



A Data-driven Approach to Large Knowledge Graph Matching

Omaima Fallatah

Information School

University of Sheffield

Submitted in Partial Fulfilment of the Requirements
for the Degree of Doctor of Philosophy

May 23, 2023

Declaration

I hereby declare that this thesis is my own work and effort and that it has not been submitted in whole or in part for any other awards or qualifications. Wherever other people work or any sources of information have been used, they have been clearly acknowledged.

Omaima Fallatah
January 2023

Acknowledgements

I express my heartfelt gratitude to ALLAH for providing me with strength and guidance throughout the completion of this thesis.

I sincerely thank my first supervisor, Dr. Ziqi Zhang, for his endless support, guidance, and inspiration as a researcher. Without his genuine encouragement and constructive feedback, this work would not have been possible. Special thanks to my second supervisor, Professor Frank Hopfgartner, for his invaluable support and advice during the development of this thesis, especially during challenging times.

I extend my gratitude to Pavel Shvaiko, Jérôme Euzenat, Ernesto Jiménez-Ruiz, Oktie Hassanzadeh, and Cássia Trojahn for their feedback and valuable discussions during the Ontology Matching workshop. I also appreciate Sven Hertling's assistance in rolling out the common knowledge graph track datasets to OAEI's evaluation platform.

I am thankful to my Information Retrieval group members for their inspiring discussions and feedback. I am grateful for my Information School family, including Sukaina, Ashwag, Nora, Anas, Sookie, Jie, Savins, Cass, and Cassie, for their support and encouragement that made my journey memorable.

I would like to express my heartfelt gratitude to my dear friends Latifah, Sabreen, Tarfah, and Hessa for their unwavering support and friendship throughout this journey. Your presence has made a significant impact, and I am truly grateful for your encouragement and companionship.

Finally, I am extremely grateful to my parents, Ahmad and Ferdos, for their unlimited emotional support, unconditional love, prayers, and belief in me. I also attribute much of my success in life to the love and support I continue to receive from my siblings, Omnia, Arwa, and Mohammed. I am also thankful to my relatives and friends for their support and encouragement.

Abstract

In the last decade, a remarkable number of open Knowledge Graphs (KGs) were developed, such as DBpedia, NELL, and YAGO. While some of such KGs are curated via crowdsourcing platforms, others are semi-automatically constructed. This has resulted in a significant degree of semantic heterogeneity and overlapping facts. KGs are highly complementary; thus, mapping them can benefit intelligent applications that require integrating different KGs such as recommendation systems, query answering, and semantic web navigation.

Although the problem of ontology matching has been investigated and a significant number of systems have been developed, the challenges of mapping large-scale KGs remain significant. KG matching has been a topic of interest in the Semantic Web community since it has been introduced to the Ontology Alignment Evaluation Initiative (OAEI) in 2018. Nonetheless, a major limitation of the current benchmarks is their lack of representation of real-world KGs. This work also highlights a number of limitations with current matching methods, such as: (i) they are highly dependent on string-based similarity measures, and (ii) they are primarily built to handle well-formed ontologies. These features make them unsuitable for large, (semi/fully) automatically constructed KGs with hundreds of classes and millions of instances. Another limitation of current work is the lack of benchmark datasets that represent the challenging task of matching real-world KGs.

This work addresses the limitation of the current datasets by first introducing two gold standard datasets for matching the schema of large, automatically constructed, less-well-structured KGs based on common KGs such as NELL, DBpedia, and Wikidata. We believe that the datasets which we make public in this work make the largest domain-independent benchmarks for matching KG classes. As many state-of-the-art methods are not suitable for matching large-scale and cross-domain KGs that often suffer from highly imbalanced class distribution, recent studies have revisited instance-based matching techniques in addressing this task. This is because such large KGs often

lack a well-defined structure and descriptive metadata about their classes, but contain numerous class instances. Therefore, inspired by the role of instances in KGs, we propose a hybrid matching approach. Our method composes an instance-based matcher that casts the schema-matching process as a text classification task by exploiting instances of KG classes, and a string-based matcher. Our method is domain-independent and is able to handle KG classes with imbalanced populations. Further, we show that incorporating an instance-based approach with the appropriate data balancing strategy results in significant results in matching large and common KG classes.

List of Publications

Parts of this Ph.D. work have been previously published, and here we provide a list of those publications:

- **Fallatah, Omaima.**, Zhang, Ziqi., & Hopfgartner, Frank. A gold standard dataset for large knowledge graphs matching. *In Ontology Matching 2020: Proceedings of the 15th International Workshop on Ontology Matching co-located with the 19th International Semantic Web Conference (ISWC 2020)* (Vol. 2788, pp. 24-35). CEUR Workshop Proceedings.
- **Fallatah, Omaima.**, Zhang, Ziqi., & Hopfgartner, Frank. A hybrid approach for large knowledge graphs matching. *In Proceedings of the 16th International Workshop on Ontology Matching (OM 2021) co-located with the 20th International Semantic Web Conference (ISWC 2021)* (Vol. 3063, pp. 37-48). CEUR-WS.
- **Fallatah, Omaima.**, Zhang, Ziqi., & Hopfgartner, Frank. The Impact of Imbalanced Class Distribution on Knowledge Graphs Matching. *In Proceedings of the 17th International Workshop on Ontology Matching (OM 2022) co-located with the 21st International Semantic Web Conference (ISWC 2022)*. CEUR-WS.
- **Fallatah, Omaima.**, Zhang, Ziqi., & Hopfgartner, Frank. KGMatcher Results for OAEI 2021. *In Proceedings of the 16th International Workshop on Ontology Matching (OM 2021) co-located with the 20th International Semantic Web Conference (ISWC 2021)* (Vol. 3063, pp. 160-166). CEUR-WS.
- **Fallatah, Omaima.**, Zhang, Ziqi., & Hopfgartner, Frank. KGMatcher+ Results for OAEI 2022. *In Proceedings of the 17th International Workshop on Ontology Matching (OM 2022) co-located with the 21st International Semantic Web Conference (ISWC 2022)*. CEUR-WS.

- Portisch, Jan., **Fallatah, Omaima.**, Neumaier, Sebastian., Jaradeh, Mohamad Yaser., & Polleres, Axel. Challenges of linking organizational information in open government data to knowledge graphs. In *International Conference on Knowledge Engineering and Knowledge Management (EKAW2020)*. (pp. 271-286). Springer.

Other publications:

- Pour, Mina Abd Nikooie; Algergawy, Alsayed; Amardeilh, Florence; Amini, Reihaneh; **Fallatah, Omaima**; Faria, Daniel; Fundulaki, Irini; Harrow, Ian; Hertling, Sven; Hitzler, Pascal; Huschka, Martin; Ibanescu, Liliana; Jiménez- Ruiz, Ernesto; Karam, Naouel; Laadhar, Amir; Lambrix, Patrick; Li, Huanyu; Li, Ying; Michel, Franck; Nasr, Engy; Paulheim, Heiko; Pesquita, Catia; Portisch, Jan; Roussey, Catherine; Tzania, Saveta; Splendiani, Andrea; Trojahn, Cássia; Vatašcinová, Jana; Yaman, Beyza; Zamazal, Ondrej; Zhou, Lu. Results of the Ontology Alignment Evaluation Initiative 2021. In: *Proceedings of the 16th International Workshop on ontology matching, co-located with the 20th International Semantic Web Conference (ISWC 2021)*.
- Pour, Mina Abd Nikooie; Algergawy, Alsayed; Buche, Patrice; Castro, Leyla J.; Chen, Jiaoyan; Dong, Hang; **Fallatah, Omaima**; Faria, Daniel; Fundulaki, Irini; Hertling, Sven; He, Yuan; Horrocks, Ian; Huschka, Martin; Ibanescu, Liliana; Jiménez- Ruiz, Ernesto; Karam, Naouel; Laadhar, Amir; Lambrix, Patrick; Li, Huanyu; Li, Ying; Michel, Franck; Nasr, Engy; Paulheim, Heiko; Pesquita, Catia; Tzania, Saveta; Shvaiko, Pavel; Trojahn, Cássia; Verhey, Chantelle; Wu, Mingfang; Yaman, Beyza; Zamazal, Ondrej; Zhou, Lu. Results of the Ontology Alignment Evaluation Initiative 2022. In: *Proceedings of the 17th International Workshop on ontology matching, co-located with the 21st International Semantic Web Conference (ISWC 2022)*.
- Portisch, Jan; Emonet, Vincent; Jaradeh, Mohamad Yaser; **Fallatah, Omaima**; Koteich, Bilal; Espinoza-Arias, Paola; Polleres, Axel. Tracking the Evolution of Public Datasets and Their Governance Bodies by Linking Open Data. In: *Knowledge Graphs Evolution and Preservation – A Technical Report from ISWS 2019*.

