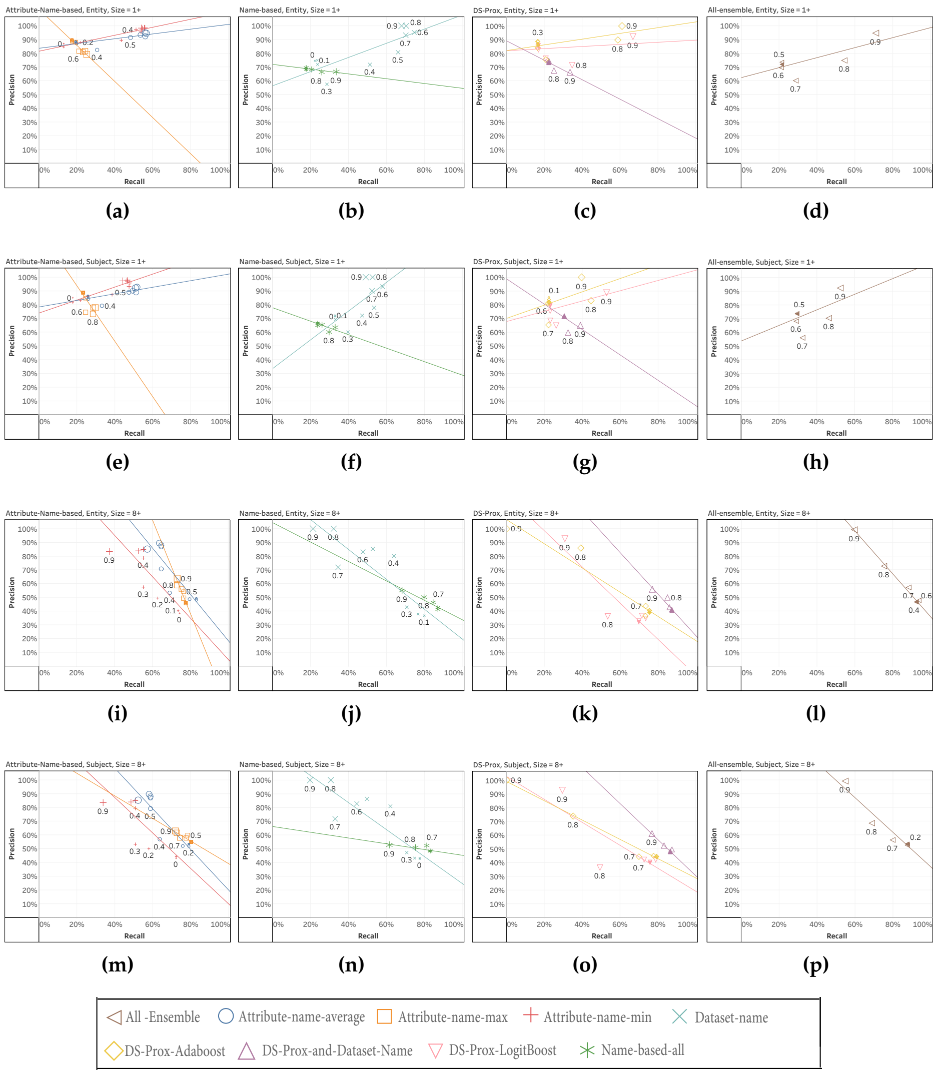


Figure 5.6: Performance of DS-kNN using **k=7**, different models, different ground-truths, and different category sizes

threshold (also indicated by the size of the points, where bigger size indicates a higher similarity threshold). We show the results for the non-restricted (category size = 1+) which includes outliers, and only the biggest category sizes (category size = 8+). Each model type has a different symbol and colour. For attribute-name based models in the first column, we show the results using the average, minimum and maximum aggregations. In the second column, we show the results of the dataset-name based model and also the combination of all the attribute-name based aggregations with the dataset name in a single ensemble model we call *name-based-all*. In the third column, we present the content-based DS-Prox models, where we include the AdaBoost-based model and the LogitBoost-based model in addition to the model that combines the meta-features from Table 3.1 and the dataset-name within a single Random Forest model (we call it *DS-Prox-and-Dataset-Name*). Finally, we present in the fourth column the model showing the combination of all meta-features into a single ensemble model we call *All-Ensemble*. We also present in Tables 5.2 - 5.5 the evaluation metrics for the top performing parameters for DS-kNN (in terms of F1-scores) for each category size for the different ground-truth types and for each model type. We highlight the top 3 performing model types for each category size and ground truth type.

As could be seen from the results, DS-kNN performs comparatively well with the attribute-name and the DS-Prox content-based models for category size 1+, while combining all the meta-features in a single ensemble model performs the best. There is a trade-off between the similarity threshold used and recall, the higher the threshold the lower the recall and the higher the precision, and vice versa. However, there is an optimum similarity threshold that can perform better than all others, e.g., the *All-Ensemble* model with a threshold of 0.9 in Figure 5.6d.

For larger category sizes the DS-Prox content models and the All-Ensemble are better in assigning the correct categories. For example, in Figure 5.3l, All-Ensemble leads to a precision of about 70% and recall of about 80% for category sizes of at least 8 members and the entity-based ground-truth, while attribute-name model can only achieve in Figure 5.3i a maximum recall of only about 70% with a precision of about 60%. Dataset-name based model performs a similar recall and slightly better precision. The results also indicate that the choice of the similarity threshold can affect the performance of DS-kNN.

In general, DS-kNN performs better with bigger category sizes than smaller category sizes as it becomes easier for the algorithm to find relevant top-k nearest neighbours. However, it is still good in detecting outliers and other categories as seen in Table 5.2 for the performance for *'min. category size'* = 1, for example a recall of 75% and precision of 95% for dataset-name based model. The dataset-name model performs better in detecting outliers as seen from this result. The DS-kNN algorithm performed equally good with both ground-truth types under the same settings and with the same parameters,

Table 5.2: The evaluation of DS-kNN for the minimum category size of 1+ with the different model types and ground-truth types. For each setting, we only show here the best performing parameters based on F1-scores.

Model Type	**Ground Truth Type**	**Min. Category Size**	**k**	**Similarity Threshold**	**Recall**	**Precision**	**F1-score**
Dataset Name	entity	1	7	0.6	74.7%	95.2%	83.7%
All-Ensemble	entity	1	7	0.9	71.0%	94.7%	81.2%
DS-Prox LogitBoost	entity	1	7	0.9	66.8%	92.3%	77.5%
DS-Prox AdaBoost	entity	1	7	0.9	61.0%	100.0%	75.8%
Attribute Name Min.	entity	1	7	0.7	55.9%	98.5%	71.3%
Attribute Name Avg.	entity	1	7	0.9	56.6%	94.6%	70.8%
Attribute Name Max.	entity	1	5	0.1	48.0%	75.5%	58.7%
DS-Prox-and-Dataset-Name	entity	1	5	0.9	53.2%	61.4%	57.0%
Name-based-all	entity	1	3	0.3	44.9%	56.0%	49.8%
Dataset Name	subject	1	7	0.6	58.0%	93.1%	71.4%
All-Ensemble	subject	1	7	0.9	52.4%	92.3%	66.8%
Attribute Name Avg.	subject	1	7	0.8	51.7%	92.8%	66.4%
DS-Prox LogitBoost	subject	1	7	0.9	52.8%	88.8%	66.3%
Attribute Name Min.	subject	1	7	0.4	47.9%	95.8%	63.9%
DS-Prox AdaBoost	subject	1	7	0.8	44.6%	83.1%	58.1%
DS-Prox-and-Dataset-Name	subject	1	5	0.9	49.7%	59.1%	54.0%
Attribute Name Max.	subject	1	5	0.4	38.3%	77.4%	51.3%
Name-based-all	subject	1	5	0.7	40.2%	48.0%	43.7%

Table 5.3: The evaluation of DS-kNN for the minimum category size of 3+ with the different model types and ground-truth types. For each setting, we only show here the best performing parameters based on F1-scores.

Model Type	Ground Truth Type	Min. Category Size	k	Similarity Threshold	Recall	Precision	F1-score
Attribute Name Avg.	entity	3	7	0.3	59.3%	75.3%	66.3%
Attribute Name Max.	entity	3	5	0	54.0%	76.0%	63.2%
All-Ensemble	entity	3	7	0.8	53.6%	66.5%	59.4%
DS-Prox-and-Dataset-Name	entity	3	7	0.7	62.9%	55.1%	58.7%
Dataset Name	entity	3	3	0.4	49.9%	70.1%	58.3%
Attribute Name Min.	entity	3	7	0	51.0%	63.6%	56.6%
DS-Prox Logit-Boost	entity	3	5	0	49.3%	55.7%	52.3%
Name-based-all	entity	3	7	0.6	56.6%	45.6%	50.5%
DS-Prox AdaBoost	entity	3	3	0.1	49.0%	50.7%	49.8%
Attribute Name Avg.	subject	3	7	0.1	53.1%	78.8%	63.4%
Attribute Name Max.	subject	3	7	0	45.7%	82.3%	58.8%
All-Ensemble	subject	3	7	0.5	56.6%	61.0%	58.7%
DS-Prox-and-Dataset-Name	subject	3	7	0.7	57.6%	57.8%	57.7%
Dataset Name	subject	3	5	0.2	58.8%	53.3%	55.9%
Name-based-all	subject	3	7	0.7	49.4%	54.4%	51.8%
Attribute Name Min.	subject	3	5	0.2	38.4%	78.6%	51.6%
DS-Prox AdaBoost	subject	3	5	0.4	40.7%	69.2%	51.2%
DS-Prox Logit-Boost	subject	3	5	0.3	41.5%	61.7%	49.6%

Table 5.4: The evaluation of DS-kNN for the minimum category size of 5+ with the different model types and ground-truth types. For each setting, we only show here the best performing parameters based on F1-scores.

Model Type	**Ground Truth Type**	**Min. Category Size**	**k**	**Similarity Threshold**	**Recall**	**Precision**	**F1-score**
Attribute Name Avg.	entity	5	7	0.8	57.4%	93.2%	71.0%
Attribute Name Max.	entity	5	5	0.9	69.2%	69.9%	69.6%
All-Ensemble	entity	5	3	0.9	51.5%	99.5%	67.9%
Dataset Name	entity	5	5	0.4	59.6%	77.8%	67.5%
Attribute Name Min.	entity	5	1	0.5	51.8%	90.8%	65.9%
DS-Prox-and-Dataset-Name	entity	5	5	0.9	69.6%	61.1%	65.1%
Name-based-all	entity	5	5	0.8	68.4%	48.1%	56.5%
DS-Prox AdaBoost	entity	5	5	0.5	63.7%	46.6%	53.8%
DS-Prox LogitBoost	entity	5	3	0.9	35.5%	95.1%	51.7%
Attribute Name Avg.	subject	5	7	0.1	66.2%	66.4%	66.3%
Attribute Name Max.	subject	5	5	0.4	59.8%	72.9%	65.7%
All-Ensemble	subject	5	7	0.6	72.0%	59.1%	65.0%
DS-Prox-and-Dataset-Name	subject	5	7	0	72.5%	57.7%	64.2%
Dataset Name	subject	5	5	0.4	55.7%	72.1%	62.9%
Attribute Name Min.	subject	5	1	0.5	47.6%	92.2%	62.8%
Name-based-all	subject	5	7	0	66.7%	46.8%	55.0%
DS-Prox AdaBoost	subject	5	5	0.4	50.3%	53.6%	51.9%
DS-Prox LogitBoost	subject	5	5	0.7	54.9%	45.7%	49.9%

Table 5.5: The evaluation of DS-kNN for the minimum category size of 8+ with the different model types and ground-truth types. For each setting, we only show here the best performing parameters based on F1-scores.

Model Type	**Ground Truth Type**	**Min. Category Size**	**k**	**Similarity Threshold**	**Recall**	**Precision**	**F1-score**
All-Ensemble	entity	8	5	0.9	59.7%	99.2%	74.6%
Attribute Name Avg.	entity	8	7	0.7	64.6%	87.9%	74.5%
Dataset Name	entity	8	7	0.4	64.0%	80.2%	71.2%
Attribute Name Max.	entity	8	5	0.9	73.0%	64.0%	68.2%
Attribute Name Min.	entity	8	7	0.5	55.1%	85.0%	66.9%
DS-Prox	entity	8	5	0.9	76.9%	55.9%	64.7%
Name-based-all	entity	8	7	0.8	79.8%	50.2%	61.6%
DS-Prox AdaBoost	entity	8	7	0.7	73.4%	43.8%	54.8%
DS-Prox LogitBoost	entity	8	7	0.7	71.4%	36.6%	48.4%
All-Ensemble	subject	8	7	0.9	54.8%	99.2%	70.6%
Attribute Name Avg.	subject	8	7	0.7	58.8%	88.0%	70.5%
Dataset Name	subject	8	7	0.4	62.1%	81.0%	70.3%
DS-Prox-and-Dataset-Name	subject	8	5	0.9	76.9%	61.2%	68.1%
Attribute Name Max.	subject	8	5	0.4	78.4%	60.0%	68.0%
Name-based-all	subject	8	7	0.7	81.2%	52.4%	63.7%
Attribute Name Min.	subject	8	1	0.5	50.8%	85.1%	63.6%
DS-Prox AdaBoost	subject	8	7	0.4	79.3%	44.6%	57.1%
DS-Prox LogitBoost	subject	8	7	0.6	78.8%	42.0%	54.8%

yet slightly better with the more specific entity-based ground-truth with small category sizes and outliers. This indicates the adaptability of DS-kNN to different DL settings and properties.