Contents

List of Publications	v
1 Introduction	1
1.1 Research Context	2
1.2 Ontology and Ontology Matching	3
1.3 Research Motivation	5
1.4 Research Questions	6
1.5 Research Aim and Objectives	6
1.6 Research Contributions	7
1.7 Thesis Outline	7
2 Literature Review	9
2.1 Introduction	10
2.2 Ontology Matching	10
2.2.1 Ontology Matching Techniques	13
2.2.1.1 Element-Level Techniques	16
String-based Techniques.	16
Language-based Techniques.	18
Constraint-based Techniques.	20
Formal Resource-based Techniques.	21
Informal Resource-based Techniques.	22
2.2.1.2 Structure-Level Techniques	23
Graph-based Techniques.	23
Taxonomy-based Techniques.	26
Instance-based Techniques	27
Model-based Techniques	28

2.2.2	Combining Matcher Results	29
2.2.2.1	Weighted Average and Weighted Sum	30
2.2.2.2	Average	31
2.2.2.3	Harmony weight	31
2.2.2.4	Maximum/Minimum	32
2.2.2.5	Machine Learning	32
2.2.3	Final Alignments Selection	33
2.2.3.1	Threshold Filter	34
2.2.3.2	Max-N	34
2.2.3.3	Max Delta	35
2.2.3.4	Machine Learning	35
2.2.3.5	Rule-Based	35
2.2.3.6	Iterative alignment selection	36
2.3	Ontology Matching Evaluation	37
2.3.1	Alignment Evaluation	37
2.3.2	Benchmark Datasets	39
2.4	Knowledge Graphs Matching	39
2.4.1	Knowledge Graphs on the Web	39
2.4.2	Current Knowledge Graphs Matchers	40
2.5	Synthesis of literature	43
2.6	Conclusion	45
3	Creating Gold-Standard Mappings	46
3.1	Introduction	47
3.2	Related work	48
3.3	The New NELL-DBpedia Dataset	49
3.3.1	Overview	49
3.3.2	Generating Candidate Pairs	50
3.3.3	Candidate Filtering	51
3.3.3.1	String-based Similarity Measure	52
3.3.3.2	Instance-based Similarity Measure	52
3.3.3.3	Combining Similarity Measures	55
3.3.4	Dataset Annotation	55
3.4	The Expanded Yago-Wikidata Dataset	57
3.5	Discussion	58

3.6	Conclusion	62
	Related Publication	62
4	Training Knowledge Graph Classifiers	63
4.1	Introduction	64
4.2	Related Work	65
	4.2.1 Entity Classification for Ontology Population	65
	4.2.2 Imbalance in Text Classification	67
4.3	KG Instances Classification	70
	4.3.1 KG Instances Resampling	70
	4.3.1.1 Random Undersampling	71
	4.3.1.2 TF-IDF Undersampling	72
	4.3.1.3 SMOTE	72
	4.3.1.4 TF-IDF + Oversampling	73
	4.3.1.5 TF-IDF + SMOTE	73
	4.3.1.6 Cost-based Learning	73
	4.3.2 Building the KG Instance Classifier	74
	4.3.2.1 KG Instances Feature Representation	74
	4.3.2.2 Training the KG Classifiers	75
4.4	Experiments	76
	4.4.1 Experiment Settings	76
	4.4.2 Evaluation Metrics	77
	4.4.3 Results on Common KG Datasets	78
	4.4.4 Results on the DBkWik dataset	81
	4.4.5 Results on the Web Data Commons dataset	84
4.5	Conclusion	88
	Related Publication	88
5	Matching Knowledge Graph Classes	89
5.1	Introduction	90
5.2	Related Work	91
	5.2.1 Knowledge Graphs Matching Overview	91
	5.2.2 Limitations of Current Methods	93
5.3	Approach	93
	5.3.1 Overview	93

5.3.2	Pre-processing	94
5.3.3	Instance-based Matcher	95
5.3.3.1	Exact name filter	95
5.3.3.2	KG Classifiers	96
5.3.3.3	Alignment Elicitation	96
5.3.3.4	Alignment Selection	96
5.3.4	Name Matcher	98
5.3.5	Final Alignment Selection	98
5.4	Evaluation	99
5.4.1	Experimental Settings	99
5.4.2	Datasets	99
5.4.3	KGMatcher+ Results and Discussion	100
5.4.4	The Impact of Different Matching Component	104
5.4.4.1	The Impact of Resampling KG Instances	104
5.4.4.2	The Impact of Matcher Combination	107
5.4.4.3	The Impact of the TF/IDF Threshold	109
5.5	Conclusion	109
	Related Publications	110
6	Conclusion and Future Work	111
6.1	Summary of the Thesis Contributions	111
6.2	Open Issues and Future Work	113
	Appendices	134
A	Annotation Task Instructions	135

List of Figures

1.1	Ontology example adapted from (Euzenat et al. 2011)	4
2.1	The ontology matching process adapted from (Shvaiko & Euzenat 2011)	11
2.2	Classification of ontology matching techniques adapted from (Euzenat et al. 2007)	13
2.3	The schema of two sample ontologies adapted from (Madhavan et al. 2001)	24
2.4	Similarity propagation example.	25
2.5	Examples of mapping inconsistency patterns used in ASMOV matcher adapted from (Jean-Mary et al. 2009)	29
2.6	Different variations of OM task workflow	30
2.7	Example of calculating the harmony weight adapted from Mao et al. (2010)	32
2.8	Using a decision tree to combine results in YAM++ adapted from (Ngo et al. 2011a)	33
2.9	Example of the iterative final alignment selection approach	37
3.1	The class distribution of two KGs from the two proposed datasets (NELL and Wikidata), compared to the distribution of instances in two of the largest OAEI KGs (MemoryAlpha and MemoryBeta)	61
4.1	The class distribution of the four KGs in the two used datasets after applying the exact name filter.	71
4.2	The macro f-measure of the BERT classifiers using the TF/IDF with oversampling compared to no sampling strategy on the common KG datasets	80

4.3	The classification reports of a 20 randomly sampled classes from the OAEI KG Marvel Cinematic Universe and Memory Alpha compared to two common KGs: YAGO and NELL.	83
4.4	Example table describing movies extracted from <i>Netflix.com</i>	85
5.1	The architecture of the proposed system (KGMatcher+)	94
5.2	An example of calculating the final alignment using the method in (Gulić et al. 2016)	97
5.3	The percentage of imbalanced class pairs discovered by different methods compared to KGMatcher+	105
5.4	Comparing the results of the name matcher component of KGMatcher+ against the full KGMatcher+ method.	108

List of Tables

2.1	Summary of current ontology and KG matching systems	15
2.2	the similarity value of Chair and Chairman using different edit distance methods	17
3.1	The number of classes and instances in the two proposed datasets . . .	59
3.2	Statistics of the number of classes and instances in the eight KGs from the OAEI Knowledge graph track	59
4.1	The macro average results of the three trained machine learning models on the common KG dataset.	78
4.2	The macro average results for the KG classifier using different data balancing strategies on the four common knowledge graph dataset	79
4.3	The macro average for the KG classifiers using the best-performing sampling strategy on the eight KGs from the DBkWik project	81
4.4	The distribution of instances across classes in the WDC datasets	85
4.5	Results of the KG classifiers on the WDC datasets	87
5.1	Evaluation results on the OAEI <i>Common Knowledge Graphs</i> track - NELL-DBpedia task	102
5.2	Evaluation results on the OAEI <i>Common Knowledge Graphs</i> track - YAGO-Wikidata task	102
5.3	Evaluation results on the OAEI <i>Knowledge Graphs</i> track	103
5.4	Evaluation results of different variations of KGMatcher+ each utilizing a different data balancing strategy (the best results are in bold). . . .	106
5.5	The impact of K value on KGMatcher+	109
A.1	Annotation Example	136

Chapter 1

Introduction

Contents

1.1	Research Context	2
1.2	Ontology and Ontology Matching	3
1.3	Research Motivation	5
1.4	Research Questions	6
1.5	Research Aim and Objectives	6
1.6	Research Contributions	7
1.7	Thesis Outline	7

1.1 Research Context

The amount of information shared throughout the web has been growing continuously. This has led to a growing number of data mining studies that looked at building **Knowledge Graphs (KGs)** by automatically extracting information from unstructured web pages. KG is a unique data structure for representing real-world entities in a structured and connected fashion (Heist et al. 2020). KGs contain heterogeneous yet a large amount of highly complementary facts. Therefore, they have potential in a wide range of downstream applications such as reasoning, query answering, e-commerce, and semantic web navigation (Pujara et al. 2013). KGs have gained more attention in recent years due to the progress of the Web towards the Semantic Web vision (Berners-Lee et al. 2001). The goal of the Semantic Web is to offer machine-readable and well-formatted web data and to facilitate sharing and reusing knowledge such as those annotated in ontologies and KGs (Clarkson et al. 2018).

Over the last decade, there has been significant growth in the creation and application of KGs. For instance, they are utilized by large companies such as Google, Facebook, and Microsoft. Besides such proprietary KGs, there are a number of large common KGs published based on the Semantic Web standards including DBpedia (Lehmann et al. 2015), NELL (Carlson et al. 2010), and YAGO (Suchanek et al. 2007). These KGs are domain-independent, i.e., carries data from multiple domains such as medical, music, publications, and organizations (Suchanek et al. 2012). Moreover, due to their automatically-constructed and independently-designed nature, such datasets contain overlapping and complementary facts. For instance, **Bone** and **Artery** can be classified under **BodyPart** in NELL while being classified as **AnatomicalStructure** in DBpedia. In Semantic Web, this problem is referred to as semantic heterogeneity. Research in this area have investigated this problem and many matching systems have been developed and surveyed (Euzenat et al. 2011, Otero-Cerdeira et al. 2015). KGs are often compared to ontologies, as both are a form of data representation. However, KGs are less well-structured in comparison to ontologies. The following section defines ontologies in the context of this work, as well as the ontology matching task.

1.2 Ontology and Ontology Matching

In the Semantic Web, an ontology is a formal means of representing, sharing, and reusing knowledge across semantic-web applications (Ochieng & Kyanda 2018). It defines concepts of a particular domain along with their relationships in the context of that domain following a hierarchical taxonomy structure. Moreover, unlike KGs ontologies are typically domain-dependent, i.e., it is unlikely to have an ontology that combines multiple domains. For instance, a biomedical ontology can describe the use of a specific medicine and deduces its compositions and how they are related. An ontology consists of a schema layer and a data layer. The Schema layer defines the classes that the ontology use, it is typically referred to as the **Terminology Box (T-Box)**. Classes are the domain concepts that the ontology describes, for instance, **Author** and **ConferenceMember** in Figure 1.1 below. A class can have subclasses, for instance, **Author** is a subclass of **ConferenceMember**. Instances are individuals of classes that describe real-world entities, for instance, a specific author's details. Moreover, instances are the main component of the data layer, which is identified as the **Assertion Box (A-Box)** (Zhu & Iglesias 2016). Further, classes and instances are considered entities of an ontology. Entities may have a set of mutual properties, e.g., **email** and **has_email** are properties of classes **person** and **Human**. Each property also has a data type, which specifies the format that a property can hold, e.g., **email** can be defined as **String**. The value of a property can be restricted with a range such as [2-8]. An entity can also have a name or label that annotates classes and instances, and a comment, which is a longer description of an entity. Since ontology encapsulates data to be processed or integrated into different applications, a standard language is needed to describe the knowledge. There are two knowledge representation languages that are broadly used to construct ontologies and KGs, which are the Web Ontology Language (OWL) and The Resource Description Framework (RDF) (Euzenat et al. 2007).

Ontologies are highly heterogeneous, i.e., different words are used to describe similar entities. In addition to entities having similar names, there are different sources of heterogeneity in ontologies. For instance, given two ontologies that describe the same domain, a class can only be part of one ontology and missing in another. Two classes can also be similar but not identical (e.g., **Author** and **Regular Author** in Figure 1.1). Moreover, these facts are encapsulated in ontologies that have been designed indepen-

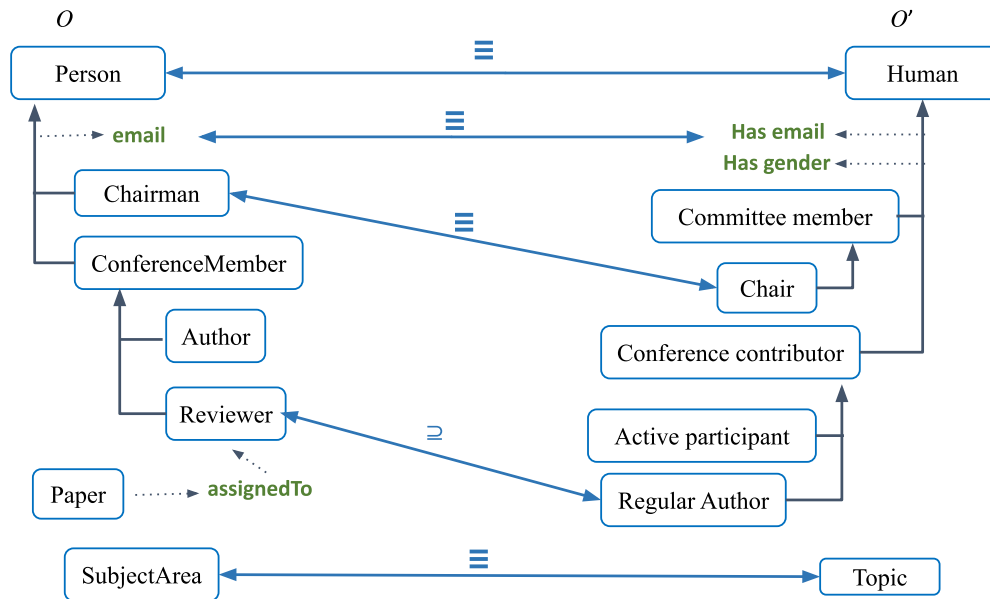


Figure 1.1: Ontology example adapted from (Euzenat et al. 2011)

dently. While it can be easy for a human to distinguish semantically similar classes, computer applications need to be trained in order to be able to understand such knowledge. As a result, applications that map heterogeneous facts have been developed for the purpose of merging the knowledge embedded in distinct ontologies. To resolve this issue, Ontology Matching (OM) was introduced, which is the process of discovering correspondences between two different ontologies (Euzenat et al. 2011). A Correspondence can be obtained by mapping similar entities (e.g., classes, instances, and properties). In Figure 1.1 above, correspondences are shown as double-ended arrows that map entities from the two ontologies. Correspondences can be either equivalence or subsumption. Equivalence (\equiv) infers that two classes are equivalent (e.g., **Chairman** and **Chair**). Whereas subsumption (\sqsubseteq, \sqsupseteq) represents more/less general relationships. For instance, **Author** is more general than the class **RegularAuthor**. Ontology matching can be targeted at one of the two ontology layers. Some methods are focused on matching the schema (T-Box), i.e., classes and properties, while others are targeted towards matching the instances or A-Box. In addition, some matching methods are capable of producing matching of both layers (Suchanek et al. 2012). The area of developing OM solutions has been gaining more attention as the number of shared ontologies and knowledge graphs are increasing (Anam et al. 2015). This problem of

semantic heterogeneity has been well studied since 2004, with matching systems being annually evaluated through the Ontology Alignment Evaluation Initiative (OAEI ¹).

1.3 Research Motivation

As we mentioned earlier common KGs are highly complementary, therefore, they are often integrated in several downstream applications such as reasoning and query answering. Despite the growth in such KGs, one problem is dealing with the quality of the data generated automatically. Due to their nature of being largely generated in a semi-automated manner, KGs are less well-formed compared to manually created and well-designed ontologies. This has resulted in continuous efforts to facilitate refining their entities by increasing their coverage (i.e., completion), and detecting errors (i.e., correctness) (Paulheim 2017). To achieve this, mapping and aligning KGs at both data and schema level is crucial. By design, KGs are known for their large number of instances (ABox). Therefore, the majority of current methods focus on matching their instances. However, recent studies have shown that the problem of matching KGs schema (TBox) remains a challenging task (Rahm & Peukert 2019, Hertling & Paulheim 2020b). Moreover, many KG matching methods use previously aligned schema to generate and refine instance matching results (Hertling & Paulheim 2020a).

While the challenges of ontology matching have been well-studied in recent years, the challenges of mapping large-scale KGs remain significant. The state-of-the-art matching methods have many limitations. **First**, a common concern with such methods is their scalability to map larger datasets. KGs are larger in scale, with millions of instances and hundreds of classes resulting from web mining efforts. **Second**, state-of-the-art matching methods are highly dependent on terminological and structural-based techniques. The reason is that ontologies in domains such as life sciences share a significant amount of terminological similarities, such as prefixes and suffixes. In contrast, KGs are often domain-independent and their classes contain information about real-world entities described with different vocabularies. **Third**, state-of-the-art systems are explicitly designed to process well-formatted ontologies, unlike KGs which mostly lack a detailed systematic structure. Due to their automatically generated nature, classes in KGs are known to be inconstant, unbalanced, and incomplete. E.g., NELL's *Country* class is arguably complete due to the limited number of instances in reality,

¹<https://oaei.ontologymatching.org>

while `Athlete` class is continuously growing. **Fourth**, the literature indicates a lack of studies that address OM methods for KGs. This has resulted from the lack of datasets that represent unbalanced, large-scale, domain-independent KGs.