In general, DS-kNN and All-Ensemble models perform better with bigger category sizes. For example, they can achieve 90% recall and more than 50% precision for $k = 5$ and a similarity threshold of 0.6 in Fig 5.5p and the subject-based ground-truth. This is comparatively similar with the entity-based ground-truth for $k = 3$ and a similarity threshold of also 0.6 in Figure 5.4l which achieves a recall of 97% with a precision of more than 50%. However, the All-Ensemble model is still average in detecting outliers when combined with other categories as seen for the performance for *'min. category size'* $= 1$, for example a recall and precision of about 60% in Figure 5.5d, while the name-based model performs nearly the same in Figure 5.5b. The dataset-name model also performs similarly in detecting outliers as seen from the result in Figure 5.5c. The DS-kNN and All-Ensemble models perform equally good with both ground-truth types under the same settings and with the same parameters, yet slightly better with the more specific entity-based ground-truth with small category sizes and outliers.

Similarly, combining the DS-Prox-and-Dataset-Name leads to a comparatively good result like the All-Ensemble model as could be seen in Figure 5.5k and Figure 5.5o where a recall of about 90% is achieved with a precision above 40%. Therefore, the combination of content-based meta-features with name-based meta-features lead to optimum results that can equally detect outliers and accurately assign a relevant category for bigger category sizes. To achieve higher recall rates, the DS-kNN algorithm generally works better when the DL contains bigger categories and by using bigger values for '*k*'. With smaller category sizes and more outliers we need to use a higher similarity threshold and vice versa.

4.4 Validation Experiment

In order to re-validate our results and to check the generalisability of the results, we test the highlighted top performing DS-kNN settings from Tables 5.2, 5.3, and 5.5 on the 118 ds sample in Table 5.1. As the sample only has entity-based granularity of annotations, we only test the top performant models for the entity-based ground truth. We also only consider the models for category size 8+ as the 118 ds sample only includes categories with size 3+ or 8+ (i.e., 5+ and 8+ category sizes have the same categories as seen in Table 5.1). We execute the same leave-one-out experimental setup as described previously, and we evaluate the recall, precision and F1-scores which are summarised in Table 5.6. We also test on the 118 ds sample some specific DS-kNN settings which met specific criteria in our experiments on the 203 ds sample, for example, at least 90% recall. Those are presented in Table 5.7.

Table 5.6: The evaluation of top performant DS-kNN settings for the minimum category sizes of 1+, 3+ and 8+ with the 118 ds validation sample.

Model Type	Ground Truth Type	Min. Category Size	k	Similarity Threshold	Recall	Precision	F1-score
Dataset Name	entity	1	7	0.6	63.1%	90.6%	74.4%
All-Ensemble	entity	1	7	0.9	67.2%	98.4%	79.8%
DS-Prox Logit-Boost	entity	1	7	0.9	42.2%	100%	59.4%
Attribute Name Avg.	entity	3	7	0.3	79.6%	70.3%	74.7%
Attribute Name Max.	entity	3	5	0	73.7%	68.1%	70.8%
All-Ensemble	entity	3	7	0.8	75.6%	73.4%	74.5%
All-Ensemble	entity	8	5	0.9	63.2%	95.2%	75.9%
Attribute Name Avg.	entity	8	7	0.7	70.0%	97.8%	81.6%
Dataset Name	entity	8	7	0.4	68.2%	76.9%	72.3%

Table 5.7: The evaluation of specific DS-kNN settings which met specific criteria with the 203 ds sample. We re-validate them with the 118 ds sample.

Criteria	Model Type	Ground Truth Type	Min. Category Size	k	Similarity Threshold	Recall	Precision	F1-score
90% recall	All-Ensemble	entity	8	3	0.6	87.2%	49.6%	63.2%
85% recall	All-Ensemble	entity	8	5	0.7	85.7%	51.0%	63.9%
85% recall	DS-Prox-and-Dataset-Name	entity	8	5	0.7	92.7%	56.1%	69.9%
85% recall	Name-based-all	entity	8	7	0	94.9%	60.2%	73.7%
70% recall 90% precision	Dataset Name	entity	1	7	0.6	63.1%	90.6%	74.4%
70% recall 90% precision	All-Ensemble	entity	1	7	0.9	67.2%	98.4%	79.8%

As could be seen in the results in Tables 5.6 and 5.7, we were able to achieve similar or better results with the 118 ds validation sample by applying the same top performing DS-kNN settings from Tables 5.2, 5.3 and 5.5 and some DS-kNN settings meeting specific criteria. This indicates the generalisability of our results and the capability of our algorithm in replicating similar results with other DL samples.

5 Related Work

Categorization of datasets from heterogeneous domains is an emerging research topic, and relevant previous research include the work by [85], where they utilise the attribute names to cluster the datasets into categories using a probabilistic model. Datasets are assigned to different categories using different probabilities. They tackle the multi-label classification of datasets and retrieval of datasets from relevant domains by querying systems. Our approach improves this by using a machine-learning based approximate proximity mining technique instead of the Jaccard similarity of exact values. We also use content-based metadata for categorizing and not only name-based metadata. This is important for DLs where datasets are not well maintained with meaningful attribute names.

Clustering could also be applied to other types of semi-structured datasets like ontologies [9] and XML documents [11, 77], etc. In [9], they propose an algorithm to cluster instances from different ontologies based on their structural properties in the ontology graphs. Their goal is to facilitate ontology matching rather than domains discovery. Similarly, in [11, 77] they cluster the semi-structured documents based on their structure similarity and linguistic matchers.

Clustering free-text without any structure is also possible. For example, [21] aim to cluster short text messages by computing TF-IDF word similarity between free-text documents. Classical text document classification can also use multi-class supervised learning based on the extracted keywords in the document [19]. Similarly, [55] categorizes free-text documents using a k-NN based algorithm by first extracting TF-IDF weighted labels and feeding them to the algorithm. Another specific application would be clustering streaming data where a sliding window algorithm could be used [61], where they also use k-NN when finding relevant clusters for a given data instance ingested in a stream of data points. k-NN was also used with image categorization based on extracted features describing the images [78].

6 Conclusion

We proposed DS-kNN, a categorization algorithm for classifying datasets into pre-defined topic-wise groups. Our algorithm can be applied in a DL environment to support users in automatically finding relevant datasets for analysis. Our algorithm uses extracted metadata from datasets to compute their similarities to other datasets in the DL using a proximity mining model and name strings comparisons. Those similarity scores are fed to DS-kNN to decide on the most relevant category for a dataset based on its top-k nearest neighbours. Our algorithm was effective in categorizing the datasets in a real-world DL and detecting outliers. In the future, we will seek to improve our algorithm with semantic analysis of values found in the attributes to complement the syntactical comparisons we compute in the proximity models.

Acknowledgements. This research has been partially funded by the European Commission through the Erasmus Mundus Joint Doctorate (IT4BI-DC).

Chapter 6

Prox-mine tool for browsing DLs using proximity mining

A picture is worth a thousand words and is better marketing for an idea than just plain words.

Abstract

We present Prox-mine, a tool for browsing DLs using proximity mining. Prox-mine integrates all the different components of this thesis and presents it in a coherent prototype. It supports the collection of descriptive statistics about tabular datasets and their attributes, calculation of overall dataset similarity scores between dataset pairs using the collected statistics and the proximity models in Chapter 4, categorization of datasets into pre-existing categories defined in the DL using the computed similarities between datasets and a k-Nearest-Neighbour algorithm described in Chapter 5, in addition to construction of proximity graph visualisations which summarise the overall structure of the DL by showing datasets and their categories as nodes and relationships (similarities) modelled as edges.

The tool can be used to support the DL users in extracting useful metadata involving descriptive statistics about the content of the datasets stored, can help data wranglers and curators in finding relevant datasets by querying similar datasets in the data lake given an input query dataset, and can also provide schema matching support by showing the overlap of similar attributes and their data / names between dataset pairs (using name-based and content-based analysis techniques as described in Chapter 4).

1 Introduction

We present a prototype integrating the different components of this thesis. We call the tool Prox-Mine, where the goal is to support data wranglers and curators in finding relevant datasets by querying similar datasets in the DL given an input query dataset, and can also provide schema matching support by showing the overlap of similar attributes and their data / names between dataset pairs (using name-based and content-based analysis techniques as described in Chapter 4).

Prox-mine supports the collection of descriptive statistics about tabular datasets and their attributes, calculation of overall dataset similarity scores between dataset pairs using the collected statistics and the proximity models in Chapter 4, categorization of datasets into pre-existing categories defined in the DL using the computed similarities between datasets and a k-Nearest-Neighbour algorithm described in Chapter 5, in addition to construction of proximity graph visualisations which summarise the overall structure of the DL by showing datasets and their categories as nodes and relationships (similarities) modelled as edges.

Prox-mine is organised into the following components:

- **Data Lake Index**: an index of the datasets stored in the DL and their ground-truth categories.
- **Similarity Search**: supports searching for similar datasets to a query dataset based on the different proximity models and a minimum similarity threshold.
- **Dataset Categorization**: applies the DS-kNN algorithm from Chapter 5 to facilitate the automatic recommendation of categories for a dataset based on the most proposed category by the most similar datasets in the DL.
- **Dataset Matching**: applies the proximity models over a pair of datasets in order to compute their overall similarity and the most similar attribute pairs between them.
- **Proximity Graph**: constructs a graph visualisation to show the relationships between datasets based on their computed similarities using the different proximity models and different minimum thresholds.

We describe each of the components above in the rest of this chapter. The tool is hosted on a web server[1] and is implemented as a Java-based web

[1]A live version of Prox-mine can be accessed at the following link: `http://dtim.essi.upc.edu:8080/proxmine` using the credentials of username:"proxmine" and password "12345". The source code for the tool is provided at: `https://github.com/AymanUPC/proxmine`, and more details can be found at the tool's main page: `https://www.essi.upc.edu/dtim/tools/proxmine`

application. The tool implements a demonstration of the above components over the OpenML *OML02* sample used in our experiments in Chapter 4. All proximity models used in the demonstration are based on the same cross-validation experimental setup described in Chapter 4, where we make sure that the datasets we apply the models on were not used in the training steps. In addition to this overview, the tool provides tooltip help and support too by hovering over the input and output fields and titles when using it.

2 Data Lake Index

In this component, an overview of all the datasets stored in the DL is presented. This includes the following for each dataset:

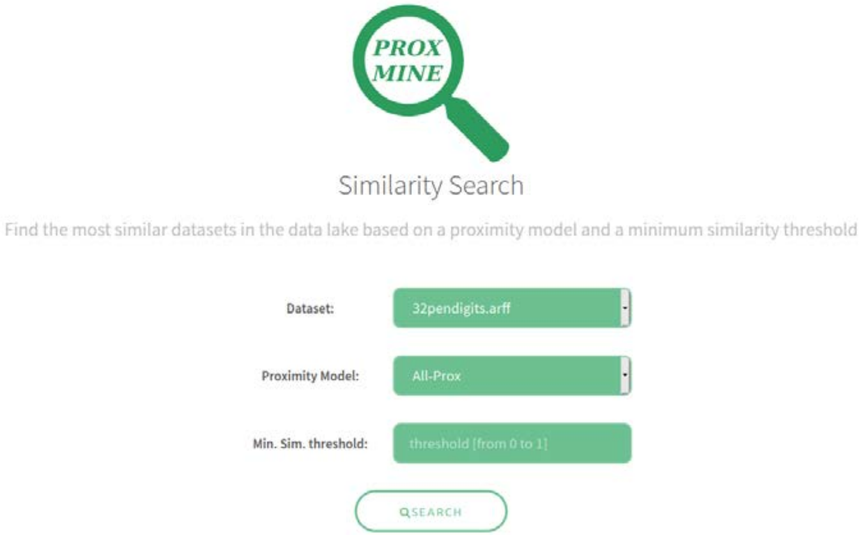
- Dataset: the filename of the dataset
- Dataset Name: the name of the dataset
- Dataset Description: a link to the webpage with the textual description and statistical metadata describing the dataset and its attributes.
- No. of Attributes: the number of attributes in the dataset
- No. of Instances: the number of instances in the dataset
- Ground-truth Category: the annotated ground-truth category assigned to the dataset based on manual inspection of it and its description (see Chapter 5.1 for more details about this annotation).