1.4 Research Questions

The main question that this thesis aims to answer is *How can we effectively map the classes of large KGs?* Due to such KGs having an abundant of instances, our hypothesis is that utilizing their instances to match the schema can be beneficial. In order to verify this, we further divide this question into three subordinate research questions, each aims to focus on a specific aspect of the task. The research questions are listed below.

- **RQ1:** *What are the current KG matching benchmarks, and how can we construct one that is more representative of the large KG matching problem?*
- **RQ2:** *Given the large yet unbalanced number of instances in KG classes, how can we make use of them effectively in machine learning?*
- **RQ3:** *How can we effectively utilize instances to match KG classes while addressing the unbalanced population issue?*

1.5 Research Aim and Objectives

The aim of this research is to develop a novel matching method specifically tailored to large, semi-automatically constructed, inadequately structured, and multi-domain KGs. The key objectives of our research are:

- To investigate the current literature on matching techniques in order to develop a matching method that addresses the challenges introduced by such KGs.
- To construct large-scale golden standard benchmarks of aligned classes from real-world KGs. Those golden standards are to be used in order to evaluate the performance of our proposed methods and to be publicly available.
- To develop a novel matching approach that automatically maps the schema (T-Box) of two large and automatically created KGs by utilizing their data level (A-Box) for the matching task.

- To deliver a comparative evaluation of our method on the produced gold standard dataset as well as against state-of-the-art and current matching methods.

1.6 Research Contributions

This thesis presents a novel data-driven approach for mapping classes in large and common KGs. We contribute to the literature in a number of ways:

- An in-depth study of current ontology matching techniques, combining different techniques and evaluation tools. This analysis also includes a synthesis of state-of-the-art matching systems and their role in mapping large and common KGs.
- A benchmark consisting of two gold standard mappings constructed from four large-scale and publicly available KGs. This benchmark makes the largest domain-independent gold standard in the context of KGs schema matching. Both datasets have been added to the OAEI’s annual matching evaluation campaign in 2021.
- An approach of constructing a machine learning model that is able to classify instance names into KG classes. This method is also domain independent and is able to cope with KGs with inconsistent and imbalanced class distribution. This approach is not only relevant to KG matching but can also be adapted for other tasks such as ontology population.
- A hybrid matching method that integrates multiple matching techniques where one uses only instances data to generate matching class pairs by incorporating the use of the KG instance classifier approach.
- A prototype, named KGMatcher+, that implements our matching approach. KGMatcher+ has participated in OAEI2021 and OAEI2022 and has been ranked as the best-performing system in the task of matching large-scale and common KGs.

1.7 Thesis Outline

The remainder of this thesis is organized as follows:

Chapter 2 gives an overview of the ontology matching problem, with an in-depth analysis of current similarity techniques used by existing methods. This chapter also

covers different essential aspects of matching systems, including combining different methods and evaluation processes. It ends with a study of current state-of-the-art systems used for KGs matching and a discussion of some of their limitations.

Chapter 3 details the approach of creating benchmark datasets for matching the schema of large-scale, automatically constructed, and multi-domain KGs. The first dataset is generated from NELL and DBpedia, two KGs with high influence in many semantic web applications, but their schema has not been interlinked before. In addition, we expand the current links between the schema of YAGO and Wikidata to propose a second benchmark for this task.

Chapter 4 presents our approach to training KG instance classifiers. The classifiers are trained using instance names as training data, and they are able to predict a KG class to which an instance belongs. Further, we experiment with various approaches to balance the training data, including two new approaches that we introduce. We also perform multiple experiments on different large KGs datasets to evaluate the performance of the proposed approach.

Chapter 5 features the implementation of a hybrid matching approach for matching the schema of large-scale KGs. This includes an instance-based method that utilizes the KG classifiers trained in the aforementioned chapter. Later, we evaluate the proposed method against the state-of-the-art ontology and KGs matching systems based on different OAEI benchmark datasets, including those proposed by this work. Finally, we perform a detailed evaluation of the proposed method, including the impact of different components on its performance.

Chapter 6 summarizes the thesis with remarks of the presented work and concludes with some outlooks and future work directions.

Chapter 2

Literature Review

Contents

2.1	Introduction	10
2.2	Ontology Matching	10
2.2.1	Ontology Matching Techniques	13
2.2.2	Combining Matcher Results	29
2.2.3	Final Alignments Selection	33
2.3	Ontology Matching Evaluation	37
2.3.1	Alignment Evaluation	37
2.3.2	Benchmark Datasets	39
2.4	Knowledge Graphs Matching	39
2.4.1	Knowledge Graphs on the Web	39
2.4.2	Current Knowledge Graphs Matchers	40
2.5	Synthesis of literature	43
2.6	Conclusion	45

2.1 Introduction

In this chapter, we provide an analysis of current techniques used for the ontology matching task. An overview of the key aspects of the task is provided in Section 2.2, including different ontology matching techniques introduced in the literature. In Section 2.3, we discuss different ontology matching evaluation techniques and current benchmarks. Further, we particularly highlight the importance of mapping and aligning KGs and the current state of this task in Section 2.4. Finally, Section 2.5 summarises and classifies state-of-the-art and recent matching approaches.

2.2 Ontology Matching

Ontology matching (OM) is a pairwise comparison between two distinct ontologies, e.g., \mathcal{O} and \mathcal{O}' . It aims to find an alignment set such that $\mathcal{A}' = \{a_1, a_2, a_3, a_x\}$, which is a group of correspondences recognized as the output of the ontology matching process (Megdiche et al. 2016). As depicted in Figure 2.1, the matching system generally takes two main inputs, which are the two ontologies to be matched, and an optional input alignment. The input alignment \mathcal{A} consists of mappings that can be improved by the matching process. Otherwise, it can be used to find additional alignments between the two input ontologies. The input alignment can be produced either by a different matching system or manually by a domain expert. An alignment has a cardinality, which is the number of entities from both ontologies that can be matched (Euzenat et al. 2011). According to Shvaiko & Euzenat (2011), given a pair of ontologies, “a correspondence is a 4-tuple such that $\langle id, e, e', r, n \rangle$,” where:

1. id is the identifier for a specific correspondence;
2. e, e' are entities of the first and second ontologies. These could be classes, properties, or instances;
3. r is the relation between e and e' , i.e., equivalence (\equiv), subsumption (\sqsubseteq, \sqsupseteq);
4. n is a number between $[0,1]$ which represents the confidence value calculated by the matching method that produced the correspondence.

The key task of an ontology matching system is to determine mappings or correspondences between the entities of the source and target ontology. In order to identify

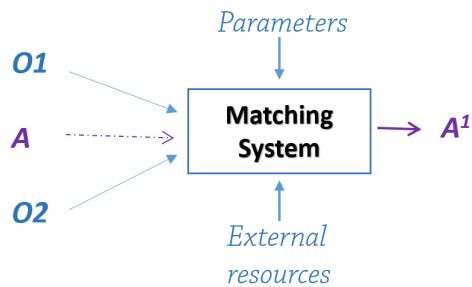


Figure 2.1: The ontology matching process adapted from (Shvaiko & Euzenat 2011)

such a result, similarity measures are utilized to give each pair of entities a number that expresses the degree of similarity between them. The similarity value is usually a number between $[0.0,1.0]$. In the example shown in Figure 1.1 above, a similarity measure applied to the concepts `SubjectArea` and `Topic` can result in a similarity score of 0.15. The matching system then needs to determine alignment based on that score.

The majority of current systems apply different basic matchers (to be detailed in Section 2.2.1), each based on a single similarity measure, to each pair of entities separately (Laadhar et al. 2017). It is then necessary to aggregate these correspondences at the end of the matching process, or at the end of each phase of the process. Basic matchers use information from one particular aspect of the ontology, such as name or structure, to calculate the similarity value. Therefore, combining a variety of basic matchers is considered common practice in ontology matching. The process of aggregating different basic matchers is an essential part of the matching process. Further discussion of this process is presented in Section 2.2.2 below.

The output of the matching process is an alignment \mathcal{A}' , a set of correspondences identified between pairs of matched ontologies. Alignments can have different cardinalities such as one-to-one, one-to-many, or many-to-many (Gulić et al. 2016). To illustrate, an alignment that only holds equivalence correspondences is considered a one-to-one alignment (i.e., one entity in an ontology \mathcal{O} can only correspond to one and only one entity in \mathcal{O}'). Both one-to-many and many-to-many alignments are known as *complex alignments*, as they can be very challenging to discover. Each correspondence holds a confidence value, n which denotes the confidence degree of the mappings produced by the algorithm (Ngo et al. 2011b). This value can be a similarity value either resulting from one similarity measure or aggregated from various measures. For example, in Fig-

ure 1.1, a matching system could produce 0.82 as a confidence value for the mapping between the two classes **Chairman** in \mathcal{O} and **Chair** in \mathcal{O}' . The correspondence in the example above could be represented as the following $\langle id_2, \mathbf{Chairman}, \mathbf{Chair}, \equiv, 0.82 \rangle$.

In addition, matching parameters are used to select and filter correspondences. The goal of using a matching parameter is to manage the quality of the matching system results. For instance, a threshold filter is utilized by the majority of state-of-the-art systems. When a threshold value is identified, any correspondence with a confidence value lower than the threshold is excluded from the final alignment (Gulić et al. 2016). Another parameter is weight, which is a value that determines the importance of different matchers. Section 2.2.3 discusses different alignment selection methods used by state-of-the-art ontology matching systems.

The use of background knowledge resources is another key aspect of the schema and ontology matching process. Many knowledge resources are currently employed to enhance the accuracy of OM systems by enriching the entities of an ontology with additional information from such resources (Nguyen & Conrad 2013). WordNet¹ and OpenThesaurus² are examples of knowledge resources that combine concepts from multiple domains organized hieratically with different semantic relations such as equal, is-a, and part-of. Section 2.2.1 covers some current matching methods that use background knowledge resources.

User involvement is an optional factor in ontology matching systems. Typically, systems produce a set of candidate mappings before deciding the final set of mappings that will contribute to the alignment. Different degrees of user involvement or automation can be adopted by ontology matching systems. A system can be interactive if the actual mapping between entities has to be confirmed by a human. Other matching systems list possible correspondences for the user to validate and manually add any mappings that the algorithm has skipped. Moreover, matching systems can be entirely automated where machine learning techniques are integrated to determine the final alignments (Laadhar et al. 2017). Further details of such an approach are provided in Section 2.2.2.

Although all the above-mentioned aspects play significant roles in the matching process, the most important part is the combination of matching techniques adopted by the ontology matching system. In the next section, we will discuss different ontology matching techniques from the literature.

¹<https://wordnet.princeton.edu/>

²<https://www.openthesaurus.de/>

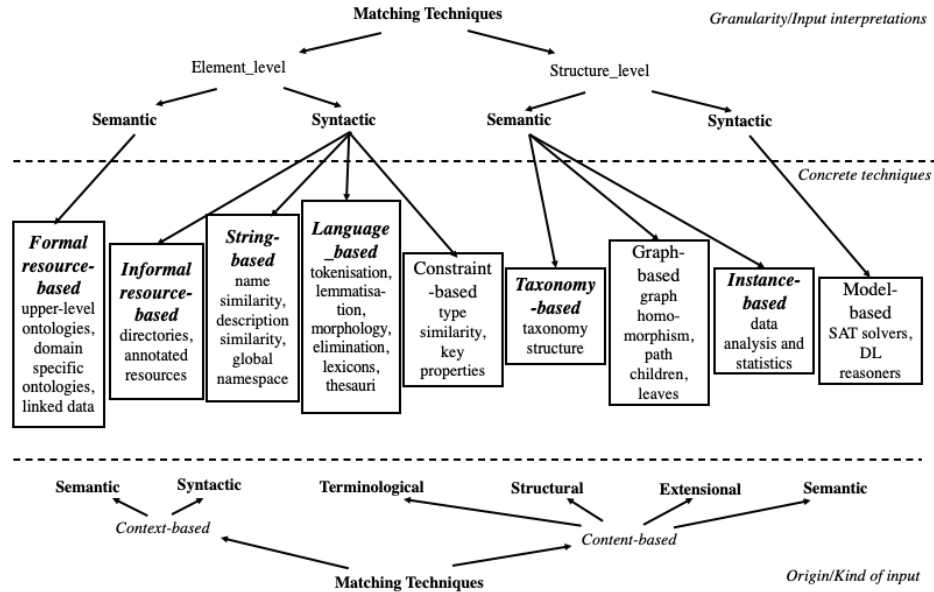


Figure 2.2: Classification of ontology matching techniques adapted from (Euzenat et al. 2007)

2.2.1 Ontology Matching Techniques

Although several basic ontology matching techniques have been introduced in the last decade, these early techniques are still widely used as essential components in more recent, complex matching systems. Such matching techniques measure the similarity of a pair of entities from two ontologies. For example, some techniques only deal with strings (i.e., textual information) while others are designed to measure the similarity of the ontology structure. As stated earlier, the majority of matching systems combine different basic matching techniques. The literature suggests different dimensions for classifying current matching techniques. In the *top-down* approach shown in Figure 2.2, OM techniques are divided into: (i) *Element-level techniques* which isolate the entities of an ontology from their structure, and relationship to other entities during the matching process, and (ii) *Structural level techniques* which take into consideration the relationship between ontology entities throughout the matching process (Euzenat et al. 2007).

On the next level, each of these techniques is further divided based on input interpretation into *Syntactic* and *Semantic* techniques. Syntactic methods strictly focus on syntactic and lexical features of surface text forms in isolation from their mean-

ings. While semantic methods interpret their inputs by using formal semantic models that help capture the meaning of words and justify their results (Otero-Cerdeira et al. 2015). Examples of semantic relationships found in ontologies that could be measured for optimal matching are synonyms (equal), hyponyms (is-a), part-of, and has-a. For example, a matcher that only uses syntactic techniques such as string-based similarity measures will not capture the relation between `car` and `vehicle`, as both are written differently. Moreover, such techniques will possibly consider words like `Cable`, `Maple`, and `Table` similar in syntax even though they have different meanings (Arnold & Rahm 2014). Hence, using a semantic-based measure, an OM system will be able to produce mappings that include entities with different lexical or syntactic forms.

Matching techniques can also be classified based on their inputs to *Content-based* and *Context-based* methods. Context-based methods take as an input two ontologies to be matched and an external resource that contains information about these ontologies and the relationships between their entities. Context-based techniques can be further divided into *semantic* techniques which extract information about ontology entities from external resources, and *syntactic*, which does not focus on the semantics of entities. On the other hand, content-based techniques are further categorized based on the actual data that the similarity measure takes as input. Content-based similarity measures are: (i) *terminological techniques* methods which deal with textual content, (ii) *Structural techniques* that handle the focus on the structure of ontologies, (iii) *Semantics techniques* which extract semantic using logic-based models, and finally (iv) *Extensional techniques* which take instances as input for the process of matching classes or properties.

Many studies have analysed and surveyed current OM systems characteristics (Euzenat et al. 2011, Otero-Cerdeira et al. 2015, Anam et al. 2015, Mohammadi et al. 2018). Table 2.1 summarises current matching systems, particularly those mentioned in this chapter. The system column represents its name, the alignment level describes whether the system maps only the schema (T-box) or the instances (A-Box) of the ontology, element, and structural level techniques lists the techniques used by each system on both levels, and additional techniques' column lists any machine learning, neural networks or rule-based methods applied by the system. In the following sections, we will follow the top-down classification approach proposed by Euzenat et al. (2007), to discuss these techniques.

Table 2.1: Summary of current ontology and KG matching systems

System	Alignment-level	Element-Level techniques	structure-level techniques	Additional techniques
ASMOV (Jean-Mary et al. 2009)	T-Box	string similarity semantic similarity (WordNet, UMLS)	graph similarity, internal structure similarity using constraint similarity, instance-based similarity	
AgreementMaker (Cruz et al. 2009)	T-Box and A-Box	string similarity semantic similarity (WordNet)	graph similarity (descendant/sibling)	
SAMBO (Lambrix & Tan 2006)	T-Box	string similarity semantic similarity (WordNet, UMLS)	Taxonomy similarity	machine learning (Naive Bayes)
AML (Faria et al. 2013)	T-Box and A-Box	string similarity semantic similarity background knowledge	graph similarity (anchor-based)	cardinality filter coherent filter
COMA/COMA++ (Do & Rahm 2002)	T-Box and A-Box	string similarity semantic similarity (WordNet) constraint similarity	graph similarity instance-based similarity	rule-based
CroMatcher (Gulić et al. 2016)	T-Box	string similarity constraint similarity	graph similarity (subclass/superclass) internal structure similarity based on constraint instance-based similarity (using TF/IDF)	
Yam++ (Ngo & Bellahsene 2012)	T-Box and A-Box	string similarity	graph similarity similarity flooding (similarity propagation)	machine learning (decision trees, SVM Naive Bayes)
Falcon-AO (Hu & Qu 2008)	T-Box	string similarity semantic similarity (WordNet)	graph similarity	
GLUE (Doan et al. 2004)	T-Box	constraint similarity	taxonomy similarity instance-based	machine learning (Naive Bayes)
LogMap (Jiménez-Ruiz & Cuenca Grau 2011)	T-Box and A-Box	string similarity semantic similarity (WordNet, UMLS)	graph similarity model/logic-based (DL)	rule-based
LSSOM (Nguyen & Conrad 2015)	T-Box	string similarity semantic similarity (WordNet)	graph similarity	
XMap/XMap++ (Djeddi & Khadir 2010)	T-Box	string similarity semantic similarity (WordNet)	graph similarity internal structure similarity using constraint similarity	
POMap/POMap++ (Laadhar et al. 2017)	T-Box	string similarity semantic similarity	graph similarity (sibling-subclass)	self-supervised machine learning
RiMOM (Zhang et al. 2016)	T-Box and A-Box	string similarity semantic similarity (WordNet)	similarity flooding (similarity propagation) instance-based similarity	
Malfom-SVM (Ichise 2008)	T-Box and A-Box	string similarity semantic similarity (WordNet)	graph similarity	machine learning (SVM)
PRIOR+ (Mao et al. 2010)	T-Box and A-Box	string similarity semantic similarity	similarity flooding (similarity propagation)	neural networks
Wiktionary (Portisch et al. 2019)	T-Box and A-Box	string similarity semantic similarity	-	
ALOD2Vec (Portisch & Paulheim 2018)	T-Box and A-Box	string similarity semantic similarity	-	
LSMatch (Sharma et al. 2021)	T-Box	string similarity semantic similarity	-	
OTMapOnto (An et al. 2021)	T-Box	semantic similarity	-	
FCAMapKG (Chang et al. 2019)	T-Box and A-Box	string similarity	Formal Concept Analysis	
ATBox (Hertling & Paulheim 2020a)	T-Box and A-Box	string similarity semantic similarity	similar neighbors instance-based similarity	cardinality filter type filter
DOME (Hertling & Paulheim 2019)	T-Box and A-Box	string similarity semantic similarity	instance-based similarity	doc2vec
TOM/Fine-TOM (Kossack et al. 2022, Knorr & Portisch 2022)	T-Box and A-Box	string similarity	-	transformer models (sentence-BERT)