We also provide a list of all the ground-truth categories and the number of datasets which belong to those categories.

3 Similarity Search

This component allows searching for similar datasets to a query dataset (as shown in Figure 6.1) based on the different proximity models and a minimum similarity threshold. The proximity models are those described in Chapter 4 and the minimum threshold should be a real-number in the range $[0, 1]$, for example 0.75.

The output can be seen in Figure 6.2. This includes a description of the input dataset used in the query. Then a table with the output similar datasets is given, ordered by the similarity score computed by the proximity model, in descending order of similarity.

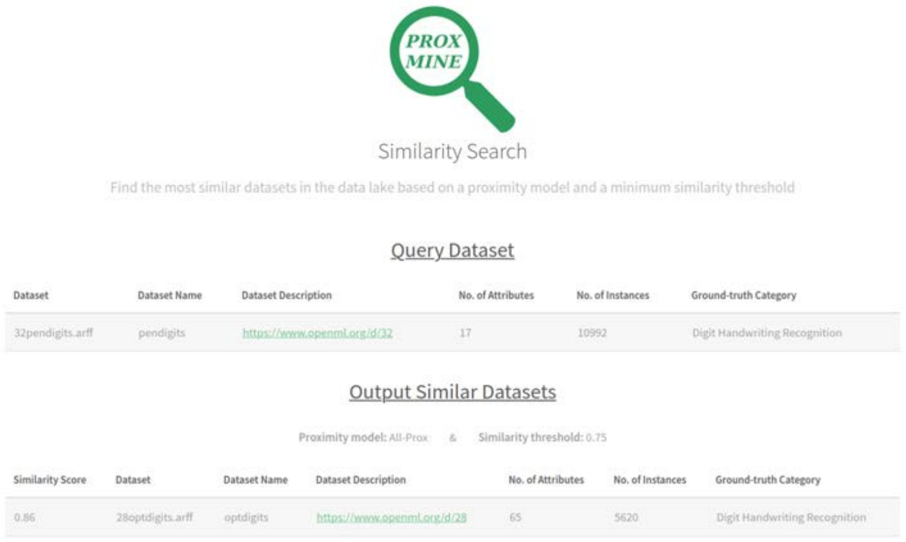

PROX
MINE
Similarity Search
Find the most similar datasets in the data lake based on a proximity model and a minimum similarity threshold
SEARCH

Dataset: 32pendigits.arff

Proximity Model: All-Prox

Min. Sim. threshold: threshold [from 0 to 1]

Figure 6.1: The input screen for the similarity search component of Prox-mine


PROX
MINE
Similarity Search
Find the most similar datasets in the data lake based on a proximity model and a minimum similarity threshold

Query Dataset

Dataset	Dataset Name	Dataset Description	No. of Attributes	No. of Instances	Ground-truth Category
32pendigits.arff	pendigits	https://www.openml.org/d/32	17	10992	Digit Handwriting Recognition

Output Similar Datasets

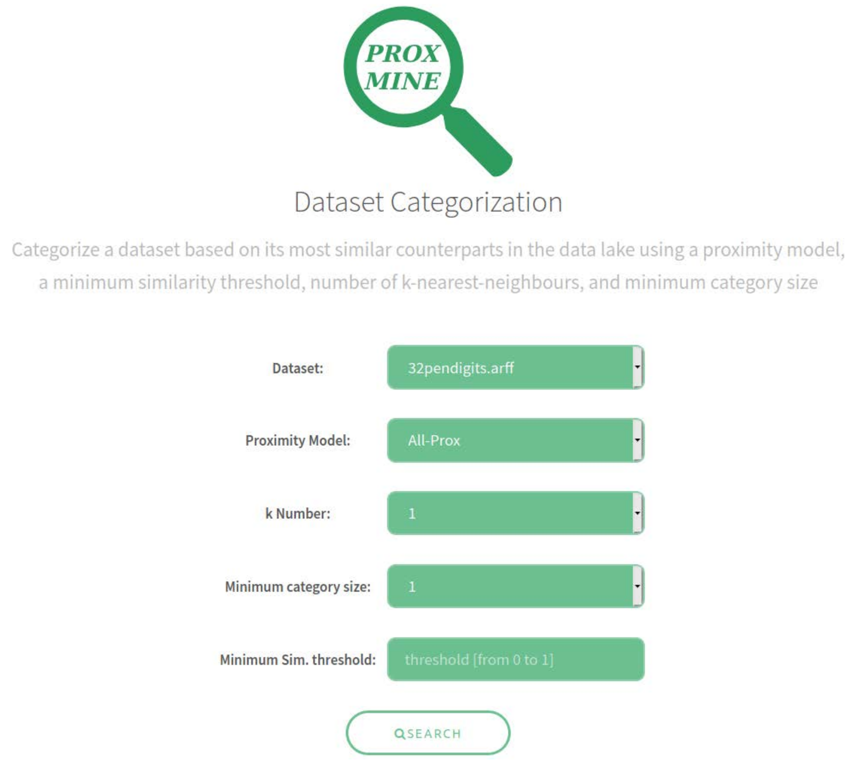
Proximity model: All-Prox & Similarity threshold: 0.75

Similarity Score	Dataset	Dataset Name	Dataset Description	No. of Attributes	No. of Instances	Ground-truth Category
0.86	28optdigits.arff	optdigits	https://www.openml.org/d/28	65	5620	Digit Handwriting Recognition

Figure 6.2: The output screen for the similarity search component of Prox-mine

4 Dataset Categorization

This component allows recommending a category for a query dataset based on the most similar datasets found in the DL. This implements the DS-kNN algorithm described in Chapter 5. The input required, as seen in Figure 6.3,


Dataset Categorization

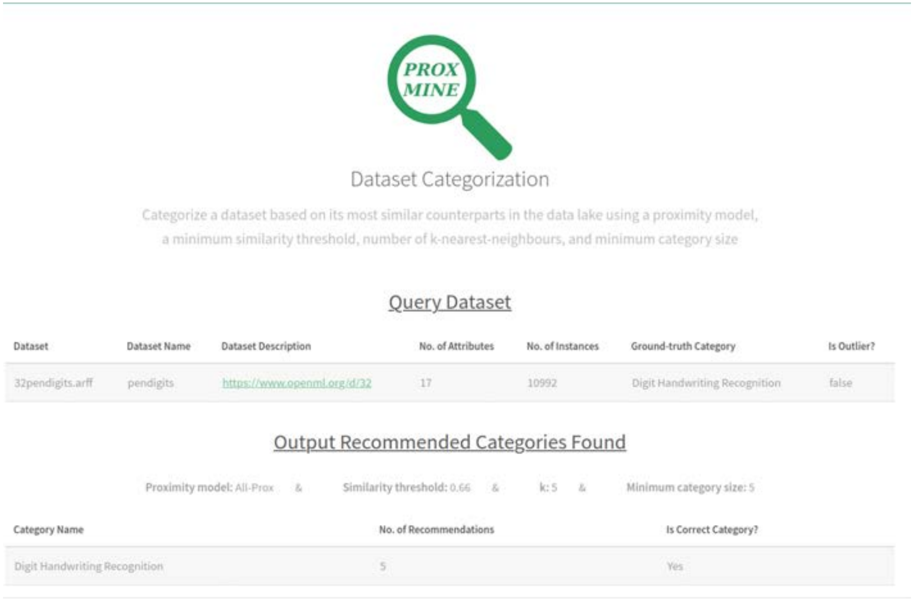
Categorize a dataset based on its most similar counterparts in the data lake using a proximity model, a minimum similarity threshold, number of k-nearest-neighbours, and minimum category size

Dataset:	32pendigits.arff
Proximity Model:	All-Prox
k Number:	1
Minimum category size:	1
Minimum Sim. threshold:	threshold [from 0 to 1]

Figure 6.3: The input screen for the dataset categorization component of Prox-mine

includes the query dataset, the proximity model, the number of k-nearest-neighbours to use for the recommendation (i.e., how many datasets from the top similar ones should recommend their category for the dataset), the minimum category size as described in Chapter 4 (i.e., only categories having this number of datasets or above can be recommended), and the minimum similarity threshold as a real-number in the range $[0, 1]$.

The output can be seen in Figure 6.4. This includes first a description of the input dataset used in the query. This table also includes an extra column called *'Is Outlier?'* which indicates if the dataset belongs to a category in the ground-truth of size 1, i.e., a category which only includes the query dataset and no other dataset stored in the DL. Then a table with the output recommended categories found using the DS-kNN algorithm is given ordered by the number of recommendations in descending order. Each of the most similar k nearest-neighbours recommends its category for the dataset, and the aggregate count is given for each category here. An indication whether the recommended category is the correct one is also given in the column *'Is Correct Category?'*. Finally, a table is given showing a drill-down of the k-nearest-neighbours found which led to the recommended categories. Here



PROX MINE

Dataset Categorization

Categorize a dataset based on its most similar counterparts in the data lake using a proximity model, a minimum similarity threshold, number of k-nearest-neighbours, and minimum category size

Query Dataset

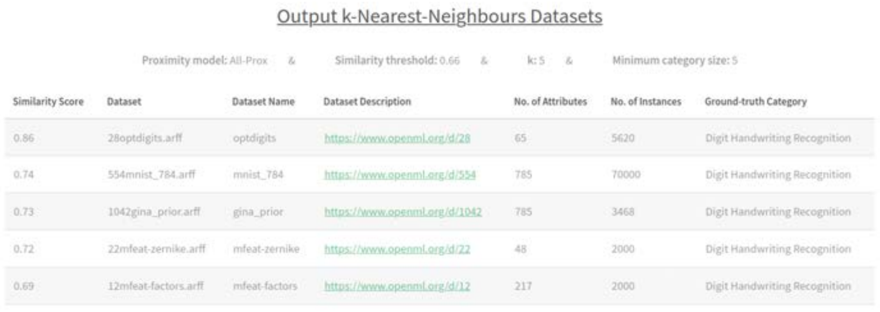
Dataset	Dataset Name	Dataset Description	No. of Attributes	No. of Instances	Ground-truth Category	Is Outlier?
32pendigits.arff	pendigits	https://www.openml.org/d/32	17	10992	Digit Handwriting Recognition	false

Output Recommended Categories Found

Proximity model: All-Prox & Similarity threshold: 0.66 & k: 5 & Minimum category size: 5

Category Name	No. of Recommendations	Is Correct Category?
Digit Handwriting Recognition	5	Yes

(a)



Output k-Nearest-Neighbours Datasets

Proximity model: All-Prox & Similarity threshold: 0.66 & k: 5 & Minimum category size: 5

Similarity Score	Dataset	Dataset Name	Dataset Description	No. of Attributes	No. of Instances	Ground-truth Category
0.86	28optdigits.arff	optdigits	https://www.openml.org/d/28	65	5620	Digit Handwriting Recognition
0.74	554mnist_784.arff	mnist_784	https://www.openml.org/d/554	785	70000	Digit Handwriting Recognition
0.73	1042gina_prior.arff	gina_prior	https://www.openml.org/d/1042	785	3468	Digit Handwriting Recognition
0.72	22mfeat-zernike.arff	mfeat-zernike	https://www.openml.org/d/22	48	2000	Digit Handwriting Recognition
0.69	12mfeat-factors.arff	mfeat-factors	https://www.openml.org/d/12	217	2000	Digit Handwriting Recognition

(b)

Figure 6.4: The output screens for the dataset categorization component of Prox-mine.

each of the top-k datasets is described in descending order of similarity scores.

5 Dataset Matching

This component allows the application of the proximity models over a pair of datasets in order to compute their overall similarity and the most similar attribute pairs between them using the models trained for numeric and nominal attributes over the OML01 sample described in Chapter 5.1. This is done using the input screen in Figure 6.5 where 2 datasets can be selected along with a minimum similarity threshold in the range $[0.5, 1]$. We restrict

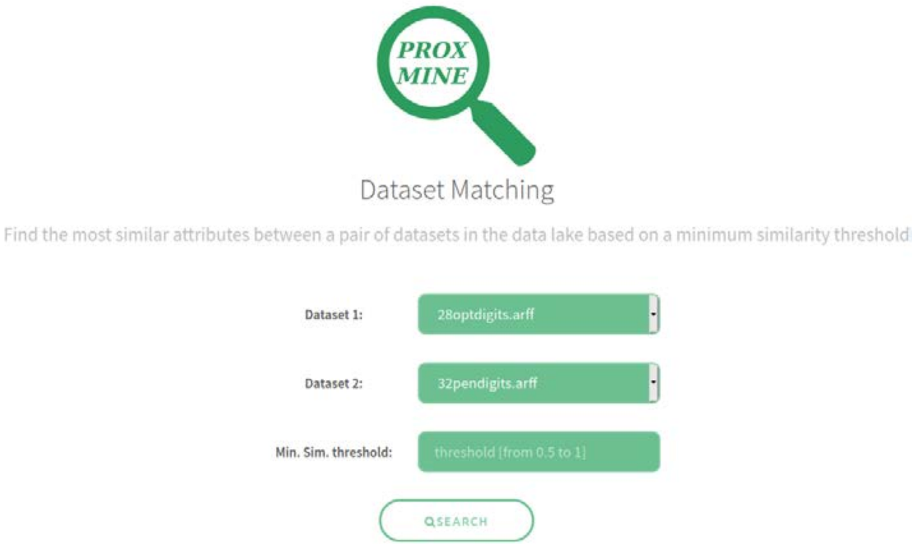

PROX MINE
Dataset Matching
Find the most similar attributes between a pair of datasets in the data lake based on a minimum similarity threshold
Dataset 1:
28optdigits.arff
Dataset 2:
32pendigits.arff
Min. Sim. threshold:
threshold (from 0.5 to 1)
SEARCH

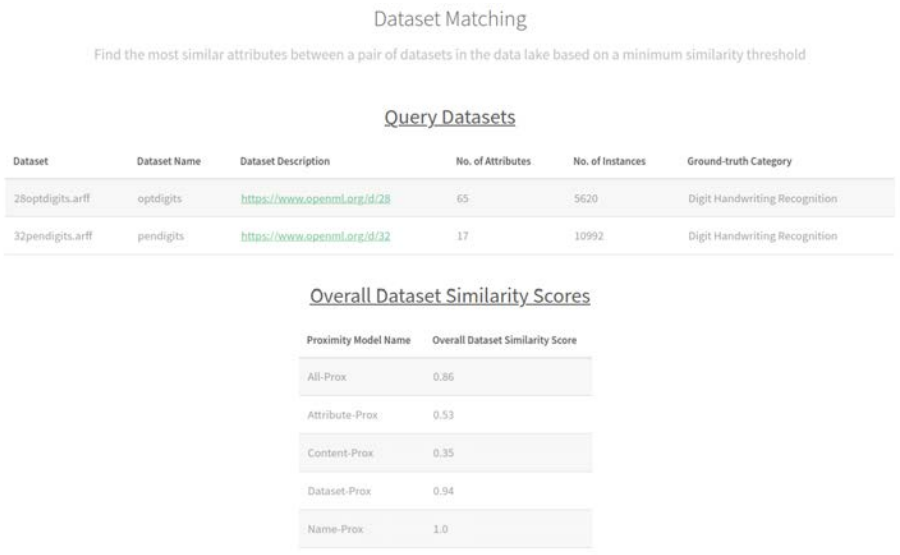
Figure 6.5: The input screen for the dataset matching component of Prox-mine

the input to this threshold range as attribute matches with a similarity score below this threshold could be considered as irrelevant and not strong enough to be significant, distracting the user when analysing the output by presenting too many attribute pair matches.

The output can be seen in Figure 6.6. This includes first a description of both input datasets and their ground-truth categories. This is followed by a table giving all the overall dataset similarity scores based on all types of proximity models described in Chapter 4. Then the matching attributes are presented in two separate tables, where the first table is based on a similarity score computed using the content-based similarity models and where the second table is based on similarity computed using Levenshtein distances of the attribute names. Each table shows attribute pair matches sorted by descending order of the similarity scores computed. We indicate the attribute types compared, and a description of each attribute matched in the pair, one attribute from each dataset. For the *'Values'* column, we give the unique distinct values found for nominal attributes separated by pipes, and for numeric attributes we give the range of values (minimum - maximum) and the mean value.

5.1 New Dataset Matching

Another component for matching datasets include the new dataset matching component, where the input screen can be seen in Figure 6.7. The user is able to select a new CSV dataset file from the filesystem (stored on their computer) as the input of the matching process. The dataset is matched against another



Dataset Matching

Find the most similar attributes between a pair of datasets in the data lake based on a minimum similarity threshold

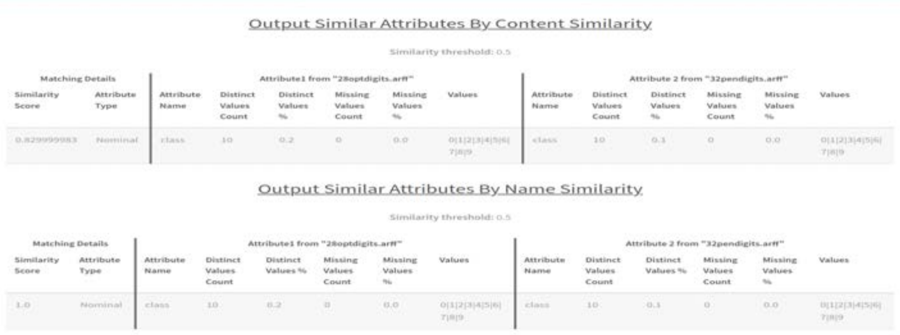
Query Datasets

Dataset	Dataset Name	Dataset Description	No. of Attributes	No. of Instances	Ground-truth Category
28optdigits.arff	optdigits	https://www.openml.org/d/28	65	5620	Digit Handwriting Recognition
32pendigits.arff	pendigits	https://www.openml.org/d/32	17	10992	Digit Handwriting Recognition

Overall Dataset Similarity Scores

Proximity Model Name	Overall Dataset Similarity Score
All-Prox	0.86
Attribute-Prox	0.53
Content-Prox	0.35
Dataset-Prox	0.94
Name-Prox	1.0

(a)



Output Similar Attributes By Content Similarity

Similarity threshold: 0.5

Matching Details		Attribute1 from "28optdigits.arff"						Attribute 2 from "32pendigits.arff"					
Similarity Score	Attribute Type	Attribute Name	Distinct Values Count	Distinct Values %	Missing Values Count	Missing Values %	Values	Attribute Name	Distinct Values Count	Distinct Values %	Missing Values Count	Missing Values %	Values
0.829999983	Nominal	class	10	0.2	0	0.0	0|1|2|3|4|5|6|7|8|9	class	10	0.1	0	0.0	0|1|2|3|4|5|6|7|8|9

Output Similar Attributes By Name Similarity

Similarity threshold: 0.5

Matching Details		Attribute1 from "28optdigits.arff"						Attribute 2 from "32pendigits.arff"					
Similarity Score	Attribute Type	Attribute Name	Distinct Values Count	Distinct Values %	Missing Values Count	Missing Values %	Values	Attribute Name	Distinct Values Count	Distinct Values %	Missing Values Count	Missing Values %	Values
1.0	Nominal	class	10	0.2	0	0.0	0|1|2|3|4|5|6|7|8|9	class	10	0.1	0	0.0	0|1|2|3|4|5|6|7|8|9

(b)

Figure 6.6: The output screens for the dataset matching component of Prox-mine.

dataset available in the DL sample. A minimum similarity threshold in the range $[0, 1]$ should also be selected.

The output can be seen in Figure 6.8. It also includes a description of the input datasets, the output content-based similarity scores of attribute matching and name-based similarity scores as well using the Levenshtein distances of the attribute names. The overall similarity score for the dataset pair is given above the matching output tables for content-based and name-based matching. Here, the content-based and name-based proximity is aggregated using the attribute-level models and algorithms in Chapter 4 to give a single approximate similarity score for the dataset pair.

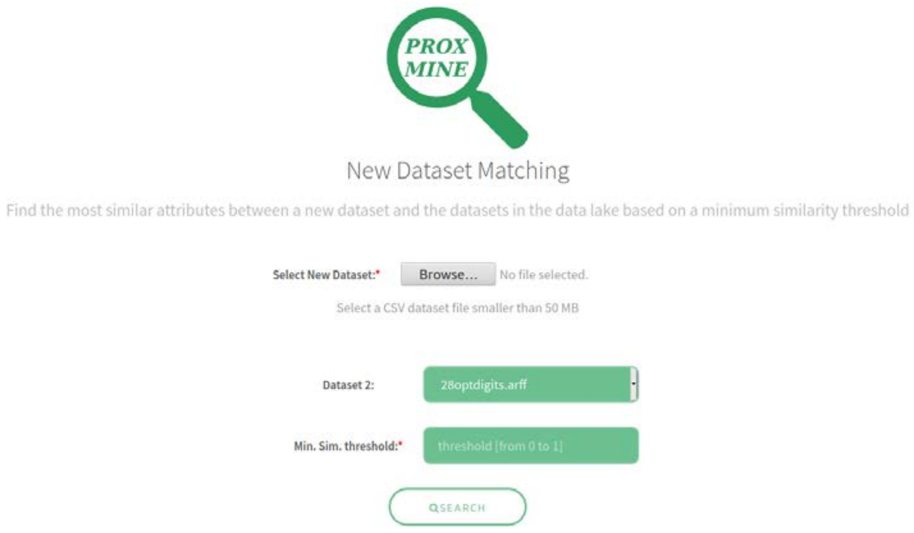

PROX MINE
New Dataset Matching
Find the most similar attributes between a new dataset and the datasets in the data lake based on a minimum similarity threshold
Select New Dataset:*
Browse...
No file selected.
Select a CSV dataset file smaller than 50 MB
Dataset 2:
28optdigits.arff
Min. Sim. threshold:*
threshold (from 0 to 1)
SEARCH

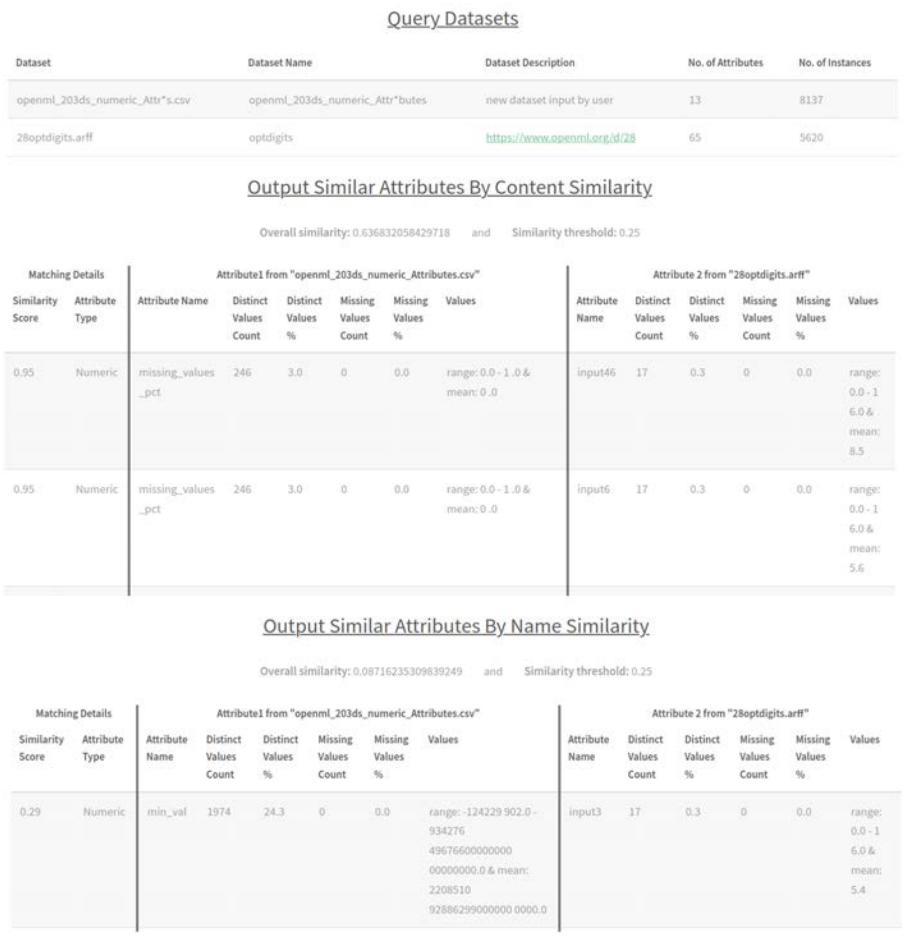
Figure 6.7: The input screen for the new dataset matching component of Prox-mine

6 Proximity Graph

In this component, we construct a graph visualisation to visualise the relationships between datasets based on their computed similarities using the different proximity models and different minimum thresholds. We call this a *proximity graph*, which was introduced in Chapter 5. The graph shows individual datasets as nodes and edges showing relationships between dataset pairs (where an edge is drawn between dataset pairs having a similarity scores above the minimum threshold). The purpose of this is to be able to see all the datasets in the DL and an overview of the relationships between them in a visually intuitive and interactive manner.