2.2.1.1 Element-Level Techniques

These techniques take textual information of ontology entities as input to measure their similarity score. Typically, textual data can be found in entity names, labels, and comments of a well-structured ontology. Moreover, element-level matchers do not take instances data (in the case of schema matching) or entity relationships as input to their similarity measures. Using element-level techniques is the most common approach utilized by state-of-the-art ontology matching systems.

String-based Techniques. Matching systems that use these techniques work with textual information, particularly strings found in the descriptions of an ontology. Such methods treat the string as a sequence of letters in order to measure the similarity of entity names, labels, and comments. Thus, the more syntactically similar these descriptions are, the higher the chance that two entities are similar and can be aligned. String-based techniques use syntactic features such as *edit-based*, *suffix*, *prefix*, or *n-gram* to determine the similarity of two strings.

An edit-based similarity technique relies on counting the minimum number of operations needed to be applied to one string to be transformed into another. This includes the number of characters to be added, removed, replaced, or swapped. There are different types of edit distance measures used in this field, such as edit distance family and *Jaro* distance family (Xiao et al. 2008). In the edit distance family, different methods have different rules to assign operation costs. For instance, the *Levenshtein* method (Levenshtein et al. 1966) gives each operation a cost of 1 while the *Hamming* method (Tan et al. 2006) only gives the replacement operation a cost of 1. Other edit distance methods, such as *Smith-Waterman*, allocate different costs to each operation (Ngo 2012). On the other hand, the *Jaro* distance family computes the similarity score based on the number of the common characters between two strings which also are shorter than half of the longer string. An extension of this method is *Jaro-Winkler*, which focus on the importance of the common initials (or prefix) between two strings to calculate the similarity score. To illustrate, taking two terms **Chair** and **Chairman** from the ontology example in Figure 1.1, the result of applying different string distance methods are shown in Table 2.2.

Automatic Semantic Matching of Ontology with Verification or ASMOV is an ontology matching system that combines different string-based similarity measures, including the Levenshtein method. This method was particularly utilized to calculate

Table 2.2: the similarity value of `Chair` and `Chairman` using different edit distance methods

Method	Edit Distance	Similarity Value
Levenshtein	3	5
Hamming	3	5
Smith-Waterman	3	2
Jaro_winkler	0.075	0.925
Jaro	0.125	0.875

the similarity score of entity comments. However, since comments usually consist of sentences, ASMOV applies edit distance algorithms to tokens extracted from the actual comments. First, a list of tokens from each comment is returned. Then, the number of token operations to transfer one comment into another is calculated by using the Levenshtein method (Jean-Mary et al. 2009). To demonstrate, assuming that the concepts `Author` and `RegularAuthor`, have the following comments respectively `{a person who contributes to a conference}` and `{a person who often contributes to a specific conference}`. After the tokenization process, `Author` will result in seven tokens and `RegularAuthor` will result in nine tokens. The total number of token operations required to transform a comment into another is two insertion operations. The lexical similarity is calculated as:

$$L_{sm} = 1 - \frac{2 * edit\ distance}{9 * max\ tokens} = 0.78$$

This result is later weighted to be considered as a part of the system’s process to find the final alignment. The same method is also applied in LSSOM, by Nguyen & Conrad (2015), a semantic-based ontology matching system that also uses string-based similarity measures.

Suffix-based measures calculate the similarity value of two strings by examining if both strings share a suffix, for example, `CommitteeMember` and `ConferenceMember` share the suffix `Member`. It can also check if one of them ends with the other one, for example, `Phone` and `MobilePhone`. Similarly, a prefix-based method determines the similarity value based on whether the two strings share the same prefix (e.g., `real` and `reality`). It can also check if one string starts with the other (e.g., `chair` and `chairman`). Suffix and Prefix measures are considered useful techniques to match

compound words that happen to appear often in ontology labels. For example, S-Match by Giunchiglia et al. (2012), is a schema-based ontology matching system that matches graph-based ontologies. S-Match combines different string-based similarity measures including suffixes and prefixes on its element-level matcher.

Concerning the n-gram measure, it computes the similarity between two strings by calculating their mutual n-grams (i.e., a subsequence of n characters). For example, a 3-gram (trigram) of the word `chairman` are `cha-hai-air-irm-rma-man`, while for the term `chair` the trigrams are `cha-hai-air`. These two terms share three trigrams and depending on the algorithm that implements this measurement, they will be considered similar. Many matching systems utilize n-gram as a similarity measure, such as SAMBO by Lambrix & Tan (2006), a state-of-the-art system that matches biomedical ontologies. In a comparison study of different matching systems on biomedical ontologies conducted by Mohammadi et al. (2018), n-gram measure has outperformed other string-based measures in terms of recall.

In summary, string-based techniques are ideal when high-quality textual information is available in the ontologies to be matched. Nonetheless, they can produce false mappings by matching similar strings that refer to different objects. For example, a string similarity measure will match the term `Date` in two different ontologies, even though the first one is referring to a time of an event while the other one is referring to the fruit `date`. Another issue typically associated with these techniques is matching synonyms. Such methods are incapable of detecting the similarity between synonyms, e.g., `Topic` and `Subject`. In this case, semantic-based or semantic similarity measures can be more useful.

Language-based Techniques. These methods are often grounded on techniques adopted from the field of Natural Language Processing (NLP). Language-based techniques can be divided into two types. The first group focuses on the internal word structure, while the other one relies on external background knowledge resources such as dictionaries to calculate the similarity score between two entities.

The first category of language-based techniques makes use of the linguistic features of words (Shvaiko & Euzenat 2005). Typically, they are used prior to applying string-based methods to enhance their results. Methods such as lemmatization can be applied to entity labels in order to return the original word in a singular form (Cheatham & Hitzler 2013). For example, for the word `athletes`, the original word `athlete` will be

returned. Words can also be stemmed to their original roots by removing suffixes and prefixes. Sentences, on the other hand, can be tokenized by removing stop words and common words like **a**, **the**, **is**. Furthermore, words can be checked for spelling mistakes that may affect the matching results. Such techniques can enhance the matching results by extracting meaningful terms from text. In the case of irregular words where the general linguistics rules do not apply, a thesaurus is needed to return the original form of that word.

The second category of language-based techniques utilizes external linguistic resources to discover semantic relations between words. Examples of linguistic resources include common thesauri and lexicons, which are useful to find synonyms and hyponyms of words (Euzenat et al. 2007). A common linguistic resource that is widely used among OM systems is WordNet (Miller 1995). It organises words into well-defined synsets (concepts) and nouns (instances) and it currently has over 117,000 synsets linked to other concepts to form meaningful pairs. It pairs words to their synonyms, where words that carry different meanings are paired with all possible synonyms. Therefore, all WordNet meaning pairs are unique. Beside synonyms, the other type of relationship that WordNet and other similar lexical databases have is hyponyms, also known as IS-A relationship (Jurafsky & Martin 2018). This relation is formed between a general synset such as `kitchen_tools` and a more specific noun, e.g., `oven` or `blender`. WordNet also carries information about other named entities like countries, companies, and public figures like athletes (Nguyen & Conrad 2015).