For the input screen seen in Figure 6.9, the user should select which proximity model to use when constructing the relationships (similarities) between datasets and the minimum similarity threshold in $\{0.5, 0.75\}$. This way we allow to build proximity graphs with at least *average* similarity strength or other graphs only showing *strong* relationships (in the case of selecting 0.75). The shape of the graph will differ based on the filtered relationships satisfying the minimum similarity threshold, as we use a force-directed graph structuring algorithm [66] which depends on the relationship strengths between the nodes (the datasets) to group them and to position them accordingly. More interesting proximity graphs can be achieved with higher minimum similarity thresholds, as this way only very similar datasets (usually having similar categories in the ground-truth and the same colour for the nodes) are grouped together in a more coherent manner.

The overview of the output proximity graph can be seen in Figure 6.10.



Query Datasets

Dataset	Dataset Name	Dataset Description	No. of Attributes	No. of Instances
openml_203ds_numeric_Attr*s.csv	openml_203ds_numeric_Attr*butes	new dataset input by user	13	8137
28optdigits.arff	optdigits	https://www.openml.org/d/28	65	5620

Output Similar Attributes By Content Similarity

Overall similarity: 0.636832058429718 and Similarity threshold: 0.25

Matching Details		Attribute1 from "openml_203ds_numeric_Attributes.csv"						Attribute 2 from "28optdigits.arff"					
Similarity Score	Attribute Type	Attribute Name	Distinct Values Count	Distinct Values %	Missing Values Count	Missing Values %	Values	Attribute Name	Distinct Values Count	Distinct Values %	Missing Values Count	Missing Values %	Values
0.95	Numeric	missing_values _pct	246	3.0	0	0.0	range: 0.0 - 1 .0 & mean: 0 .0	input46	17	0.3	0	0.0	range: 0.0 - 1 6.0 & mean: 8.5
0.95	Numeric	missing_values _pct	246	3.0	0	0.0	range: 0.0 - 1 .0 & mean: 0 .0	input6	17	0.3	0	0.0	range: 0.0 - 1 6.0 & mean: 5.6

Output Similar Attributes By Name Similarity

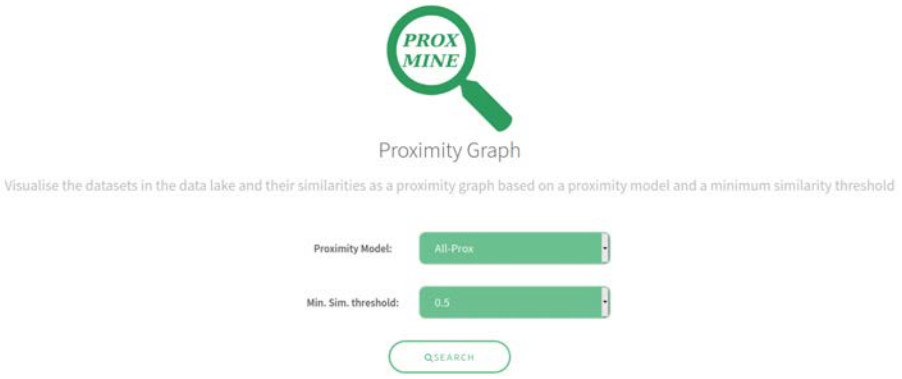
Overall similarity: 0.08716235309839249 and Similarity threshold: 0.25

Matching Details		Attribute1 from "openml_203ds_numeric_Attributes.csv"						Attribute 2 from "28optdigits.arff"					
Similarity Score	Attribute Type	Attribute Name	Distinct Values Count	Distinct Values %	Missing Values Count	Missing Values %	Values	Attribute Name	Distinct Values Count	Distinct Values %	Missing Values Count	Missing Values %	Values
0.29	Numeric	min_val	1974	24.3	0	0.0	range: -124229 902.0 - 934276 49676600000000 00000000.0 & mean: 2208510 92886299000000 0000.0	input3	17	0.3	0	0.0	range: 0.0 - 1 6.0 & mean: 5.4

Figure 6.8: The output screen for the new dataset matching component of Prox-mine

This is an interactive graph visualisation, where the graph is shown in the middle. On the left-side, there is a graph search and category-filtration selector, and at the bottom there are three buttons to zoom-in, zoom-out or to return to default view. When the graph is zoomed-in, the nodes show the names of the datasets as their labels. The colours of the nodes represent the ground-truth category they belong to, therefore datasets in the same category have the same colour.

The user can interact with the graph by zooming in and out and toggling the view with the cursor to move the view to a different part of the graph. In addition, the user can search for a specific dataset node using the search bar in the left panel, as seen in Figure 6.11. The datasets can also be filtered to only highlight those belonging to a specific ground-truth category by selecting the category from the drop-down group selector menu. The result of filtration



Proximity Graph

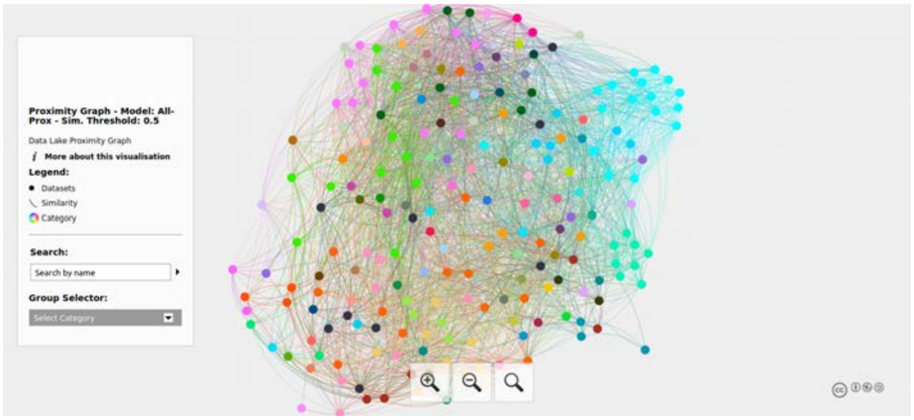
Visualise the datasets in the data lake and their similarities as a proximity graph based on a proximity model and a minimum similarity threshold

Proximity Model: All-Prox

Min. Sim. threshold: 0.5

SEARCH

Figure 6.9: The input screen for the proximity graph component of Prox-mine


Proximity Graph - Model: All-Prox - Sim. Threshold: 0.5

Data Lake Proximity Graph

More about this visualisation

Legend:

● Datasets

Similarity

Category

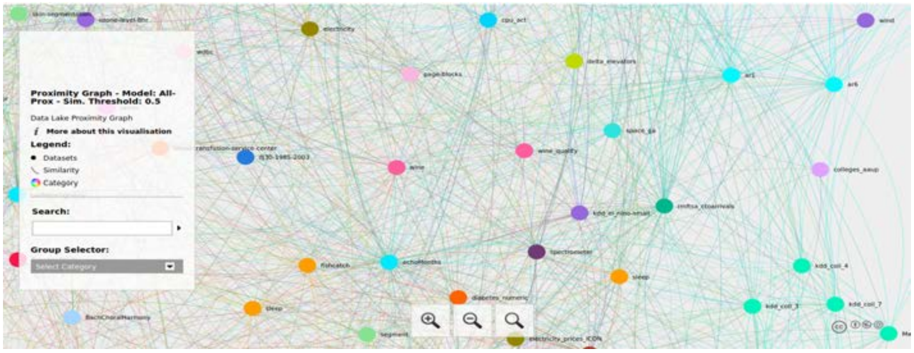
Search:

Search by name

Group Selector:

Select Category

(a)



Proximity Graph - Model: All-Prox - Sim. Threshold: 0.5

Data Lake Proximity Graph

More about this visualisation

Legend:

● Datasets

Similarity

Category

Search:

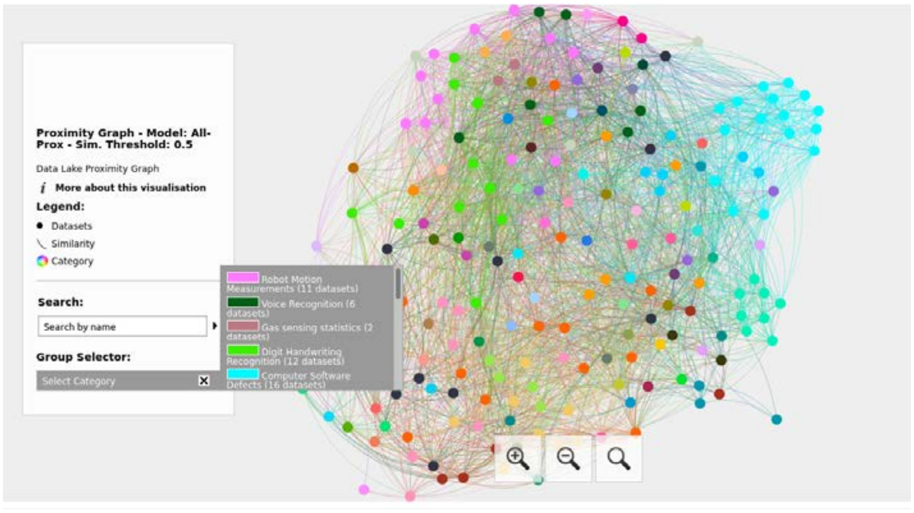
Group Selector:

Select Category

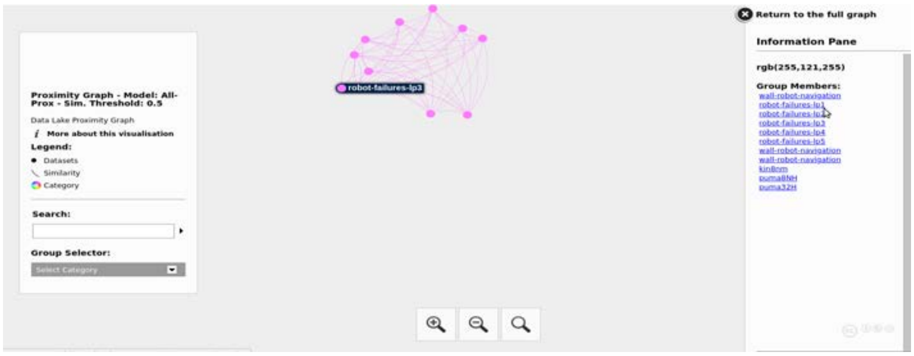
(b)

Figure 6.10: An overview of the output proximity graph component of Prox-mine, where (a) gives a zoomed-out view and (b) gives a zoomed-in view

by category is given in Figure 6.11b, where a only the datasets belonging to


Proximity Graph - Model: All-Prox - Sim. Threshold: 0.5
Data Lake Proximity Graph
More about this visualisation
Legend:
Datasets
Similarity
Category
Search:
Search by name
Group Selector:
Select Category
Robot Motion Measurements (11 datasets)
Voice Recognition (6 datasets)
Gas sensing statistics (2 datasets)
Digit Handwriting Recognition (12 datasets)
Computer Software Defects (16 datasets)

(a)

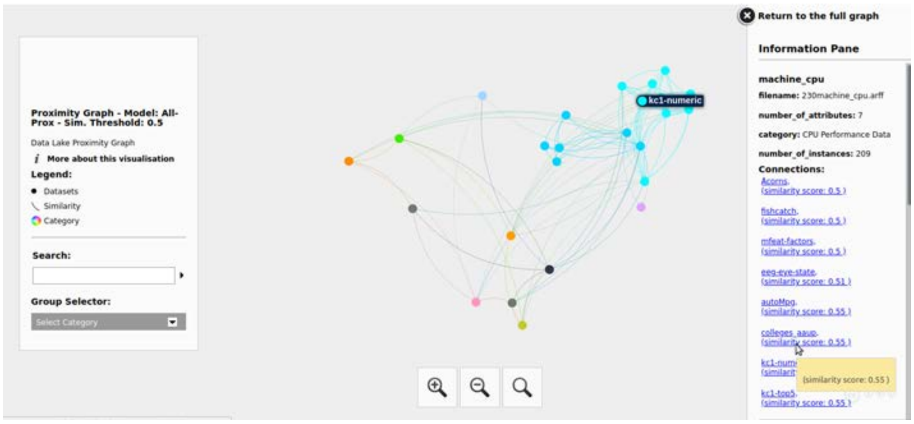


(b)

Figure 6.11: The search and filtration panel of the output proximity graph component of Prox-mine, where (a) gives a view of the category selector in the left-panel and (b) gives the result of applying the filtration step

the specific category are shown and a list showing those datasets is given in the information pane on the right.