LSSOM is an ontology matching system that combines three different similarity measures in a parallel form. The semantic similarity is calculated by the method proposed in (Nguyen & Conrad 2013) which explores entity relationships, semantics, and structure in the WordNet hierarchy. Considering two concepts C_1 and C_2 , the method will examine: (i) the direct path between C_1 and C_2 , (ii) the connection between each concept and the other concept's children (i.e., descendants concepts), (iii) the connection between the two concepts parents and children concepts, and (iv) the connection between one concept's parents and the other classes children and vice versa. WordNet is used to find out the semantic similarity between two entities by first tokenizing class names, labels, and comments. Then, each token is placed in a group of its source type, e.g., tokens from an ontology comment are grouped into one set. Then, the items of each group are compared to all tokens in the corresponding group. Their similarity is measured based on the method explained above. Next, tokens are ranked to choose

the best similarity. For instance, given two comments in two ontologies entities e_1 and e_2 respectively, where $Comment(\mathbf{Topic}) = \text{“The title of a journal article.”}$ and $Comment(\mathbf{SubjectArea}) = \text{“A topic of an article in a journal”}$, after the tokenization process and removing stop words. Two sets of comments will be created, $Comment_1 = \{title, journal, article\}$, and $Comment_2 = \{topic, journal, article\}$. Comparing the first token `title` to tokens in $Comment_2$ will result in a higher similarity score with `topic`. Finally, the semantic similarity of two entities, i.e., `Topic` and `SubjectArea`, is the average of the best similarities from both comment sets (Nguyen & Conrad 2015).

In terms of using a domain-specific thesaurus, many systems have implemented this technique, especially those aimed to match medical and biomedical ontologies such as ASMOV (Jean-Mary et al. 2009). It uses UMLS³, a domain-specific thesaurus for medical terminologies. The system combines string, structural and instance-based matchers and was adapted to perform semantic verification by using a specialized thesaurus to increase the mapping accuracy. This approach of matching two ontologies to a domain-based thesaurus has proved an improvement of the overall result of the final alignment of many biomedical OM systems such as SAMBO (Lambrix & Tan 2006) and ServOMap (Diallo 2014). Both UMLS and WordNet are part of a system targeted to enhance the semantic similarity of current ontology matching systems called STROMA. Arnold & Rahm (2014) proposed STROMA as an attempt to encourage ontology matching systems to expand their mapping beyond equivalence (equal) and subsumption (is-a) relationships. It involves using different linguistic strategies including compound words and background knowledge to measure the semantic similarity of ontologies in different domains and different languages. STROMA was integrated with some state-of-the-art matching systems like COMA (Aumueller et al. 2005). In the latter, it was used to enrich the final results with more complex alignments of type “is-a” and “part-of” with a semantic-based technique.

Constraint-based Techniques. As mentioned earlier, ontologies are generally distinct in terms of their nature and structure because they have been independently designed and engineered. While previously mentioned similarity techniques can be generally applied to most ontologies, constraint-based techniques are more specific to the scheme of an ontology. This type of method can only be applied to a specific type of

³<https://www.nlm.nih.gov/research/umls/index.html>

ontologies where information about data constraint is specified (e.g., datatype, number of attributes, and ranges of values). To illustrate, if an entity is defined with a specific datatype such as `Float` and `Double` (i.e., decimal point numbers), a datatype-based similarity measure can be applied to calculate the similarity score. Thus, `Float` and `Double` will have a higher similarity score than having one of the properties of the type `String` or `Integer`.

Constraint-Based techniques can also be applied where range values can occur, as well as for cardinality of lists and arrays (Laadhar et al. 2017). A system can combine different types of constraints to find mappings. A valid example is Schema Matcher Boosting or (SMB) (Marie & Gal 2008), a schema matcher that uses the domain and name constraints to map the schema of two ontologies. For example, when the system measures the similarity of two attributes `Dropoff_Date` and `Return_Date`, their string similarity is low because the two names are different. However, adding a constraint-based measure will evaluate their domains too, e.g., in this case, they are both a drop list of days/months. Thus, their constraint similarity score can improve their overall similarity value. Another useful example of using data type to measure similarity is in the Cupid approach (Madhavan et al. 2001). It is a structure-based ontology matching system that considers the datatype and domain compatibility between ontology entities by asserting them a similarity score between 0 and 0.5. This value is later combined with other matching technique results.

Formal Resource-based Techniques. These techniques utilize formal resources during the matching process. Formal resources can be an upper-level ontology, a domain-specific ontology, or an alignment of previously mapped ontologies. Different from language-based techniques, these methods support the reuse of ontologies and other datasets. This includes using their alignments as input to the matching process. This practice is referred to as alignment reuse (Otero-Cerdeira et al. 2015).

Upper-level formal ontologies typically cover knowledge from general domains. Examples of upper-level ontologies are the Suggested Upper Merged Ontology (SUMO⁴), and The Descriptive Ontology for Linguistic and Cognitive Engineering (DOLCE⁵). These ontologies are logic-based, therefore, they can be used by OM systems to infer knowledge. For example, the DOLCE ontologies are implemented with first-order logic and cover multi-domain knowledge bases including WordNet. Using upper-level

⁴<https://www.ontologyportal.org/>

⁵<http://www.loa.istc.cnr.it/dolce/overview.html>

ontologies allows an OM system to discover semantically similar entities across two different ontologies. For instance, two entity names `apple` can be mapped only if they belong to the same class, i.e, either `Fruit` or `ElectronicCompany`.

In the work done by Mascardi et al. (2009), three algorithms were built to prove that the matching process can benefit from using upper-level ontologies. Both SUMO and DOLCE ontologies were used to improve the performance of a matching algorithm named OMViaUO. One of the implemented methods aims to match two ontologies in a parallel form. The first parallel process matches the two input ontologies to an upper ontology separately. Then, it aggregates the alignments from the two sub-matching processes. The second parallel process applies a series of element-level matchers such as Levenshtein and Jaro-Winkler, and WordNet to measure both string and semantic-based similarities. In the aggregation phase, the final alignments are generated by keeping pairs with confidence values higher than an assigned threshold and excluding other pairs.

Incorporating LOD data is another practice of using formal resources for ontology matching. The LOD is a collection of datasets shared among the web as a result of the linked data practice (Gruetze et al. 2012). Generally, the information in such datasets is used as an external resource of knowledge during the matching process. Extracting semantics can lead to more accurate matching results, especially with multi-domain ontologies. Linked data in the form of KG can also be used to extract the meaning of ontology entities. Examples include using DBpedia (Lehmann et al. 2015), a domain-independent KG which will be further introduced in Section 3.2, to find semantic relationships between ontology entities are BLOOMS (Jain et al. 2010) and LDOA (Kachroudi et al. 2011). The latter measures three types of similarities which are terminological (i.e., string-based), structural, and semantic, and then aggregates them in the final results. Semantic similarity is measured by aligning ontologies entities such as labels to DBpedia’s entities. To illustrate, if two different entities from the two input ontologies map to the same DBpedia entity, then they are aligned.

Informal Resource-based Techniques. Similar to formal resource-based methods, these techniques map ontologies by utilizing external resources. Different from previous techniques, these methods exploit informal resources (Otero-Cerdeira et al. 2015). Examples of informal resources can be large corpora such as reference books and encyclopedias, or a collection of pictures that a certain ontology annotates. For

example, matching two classes if they are annotating the same picture (Euzenat et al. 2007). However, to the best of our knowledge, examples of ontology matching systems that implement such a technique are yet to be implemented.

2.2.1.2 Structure-Level Techniques

Different from element-level matchers that only focus on entities as separate objects, structure-based matchers consider the context of ontology entities during the matching process. These techniques are divided into *graph*, *taxonomy*, *model* and *instance*-based techniques.

Graph-based Techniques. These are built to handle ontologies as a tree of nodes. In this context, nodes are entities of the ontology, while edges are the properties connecting the entities. Graph-based techniques compute the similarity of a pair of entities by examining their position in a graph structure. This section will discuss some current structure-based similarity measures used by existing ontology matching systems.

In an early work, Madhavan et al. (2001) represented a method that matches hierarchical schemas, known as the Cupid approach, where the matching process is divided into two phases. The first one is named *linguistic similarity* (*lsim*) which measures the similarity of entities based on their names, as well as their datatype and domain constraints (i.e., using element-level techniques). The second phase targets the structure of the ontologies, and it is called *structural similarity* (*ssim*). Structural similarity for a pair of entities is calculated based on the context of both entities in the two schemas. The following example illustrates the Cupid method using the sample of two ontologies PO and PurchaseOrder, shown in Figure 2.3 below. In the structural phase, the algorithm will be able to match classes POShipTo and DeliverTo based on the similarity of their entities: City and Street. Furthermore, it will be able to match City and Street under POShipTo to City and Street under DeliverTo rather than entities under InvoiceTo. A semantic-based matcher will confirm that the words Deliver and Ship in the classes' names are synonyms. Then both similarity results (i.e., *lsim* and *ssim*) are aggregated into a single score *weighted similarity* (*wsim*). Finally, two entities are declared similar if their *wsim* is higher than a predefined threshold.