When a specific node is clicked and selected, only the datasets having a relationship with a similarity score above the minimum similarity threshold are shown, as could be seen in Figure 6.12. The description of the selected dataset and the connections (edges) list is given in the information pane on the right side. This includes a list of all connected (related) datasets and the similarity scores for each related dataset. Hovering over a specific dataset in the connections information pane highlights the dataset in the main proximity graph view and shows its name label. To exit back to the full proximity graph, the user should click the 'X' symbol at the top of the information pane on the right side.


Proximity Graph - Model: All-Prox - Sim. Threshold: 0.5

Data Lake Proximity Graph

More about this visualisation

Legend:

● Datasets

Similarity

Category

Search:

Group Selector:

Select Category

Return to the full graph

Information Pane

machine_cpu

filename: 230machine_cpu.arff

number_of_attributes: 7

category: CPU Performance Data

number_of_instances: 209

Connections:

Acorns, (similarity score: 0.5)

fishcatch, (similarity score: 0.5)

mfeat-factors, (similarity score: 0.5)

eeg-eye-state, (similarity score: 0.51)

autoMpg, (similarity score: 0.55)

colleges_aaup, (similarity score: 0.55)

kc1-num (similarit (similarity score: 0.55)

kc1-top5, (similarity score: 0.55)

Figure 6.12: The selection of a specific dataset node in the proximity graph and the relationships information panel shown on the right side

Chapter 7

Conclusions and Future Directions

Positivity leads to positive energy and success, negativity on the other hand leads to negative energy and dismay.

Abstract

In this chapter, we summarise the outcome and results of this thesis, presented in Chapters 2 – 6. *Additionally, we present several interesting future directions arising from this thesis work.*

1 Conclusions

Everything must eventually come to an end. [...] Ctrl+S [...] Alt+F4 [...] Ctrl+Alt+Delete.

We have presented a novel metadata-based framework for computing dataset proximity using data profiling and supervised machine learning techniques. The output from this framework is a metadata repository which can support in DL governance efforts, whereby datasets are automatically profiled, matched against other datasets in the DL and categorised into topical domains in order to support information discovery and analytical tasks over the DL.

We proposed novel approaches for computing *approximate* dataset similarity where dataset pairs contain overlapping attribute pairs based on *content statistics* summarising the information stored in the attributes (instead of exact values matching where attribute names should be identical and their values should be coded using the same string patterns). This makes it possible to provide estimation of overall dataset similarity to support early-pruning tasks of holistic schema matching and automatic dataset categorization based on most similar datasets already categorized. The developed approach and its techniques are novel solutions for those research challenges.

To implement this approach, we presented in Chapter 2 an overview of exact value-based matching techniques for the dataset similarity computation challenge. We were able to demonstrate the feasibility of this approach using a real-world DL sample and we summarised the shortcomings of such value-based techniques; mainly: 1. high computational costs for the comparisons and 2. inability to detect all related attributes due to the differences in the way values are coded or stored.

To tackle the shortcoming from the value-based approaches in Chapter 2, we developed a novel technique called *proximity mining* in Chapter 3 for computing approximate dataset similarity based on schema and content metadata we extract from the datasets. We utilise supervised machine learning techniques to learn models which can detect similar and duplicate schemata stored in the DL. The effectiveness and efficiency of the approach was evaluated on an expanded real-world DL sample, where we were able to achieve high recall and efficiency gain.

We improve the proximity mining techniques using finer-grained attribute-level metadata in Chapter 4. Here, we collect finer grained metadata summarising the content of each attribute in the datasets, we propose a greedy algorithm for finding top-matching attribute pairs between different datasets, and we utilise supervised machine learning models to compute overall dataset similarity between dataset pairs in the DL. The approach uses different metadata, mainly schema metadata consisting of attribute names and types in addition to content metadata consisting of data profiles for the different attributes. We test our proposed approach against both techniques from Chapter 2 and Chapter 3 and we discover that the improved technique can achieve better performance in terms of effectiveness and efficiency gains. This was proven to be useful for the holistic schema matching early-pruning task, whereby dataset pairs having an overall similarity score exceeding a minimum threshold are proposed for further, more detailed schema matching, and those which are below the similarity threshold are pre-filtered from any further scrutinization.

In Chapter 5, we use the output dataset similarity cross-dataset relationships to automatically categorise newly ingested datasets in the DL to topics of interest already existing or to detect that the ingested dataset is an outlier which has no similar datasets stored in the DL. This is done using a k-nearest-neighbour algorithm which uses the computed proximity between datasets in finding the top-k most similar datasets in the DL and to propose their topics as candidates for the newly ingested dataset.

Finally, in Chapter 6, we demonstrate a prototype which uses the output from Chapters 3 and 4 in finding related dataset pairs, their overlapping attributes and to automatically categorize datasets based on those most similar datasets. We develop a simple web application which is able to store annotations of datasets stored in the DL using a metadata repository (a DB) and to visualise the output from our approach using a querying interface to support data wranglers and DL uses in information discovery and DL exploration.

This thesis is a first step towards fully automated DL governance for supporting information discovery and DL segmentation. We were able to demonstrate techniques which are able to effectively and efficiently compute overall dataset similarity in an approximate manner. The proposed techniques were found to be more effective than state-of-the-art value-based schema matching techniques and human-based manual analysis.

2 Future Directions

What is partially left should never be completely abandoned.

This thesis opens multiple research directions for future work. Firstly, the same techniques can be adopted for other dataset types like semi-structured and unstructured datasets. This can support the different variety of datasets stored in the DL. In addition, the techniques proposed can be augmented with semantic matching where datasets, their attributes and their values can be mapped to a knowledge base to support the computation of their similarity. This can be useful for datasets storing different content but which are modelling the same topics of interest.

The framework can also incorporate human-in-the-loop techniques in a pay-as-you-go approach whereby the data wrangler can interfere in the matching process to improve the supervised learning models and to help the models when they are not accurately detecting similar attribute and dataset pairs. This can be done by manually correcting false positives and false negatives, and incorporating this in the training process of the models. This can be seen as an incremental learning process.

Finally, the output of this thesis can be used for other applications like dataset search ranking engines, data integration tasks where *joinable* datasets are detected, and de-duplication tasks where highly similar schemata are integrated into a single dataset.

Standing on the shoulders of giants, a person's legacy continues via the knowledge and work passed on to their students and to others.

References

[1] Z. Abedjan, L. Golab, and F. Naumann. Profiling relational data: a survey. *The VLDB Journal*, 24(4):557–581, 2015.

[2] Z. Abedjan, T. Gruetze, A. Jentzsch, and F. Naumann. Profiling and mining RDF data with ProLOD++. In *Proceedings - International Conference on Data Engineering*, pages 1198–1201, 2014.

[3] Z. Abedjan and F. Naumann. Improving RDF Data Through Association Rule Mining. *Datenbank-Spektrum*, 13(2):111, 2013.

[4] A. Abelló. Big Data Design. In *Proceedings of ACM DOLAP*, pages 35–38, 2015.

[5] A. Abelló, J. Darmont, L. Etcheverry, M. Golfarelli, J. N. Mazón, F. Naumann, T. B. Pedersen, S. Rizzi, J. Trujillo, P. Vassiliadis, and G. Vossen. Fusion cubes: Towards self-service business intelligence. *International Journal of Data Warehousing and Mining*, 9(2):66–88, 2013.

[6] G. Adomavicius, R. Sankaranarayanan, S. Sen, and A. Tuzhilin. Incorporating contextual information in recommender systems using a multidimensional approach. *ACM Transactions on Information Systems (TOIS)*, 23(1):103–145, 2005.

[7] J. Akoka, L. Berti-Équille, O. Boucelma, M. Bouzeghoub, I. Comyn-Wattiau, M. Cosquer, V. Goasdoué-Thion, Z. Kedad, S. Nugier, V. Peralta, and S. S. Cherfi. A framework for quality evaluation in data integration systems. In *9th International Conference on Entreprise Information Systems (ICEIS)*, pages 170–175, 2007.

[8] A. Alexandrov, R. Bergmann, S. Ewen, J.-C. Freytag, F. Hueske, A. Heise, O. Kao, M. Leich, U. Leser, V. Markl, F. Naumann, M. Peters, A. Rheinländer, M. J. Sax, S. Schelter, M. Höger, K. Tzoumas, and D. Warneke. The Stratosphere platform for big data analytics. *The VLDB Journal*, 23(6):939–964, 2014.

[9] A. Algergawy, S. Massmann, and E. Rahm. A Clustering-Based Approach for Large-Scale Ontology Matching. In *East European Conference on Advances in Databases and Information Systems (ADBIS)*, pages 415–428. Springer, 2011.

[10] A. Algergawy, M. Mesiti, R. Nayak, and G. Saake. XML data clustering: An overview. *ACM Computing Surveys (CSUR)*, 43(4):25, 2011.

[11] A. Algergawy, E. Schallehn, and G. Saake. A schema matching-based approach to XML schema clustering. In *Proceedings of the International Conference on Information Integration and Web-based Applications & Services*, pages 131–136. ACM, 2008.

[12] I. Alhassan, D. Sammon, and M. Daly. Data governance activities: an analysis of the literature. *Journal of Decision Systems*, 25:64–75, 2016.

[13] H. Alrehamy and C. Walker. Personal Data Lake With Data Gravity Pull. In *IEEE Fifth International Conference on Big Data and Cloud Computing (BDCloud)*, pages 160–167, 2015.

[14] H. Alrehamy and C. Walker. SemLinker: automating big data integration for casual users. *Journal of Big Data*, 5(1), 2018.

[15] A. Alserafi, A. Abelló, O. Romero, and T. Calders. Towards Information Profiling: Data Lake Content Metadata Management. In *DINA Workshop, ICDM*, pages 178–185. IEEE, 2016.

[16] A. Alserafi, A. Abelló, O. Romero, and T. Calders. Keeping the data lake in form: Ds-knn datasets categorization using proximity mining. In *International Conference on Model and Data Engineering*, volume 11815 of *Lecture Notes in Computer Science*, pages 35–49. Springer, 2019.

[17] A. Alserafi, A. Abelló, O. Romero, and T. Calders. Keeping the data lake in form: Proximity mining for pre-filtering schema matching. *ACM Transactions on Information Systems (TOIS)*, 38(3: 26):1–30, 2020.

[18] A. Alserafi, T. Calders, A. Abelló, and O. Romero. DS-prox: Dataset proximity mining for governing the data lake. In *International Conference on Similarity Search and Applications*, volume 10609 LNCS, pages 284–299. Springer, 2017.

[19] M.-L. Antonie and O. R. Zaiane. Text document categorization by term association. In *2002 IEEE International Conference on Data Mining, 2002. Proceedings.*, pages 19–26. IEEE, 2002.

[20] L. G. Ares, N. R. Brisaboa, A. Ordoñez, and O. Pedreira. Efficient Similarity Search in Metric Spaces with Cluster Reduction. In *SISAP*, pages 70–84. Springer, 2012.

[21] E. Baralis, T. Cerquitelli, S. Chiusano, L. Grimaudo, and X. Xiao. Analysis of twitter data using a multiple-level clustering strategy. In *International Conference on Model and Data Engineering*, pages 13–24. Springer, 2013.

[22] M. Ben Ellefi, Z. Bellahsene, S. Dietze, and K. Todorov. Dataset Recommendation for Data Linking: An Intensional Approach. In *Proceedings of the International Semantic Web Conference: The Semantic Web. Latest Advances and New Domains*, volume 9678, pages 36–51. Springer, 2016.

[23] R. Berlanga and V. Nebot. XML Mining for Semantic Web. In *XML Data Mining: Models, Methods, and Applications*, pages 317 – 342. IGI Global, 2011.

[24] P. A. Bernstein, J. Madhavan, and E. Rahm. Generic Schema Matching , Ten Years Later. *Proceedings of the VLDB Endowment*, 4(11):695–701, 2011.

[25] B. Bilalli, A. Abelló, T. Aluja-Banet, and R. Wrembel. Towards intelligent data analysis: The metadata challenge. In *IoTBD*, pages 331–338, 2016.

[26] M. Bilenko and R. J. Mooney. Adaptive Duplicate Detection Using Learnable String Similarity Measures. In *ACM SIGKDD*, pages 39–48, 2003.

[27] A. Bilke and F. Naumann. Schema Matching using Duplicates. In *Proceedings of the 21st International Conference on Data Engineering*, pages 69–80. IEEE, 2005.

[28] A. Bogatu, A. A. A. Fernandes, N. W. Paton, and N. Konstantinou. Dataset Discovery in Data Lakes. In *2020 IEEE 36th International Conference on Data Engineering (ICDE)*, pages 709–720. IEEE, apr 2020.

[29] L. Breiman. Random Forests. *Machine Learning*, 45(1):5–32, 2001.

[30] S. Bykau, N. Kiyavitskaya, C. Tsinaraki, and Y. Velegrakis. Bridging the gap between heterogeneous and semantically diverse content of different disciplines. In *IEEE 2010 Workshops on Database and Expert Systems Applications (DEXA)*, pages 305–309, 2010.

[31] T. Calders. Three big data tools for a data scientist's toolbox. In *European Business Intelligence and Big Data Summer School*, pages 112–133. Springer, 2017.

[32] T. Calders and B. Custers. What Is Data Mining and How Does It Work? In *Discrimination and Privacy in the Information Society, Studies in Applied Philosophy, Epistemology and Rational Ethics*, volume 3, pages 27–42. Springer Berlin Heidelberg, 2013.

[33] R. Castro Fernandez, Z. Abedjan, F. Koko, G. Yuan, S. Madden, and M. Stonebraker. Aurum: A Data Discovery System. In *2018 IEEE 34th International Conference on Data Engineering (ICDE)*, pages 1001–1012. IEEE, apr 2018.

[34] A. Chapman, E. Simperl, E. Kacprzak, P. Groth, L. Koesten, G. Konstantinidis, and L.-d. Ibáñez. Dataset search : a survey. *The VLDB Journal*, 29(1):251–272, 2020.

[35] C. Chen, A. Halevy, and W.-c. Tan. BigGorilla : An Open-Source Ecosystem for Data Preparation and Integration. *IEEE Data Engineering Bulletin*, 41(2):10–22, 2018.

[36] Z. Chen, H. Jia, J. Heflin, and B. D. Davison. Generating Schema Labels through Dataset Content Analysis. In *Companion of the The Web Conference 2018 on The Web Conference 2018 - WWW '18*, pages 1515–1522, 2018.

[37] J. Cheney, S. Chong, N. Foster, M. Seltzer, and S. Vansummeren. Provenance: a future history. In *Proceedings of the 24th ACM SIGPLAN conference companion on Object oriented programming systems languages and applications*, pages 957–964, 2009.

[38] X. Chu, I. F. Ilyas, and P. Papotti. Discovering denial constraints. *Proceedings of the VLDB Endowment*, 6(13):1498–1509, 2013.

[39] J. A. C. Cruz, S. E. Garza, and S. E. Schaeffer. Entity Recognition for Duplicate Filtering. In *SISAP*, pages 253–264. Springer, 2014.

[40] M. D'Aquin and N. Jay. Interpreting data mining results with linked data for learning analytics: motivation, case study and directions. In *LAK '13 Proceedings of the Third International Conference on Learning Analytics and Knowledge*, pages 155 – 164. ACM, 2013.

[41] H. R. de Oliveira, A. T. Tavares, and B. F. Lóscio. Feedback-based data set recommendation for building linked data applications. In *Proceedings of the 8th International Conference on Semantic Systems - I-SEMANTICS '12*, page 49. ACM, 2012.

[42] D. Deng, R. Castro, F. Ziawasch, A. Sibo, A. Elmagarmid, I. F. Ilyas, S. Madden, M. Ouzzani, and N. Tang. The Data Civilizer System. In *8th Biennial Conference on Innovative Data Systems Research (CIDR '17)*, 2017.

[43] D. Deng, A. Kim, S. Madden, and M. Stonebraker. SilkMoth: An Efficient Method for Finding Related Sets with Maximum Matching Constraints. *Proceedings of the VLDB Endowment*, 10(10):1082–1093, 2017.

[44] S. Duan, A. Fokoue, O. Hassanzadeh, A. Kementsietsidis, K. Srinivas, and M. J. Ward. Instance-based matching of large ontologies using locality-sensitive hashing. In *International Semantic Web Conference*, pages 49–64. Springer, 2012.

[45] S. Džeroski and B. Ženko. Is Combining Classifiers with Stacking Better than Selecting the Best One ? *Machine learning*, 54(3):255–273, 2004.

[46] J. Ellis and M. J. Ward. Exploring Big Data with Helix : Finding Needles in a Big Haystack. *ACM SIGMOD Record*, 43(4):43–54, 2015.

[47] R. C. Fernandez, Z. Abedjan, S. Madden, and M. Stonebraker. Towards Large-Scale Data Discovery. In *Proceedings of the International Workshop on Exploratory Search in Databases and the Web - ExploreDB*, pages 3–5. ACM, 2016.

[48] K. Figueroa and R. Paredes. List of Clustered Permutations for Proximity Searching. In *SISAP*, pages 50–58. Springer, 2013.

[49] Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *J. Comput. Syst. Sci.*, 55(1):119–139, 1997.

[50] J. Friedman, T. Hastie, R. Tibshirani, et al. Additive logistic regression: a statistical view of boosting (with discussion and a rejoinder by the authors). *The annals of statistics*, 28(2):337–407, 2000.

[51] T. Furche, G. Gottlob, L. Libkin, G. Orsi, and N. W. Paton. Data wrangling for big data: Challenges and opportunities. In *EDBT*, volume 16, pages 473–478, 2016.

[52] E. Gallinucci, M. Golfarelli, and S. Rizzi. Schema profiling of document-oriented databases. *Information Systems*, 75:13–25, 2018.

[53] C. Giebler, C. Gröger, E. Hoos, H. Schwarz, and B. Mitschang. Leveraging the Data Lake: Current State and Challenges. In C. Ordonez, I.-Y. Song, G. Anderst-Kotsis, A. M. Tjoa, and I. Khalil, editors, *Big Data Analytics and Knowledge Discovery (DaWaK)*, volume 11708 of *Lecture Notes in Computer Science*, pages 250–265, Cham, 2019. Springer International Publishing.

[54] A. Halevy, F. Korn, N. F. Noy, C. Olston, N. Polyzotis, S. Roy, and S. E. Whang. Goods : Organizing Google's Datasets. In *Proceedings of the ICMD, ACM*, pages 795–806, 2016.

[55] E.-H. S. Han, G. Karypis, and V. Kumar. Text categorization using weight adjusted k-nearest neighbor classification. In *Pacific-asia conference on knowledge discovery and data mining*, pages 53–65. Springer, 2001.

[56] O. Hassanzadeh, F. Chiang, H. C. Lee, and R. J. Miller. Framework for evaluating clustering algorithms in duplicate detection. *Proceedings of the VLDB Endowment*, 2(1):1282–1293, 2009.

[57] O. Hassanzadeh, S. Duan, A. Fokoue, A. Kementsietsidis, K. Srinivas, and M. J. Ward. Helix : Online Enterprise Data Analytics. In *WWW '11 Proceedings of the 20th international conference companion on World wide web*, pages 225–228, 2011.

[58] R. Hauch, A. Miller, and R. Cardwell. Information Intelligence : Metadata for Information Discovery , Access , and Integration. In *Proceedings of the 2005 ACM SIGMOD international conference on Management of data*, pages 793–798, 2005.

[59] Y. He, K. Chakrabarti, T. Cheng, and T. Tylenda. Automatic Discovery of Attribute Synonyms Using Query Logs and Table Corpora. In *Proceedings of the 25th International Conference on World Wide Web - WWW '16*, pages 1429–1439, New York, New York, USA, 2016. ACM.

[60] Y. He, K. Ganjam, and X. Chu. SEMA-JOIN: joining semantically-related tables using big table corpora. *Proceedings of the VLDB Endowment*, 8(12):1358–1369, aug 2015.

[61] H. Hentech, M. S. Gouider, and A. Farhat. Clustering heterogeneous data streams with uncertainty over sliding window. In *International Conference on Model and Data Engineering*, pages 162–175. Springer, 2013.

[62] J. Herlocker, J. A. Konstan, L. G. Terveen, and J. T. Riedel. Evaluating Collaborative Filtering Recommender Systems. *ACM Transactions on Information Systems (TOIS)*, 22(1):5–53, 2004.

[63] M. Hewasinghage, J. Varga, A. Abelló, and E. Zimányi. Managing polyglot systems metadata with hypergraphs. In *International Conference on Conceptual Modeling*, pages 463–478. Springer, 2018.

[64] W. H. Inmon, C. Imhoff, and R. Sousa. *Corporate information factory*. John Wiley & Sons, 2002.

[65] M. Interlandi, K. Shah, S. D. Tetali, M. A. Gulzar, S. Yoo, M. Kim, T. Millstein, and T. Condie. Titian: Data Provenance Support in Spark. *Proc. VLDB Endow.*, 9(3):216–227, 2015.

[66] M. Jacomy, T. Venturini, S. Heymann, and M. Bastian. ForceAtlas2, a continuous graph layout algorithm for handy network visualization designed for the Gephi software. *PLoS ONE*, 9(6):1–12, 2014.

[67] S. R. Jeffery, M. J. Franklin, and A. Y. Halevy. Pay-as-you-go user feedback for dataspace systems. In *Proceedings of the ACM SIGMOD international conference on Management of data - SIGMOD '08*, page 847, 2008.

[68] P. Jovanovic, O. Romero, A. Simitsis, and A. Abelló. Integrating etl processes from information requirements. In *International Conference on Data Warehousing and Knowledge Discovery*, pages 65–80. Springer, 2012.

[69] S. Kandel, J. Heer, C. Plaisant, J. Kennedy, F. Van Ham, N. H. Riche, C. Weaver, B. Lee, D. Brodbeck, and P. Buono. Research directions in data wrangling: Visualizations and transformations for usable and credible data. *Information Visualization*, 10(4):271–288, 2011.

[70] S. Kandel, R. Parikh, and A. Paepcke. Profiler: Integrated statistical analysis and visualization for data quality assessment. In *AVI '12 Proceedings of the International Working Conference on Advanced Visual Interfaces*, pages 547–554, 2012.

[71] J. Kim, Y. Peng, N. Ivezic, and J. Shin. An Optimization Approach for Semantic-based XML Schema Matching. *International Journal of Trade, Economics and Finance*, 2(1):78 – 86, 2011.

[72] R. Kohavi. The Power of Decision Tables. In *ECML*, pages 174–189, 1995.

[73] H. Köpcke and E. Rahm. Frameworks for entity matching: A comparison. *Data & Knowledge Engineering*, 69(2):197–210, 2010.

[74] S. Krishna and S. Bhavani. An efficient approach for text clustering based on frequent itemsets. *European Journal of Scientific Research*, 42(3):385–396, 2010.

[75] S. Kruse, T. Papenbrock, H. Harmouch, and F. Naumann. Data Anamnesis : Admitting Raw Data into an Organization. *Bulletin of the IEEE Computer Society Technical Committee on Data Engineering*, pages 8–20, 2016.

[76] S. Lacoste-Julien, K. Palla, A. Davies, G. Kasneci, T. Graepel, and Z. Ghahramani. SiGMa: Simple Greedy Matching for Aligning Large Knowledge Bases. In *Proceedings of the 19th ACM SIGKDD international conference*, pages 572–580, 2013.

[77] M. L. Lee, L. H. Yang, W. Hsu, and X. Yang. Xclust: clustering XML schemas for effective integration. In *Proceedings of the international conference on Information and knowledge management*, pages 292–299. ACM, 2002.

[78] T. M. Lehmann, M. O. Güld, T. Deselaers, D. Keysers, H. Schubert, K. Spitzer, H. Ney, and B. B. Wein. Automatic categorization of medical images for content-based retrieval and data mining. *Computerized Medical Imaging and Graphics*, 29(2-3):143–155, 2005.

[79] O. Lehmberg and C. Bizer. Stitching web tables for improving matching quality. *Proceedings of the VLDB Endowment*, 10(11):1502–1513, aug 2017.

[80] M. R. Llave. Data lakes in business intelligence : reporting from the trenches. *Procedia Computer Science*, 138:516–524, 2018.

[81] J. Lokoc, P. Cech, J. Novak, and T. Skopal. Cut-Region : A Compact Building Block for Hierarchical Metric Indexing. In *SISAP*, pages 85–100. Springer, 2012.

[82] A. Maccioni and R. Torlone. KAYAK: A Framework for Just-in-Time Data Preparation in a Data Lake. In *International Conference on Advanced Information Systems Engineering*, pages 474–489. Springer International Publishing, 2018.

[83] J. Madhavan, P. a. Bernstein, and E. Rahm. Generic Schema Matching with Cupid. *VLDB*, 1:49–58, 2001.

[84] H. Mahgoub, N. Ismail, and F. Torkey. A Text Mining Technique Using Association Rules Extraction. *International Journal of Computational Intelligence*, 4(1):21–28, 2008.

[85] H. A. Mahmoud and A. Aboulnaga. Schema clustering and retrieval for multi-domain pay-as-you-go data integration systems. In *Proceedings of the ACM SIGMOD International Conference on Management of data*, pages 411–422. ACM, 2010.

[86] C. D. Manning, P. Raghavan, and H. Schütze. *An Introduction to Information Retrieval*. Cambridge University Press, USA, 2009.

[87] M. Mazuran, E. Quintarelli, and L. Tanca. Data Mining for XML Query-Answering Support. *IEEE Transactions on Knowledge and Data Engineering*, 24(8):1393–1407, 2012.

[88] V. M. Megler, D. Maier, and S. Member. Are Data Sets Like Documents ?: Evaluating Similarity-Based Ranked Search over Scientific Data. *IEEE Transactions on Knowledge and Data Engineering*, 27(1):32–45, 2015.

[89] J. Miller. Open Data Integration. *PVLDB*, 11(12):2130–2139, 2018.

[90] S. Moawed, A. Algergawy, A. Sarhan, A. Eldosouky, and G. Saake. A Latent Semantic Indexing-Based Approach to Determine Similar Clusters in Large-scale. *New Trends in Databases and Information Systems*, pages 267–276, 2014.

[91] K. Morton, M. Balazinska, D. Grossman, and J. Mackinlay. Support the Data Enthusiast : Challenges for Next-Generation Data-Analysis Systems. *Proceedings of the VLDB Endowment*, 7(6):453–456, 2014.

[92] K. Murthy, P. M. Deshpande, A. Dey, M. Mohania, and D. P. Jennifer. Exploiting Evidence from Unstructured Data to Enhance Master Data Management. *Proceedings of the VLDB Endowment*, 5(12):1862–1873, 2012.

[93] S. Nadal, A. Abelló, O. Romero, S. Vansummeren, and P. Vassiliadis. MDM: Governing evolution in big data ecosystems. *Advances in Database Technology - EDBT*, 2018-March:682–685, 2018.

[94] S. Nadal, K. Rabbani, O. Romero, and S. Tadesse. ODIN: A dataspace management system. In *Proceedings of the ISWC 2019 Satellite Tracks (Posters & Demonstrations, Industry, and Outrageous Ideas) co-located with 18th International Semantic Web Conference*, volume 2456 of *CEUR Workshop Proceedings*, pages 185–188. CEUR-WS.org, 2019.

[95] F. Nargesian, E. Zhu, R. J. Miller, K. Q. Pu, and P. C. Arocena. Data lake management: challenges and opportunities. *Proceedings of the VLDB Endowment*, 12(12):1986–1989, aug 2019.

[96] F. Nargesian, E. Zhu, K. Q. Pu, and R. J. Miller. Table union search on open data. *Proceedings of the VLDB Endowment*, 11(7):813–825, 2018.

[97] R. P. D. Nath, K. Hose, T. B. Pedersen, and O. Romero. Setl: A programmable semantic extract-transform-load framework for semantic data warehouses. *Information Systems*, 68:17–43, 2017.

[98] F. Naumann. Data profiling revisited. *ACM SIGMOD Record*, 42(4):40–49, 2014.

[99] V. Nebot and R. Berlanga. Finding association rules in semantic web data. *Knowledge-Based Systems*, 25(1):51–62, 2012.

[100] A. Oliveira, G. Tessarolli, G. Ghiotto, B. Pinto, F. Campello, M. Marques, C. Oliveira, I. Rodrigues, M. Kalinowski, U. Souza, L. Murta, and V. Braganholo. An efficient similarity-based approach for comparing XML documents. *Information Systems*, 78:40–57, 2018.

[101] M. Ota, H. Müller, J. Freire, and D. Srivastava. Data-driven domain discovery for structured datasets. *Proceedings of the VLDB Endowment*, 13(7):953–967, mar 2020.

[102] T. Papenbrock and F. Naumann. Data Profiling with Metanome. *Proceedings of the VLDB Endowment - Proceedings of the 41st International Conference on Very Large Data Bases*, 8(12):1860–1863, 2015.

[103] M. Patella and P. Ciaccia. Approximate similarity search : A multi-faceted problem. *Journal of Discrete Algorithms*, 7(1):36–48, 2009.

[104] J. Pei, J. Hong, and D. Bell. A novel clustering-based approach to schema matching. In *Proceedings of the international conference on Advances in Information Systems*, pages 60–69. Springer, 2006.

[105] M. Piernik, D. Brzezinski, and T. Morzy. Clustering XML documents by patterns. *Knowledge and Information Systems*, 46(1):185–212, 2016.

[106] B. Quinto. Big Data Governance and Management. In *Next-Generation Big Data*, pages 495–506. Apress, Berkeley, CA, 2018.

[107] C. Quix and R. Hai. Data Lake. *Encyclopedia of Big Data Technologies*, pages 552–559, 2019.

[108] C. Quix, R. Hai, and I. Vatov. GEMMS: A Generic and Extensible Metadata Management System for Data Lakes. In *Proceedings of the CAiSE'16 Forum at the International Conference on Advanced Information Systems Engineering*, pages 129–136, 2016.

[109] E. Rahm. Towards large-scale schema and ontology matching. In *Schema matching and mapping*, pages 3–27. Springer Berlin Heidelberg, 2011.

[110] E. Rahm. The Case for Holistic Data Integration. In *ADBIS*, pages 11–27, 2016.

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

[112] B. Ranjbar-sahraei, J. Efremova, H. Rahmani, T. Calders, K. Tuyls, and G. Weiss. HiDER: Query-Driven Entity Resolution for Historical Data. *Machine Learning and Knowledge Discovery in Databases*, 9286:281–284, 2015.

[113] F. Ravat and Y. Zhao. Data lakes: Trends and perspectives. In S. Hartmann, J. Küng, S. Chakravarthy, G. Anderst-Kotsis, A. M. Tjoa, and I. Khalil, editors, *International Conference on Database and Expert Systems Applications (DEXA)*, volume 11706 of *Lecture Notes in Computer Science*, pages 304–313, Cham, 2019. Springer International Publishing.

[114] S. Sakr and A. Y. Zomaya, editors. *Encyclopedia of Big Data Technologies*. Springer, 2019.

[115] A. B. Salem, F. Boufares, and S. Correia. Semantic Recognition of a Data Structure in Big-Data. *Journal of Computer and Communications*, 02(09):93–102, 2014.

[116] V. Santos, F. A. Baião, and A. Tanaka. An architecture to support information sources discovery through semantic search. In *IEEE International Conference on Information Reuse and Integration (IRI)*, pages 276–282, 2011.

[117] A. Sarma, L. Fang, N. Gupta, A. Halevy, H. Lee, F. Wu, R. Xin, and C. Yu. Finding related tables. In *Proceedings of the 2012 international conference on Management of Data - SIGMOD '12*, pages 817–828, New York, New York, USA, 2012. ACM Press.

[118] P. Sawadogo and J. Darmont. On data lake architectures and metadata management. *Journal of Intelligent Information Systems*, jun 2020.

[119] P. Sawadogo, É. Scholly, C. Favre, É. Ferey, S. Loudcher, and J. Darmont. Metadata Systems for Data Lakes: Models and Features. In T. Welzer, J. Eder, V. Podgorelec, R. Wrembel, M. Ivanović, J. Gamper, M. Morzy, T. Tzouramanis, J. Darmont, and A. Kamišalić Latifić, editors, *New Trends in Databases and Information Systems ADBIS*, volume 1064 of *Communications in Computer and Information Science*, pages 405–416, Cham, 2019. Springer International Publishing.

[120] S. Scherzinger, M. Klettke, and U. Störl. Managing Schema Evolution in NoSQL Data Stores. In *Proceedings of the 14th International Symposium on Database Programming Languages*, 2013.

[121] P. Shvaiko. A Survey of Schema-based Matching Approaches. *Journal on Data Semantics*, 3730:146–171, 2005.

[122] K. Smith, L. Seligman, A. Rosenthal, C. Kurcz, M. Greer, C. Macheret, M. Sexton, and A. Eckstein. "Big Metadata": The Need for Principled Metadata Management in Big Data Ecosystems. In *DanaC'14 Proceedings of Workshop on Data Analytics in the Cloud*, pages 13:1—-13:4, 2014.

[123] B. Spahiu, C. Xie, A. Rula, A. Maurino, and H. Cai. Profiling Similarity Links in Linked Open Data. In *Proceedings of the 7th International Workshop on Data Engineering meets the Semantic Web (DESWeb), ICDE*, pages 103–108, 2016.

[124] R. Steorts, S. Ventura, M. Sadinle, and S. Fienberg. A Comparison of Blocking Methods for Record Linkage. In *International Conference on Privacy in Statistical Databases*, pages 253–268, 2014.

[125] M. Stonebraker, D. Bruckner, I. F. Ilyas, G. Beskales, M. Cherniack, S. B. Zdonik, A. Pagan, and S. Xu. Data Curation at Scale: The Data Tamer System. In *6th Biennial Conference on Innovative Data Systems Research (CIDR)*, 2013.

[126] F. M. Suchanek, S. Abiteboul, and P. Senellart. PARIS : Probabilistic Alignment of Relations , Instances , and Schema. *Proceedings of the VLDB Endowment*, 5(3):157–168, 2011.

[127] P.-N. Tan, M. Steinbach, and V. Kumar. *Introduction to data mining*. Pearson Education, 2006.

[128] I. Terrizzano, P. Schwarz, M. Roth, and J. E. Colino. Data Wrangling: The Challenging Journey from the Wild to the Lake. In *7th Biennial Conference on Innovative Data Systems Research CIDR'15*, 2015.

[129] V. Theodorou, A. Abelló, M. Thiele, and W. Lehner. A framework for user-centered declarative etl. In *Proceedings of the 17th international workshop on data warehousing and OLAP*, pages 67–70, 2014.

[130] R. Touma, O. Romero, and P. Jovanovic. Supporting Data Integration Tasks with Semi-Automatic Ontology Construction. In *Proceedings of the ACM Eighteenth International Workshop on Data Warehousing and OLAP , DOLAP '15*, pages 89–98, 2015.

[131] J. Vanschoren, J. N. van Rijn, B. Bischl, and L. Torgo. OpenML: networked science in machine learning. *ACM SIGKDD Explorations Newsletter*, 15(2):49–60, 2014.

[132] J. Varga, O. Romero, and T. B. Pedersen. Towards Next Generation BI Systems : The Analytical Metadata Challenge. *Data Warehousing and Knowledge Discovery - Lecture Notes in Computer Science*, 8646:89–101, 2014.

[133] M. Yakout, K. Ganjam, K. Chakrabarti, and S. Chaudhuri. InfoGather: entity augmentation and attribute discovery by holistic matching with web tables. In *Proceedings of the 2012 international conference on Management of Data - SIGMOD '12*, pages 97–108, New York, New York, USA, 2012. ACM Press.

[134] M. Zhang and K. Chakrabarti. InfoGather+: semantic matching and annotation of numeric and time-varying attributes in web tables. In *Proceedings of the 2013 international conference on Management of data - SIGMOD '13*, pages 145–156, New York, New York, USA, 2013. ACM.

Dataset Proximity Mining for Supporting Schema Matching and Data Lake Governance

The goal of this thesis is to extract metadata and information about datasets stored in the Data Lake (DL) to support the data scientist in finding relevant sources. We explore different techniques of data profiling, holistic schema matching and analysis recommendation to support the data scientist. We propose a novel framework based on supervised machine learning to automatically extract metadata describing datasets, including computation of their similarities and data overlaps using holistic schema matching techniques. We use the extracted relationships between datasets in automatically categorizing them to support the data scientist in finding relevant datasets with intersection between their data. This is done via a novel metadata-driven technique called *proximity mining* which consumes the extracted metadata via automated data mining algorithms in order to detect related datasets and to propose relevant categories for them. We focus on flat (tabular) datasets organised as rows of data instances and columns of attributes describing the instances. We implement our proposed algorithms via a prototype that shows the feasibility of this framework. We apply the prototype in an experiment on a real-world DL scenario to prove the effectiveness and efficiency of our approach. We were able to achieve high recall rates and efficiency gains while improving the computational space and time consumption by two orders of magnitude via our proposed early-pruning and pre-filtering techniques in comparison to classical instance-based schema matching techniques. This proves the effectiveness of our proposed automatic methods, while also demonstrating improvements over human-based data analysis for the same tasks.

Keywords: *Data Lake Governance, Dataset Similarity Mining, Holistic Schema Matching, Metadata Management, Supervised Machine Learning*

A PhD thesis by
Ayman AlSerafi