The intuitions behind the Cupid approach are: (i) Two leaves (atomic entities) are similar if they are linguistically matched by an element-level matcher, and if their ancestors and siblings are highly similar; (ii) Two non-leaf entities are considered similar

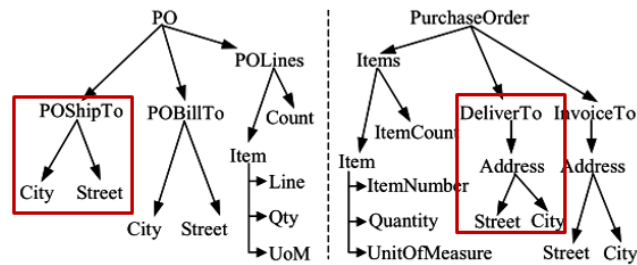


Figure 2.3: The schema of two sample ontologies adapted from (Madhavan et al. 2001)

if they are matched by an element-level matcher, and when their immediate children are also similar; (iii) For non-leaf schemas, they are aligned if they share similar roots (ancestors) even when their immediate children are not similar (Madhavan et al. 2001). Since then, this algorithm has been adopted by some state-of-the-art systems, such as COMA++ (Aumueller et al. 2005).

Similarity Flooding (SF) is a graph-based algorithm that finds correspondences between two schema graphs and produces a set of mapping between their entities. It works with different representations of schemas, catalogs and data structures, referred to as a model in the context of this method. The idea is that the similarity of two nodes highly depends on the similarity of their neighbours. Thus, if the neighbours of the two compared nodes are matched, then they are mapped. Many OM systems have adapted the SF algorithm to match the hierarchical structure of graph-based ontologies. For example, the structure-based matcher in YAM++ (Ngo & Bellahsene 2012) was built based on this algorithm with some changes that calculate the similarity propagation of each node in a graph. Similarity propagation means that a similar pair of nodes propagate their similarity to their neighbouring pairs of nodes (Ngo 2012). For example, in Figure 2.4 below, assuming that the entities (**Organization**, **Company**) and (**Manager**, **Executive**) have been mapped by a structure-based matcher. Similarity propagation means that these two mappings can influence the similarity of other mappings. Thus, the matching system can also produce the candidate mapping between (**Company**, **Organization**) and (**Executive**, **Manager**).

Another algorithm for measuring a similar structure in ontologies is GMO (Graph Matching for Ontologies). It takes two pairs of already matched ontology and uses the external matching result to discover further similar pairs by measuring their structural similarities. The input alignments come from another matcher, typically element-level

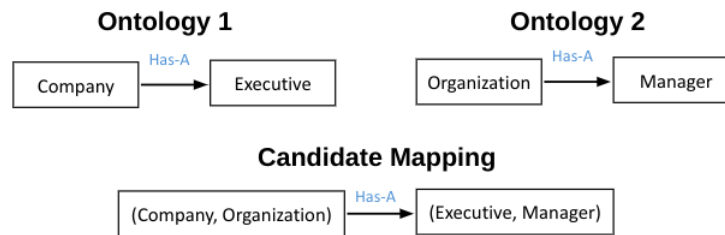


Figure 2.4: Similarity propagation example.

matchers (Hu et al. 2005). GMO is implemented in Falcon-AO along with an element-level matcher named LMO (Linguistics Matching for Ontologies). Alignments resulted from both matchers are integrated into the final alignments of Falcon-AO. However, since GMO needs the result of a linguistic matcher to extract more mappings, the alignments from LMO are used as the input alignments for GMO to run. Then, the alignments from both algorithms are evaluated at the last phase of the system according to specific rules. For instance, if the linguistic similarity of a pair of entities is lower than a predefined low threshold such as 0.01, and their structural similarity is very low too, the alignment will not be included in the final alignments (Hu & Qu 2008).

Cruz & Sunna (2008) developed an ontology matching system that combines two structural matchers: the Descendant’s Similarity Inheritance (DSI), and the Sibling’s Similarity Contribution (SSC). Both methods were primarily used to improve the mapping resulted from the element-level matcher. The first method (DSI) is used to target the relationships between parent nodes (concepts). This method was implemented as a component of AgreementMaker (Cruz et al. 2009), a state-of-the-art matching system. AgreementMaker starts by using basic similarity measures (i.e., element-level) to measure the similarity of two concepts. However, due to the heterogeneous aspect of ontologies, the similarity value can be lower than the identified threshold even though the entities are similar. In this case, another similarity measure (i.e., structure-based) can be used to verify a match or not a match. Moreover, element-level matchers can produce false mappings with higher similarity values, which often need to be verified.

In order to resolve this issue, Sunna & Cruz (2007) proposed two methods named DSI and SSC. DSI considers two concepts similar if their parents are also similar. According to Sunna & Cruz (2007), the DSI method allows parent concepts to impact their similarity on their descendant concepts. The second method SSC is similar to DSI, but mostly focuses on sibling concepts (i.e., similar siblings affect the similarity of their

ancestor concepts). Both methods have their own limitations in terms of scalability. For example, SSC performs inefficiently when the number of a sibling in the graph is large, while DSI causes the runtime to be significantly higher when processing deeper ontology graphs (Cruz & Sunna 2008).

Another graph-based similarity measure is an anchor-based method applied in Anchor-flood (Seddiqui & Aono 2009). It is a system that considers adjacent concepts and relations during the matching process. It starts from an anchor which is a pair of similar concepts (i.e., similarity produced by a lexical matcher) from the two ontologies to be matched. Then, the algorithm incrementally collects blocks of related entities by considering subclasses, super classes, and siblings closer to that anchor until the algorithm cannot discover any further entities to be matched. The matching process is based on using different lexical matchers as well as another open-source semantic matching algorithm, e.g., S-Match (Giunchiglia et al. 2012). The resulting matched pair is considered as an anchor for the next iteration. Finally, fragments of both matched ontologies are produced as mapping results.

Taxonomy-based Techniques. Taxonomy-based (or structural analysis) techniques are often associated with matching the schemas of ontologies. These methods assert the similarity between the two concepts by analysing their attributes and their relationships with other concepts. The intuition behind such an algorithm is that two concepts in two ontologies are similar if they share similar attributes and if the two concepts have identical neighbour classes. The similarity of attributes is determined by using an element-level similarity measure. This technique is highly similar to the graph-based technique, except that graph-based algorithms focus on the graphical representation of an ontology structure by considering paths and leaves of the graph classes (Martinez-Gil et al. 2012). While taxonomy-based methods focus on one type of relationship, which is the specialization relation (is-a) that identifies if a class is a specific part of another class (Otero-Cerdeira et al. 2015). The literature shows a lack of examples of applying this method in state-of-the-art OM systems. The most relative example can be in SAMBO (Lambrix & Tan 2006) where the structural matcher does not follow a graph-based structure to match concepts. Rather, it focuses on two types of relations between classes to determine their similarity: (is-a) and (part-of) relationships. Further, as stated by Otero-Cerdeira et al. (2015), these techniques are considered a special case of graph-based ones that only focuses on specialization relationship, thus

only a few OM systems incorporate them.

Instance-based Techniques These techniques are also known as *extensional* methods. They aim to measure the similarity of two classes or properties based on the similarity of their annotated instances, i.e., extensions.

A number of OM studies have incorporated different instance-based techniques. The first approach is by analysing the common instances between two classes. The idea is that if two classes share a distinguished number of shared instances, then they are semantically similar and therefore can be mapped. For example, if the same papers are published in different publication ontologies, or if the same products are found in different ontologies with different category names. In this case, it is easier for an ontology matching system to detect classes with overlapping instances in order to determine their similarity score. This is often done by utilizing methods that measure the degree of overlapping data in two sets, such as *Dice*, *Jaccard*, and *Cosine* measures (Thada & Jaglan 2013). Two useful examples of this application are in the studies carried out by Kirsten et al. (2007) and Thor et al. (2007), where this technique was used to map the schema of large life science ontologies and product catalogues. A recent example of using such a method is called DOME (Hertling & Paulheim 2019), which is an ontology matching system that incorporates an instance-based matcher. After aligning instances with an element-level matcher, classes are then mapped based on their instances overlap using the *dice* method.

SILAS (Ossewaarde 2007) is another ontology matching system based on measuring the overlapping instances of two library thesauruses. The system gives each pair of similar sets a confidence value based on: (i) two concepts are highly similar if they share a significant number of overlapping instances, (ii) in case of a size difference, where one concept holds more instances than the other, they are considered similar if the average of their overlapped instances is higher than the non-joint instances, (iii) if two concepts are identified with the same name by an element-level matcher they are also matched.

Nonetheless, measuring the overlap between class instances may not be as sufficient in large-scale tasks as well as when classes are not as well-balanced in terms of the number of instances. Later studies have experimented with more scalable approaches, such as using virtual documents that combine class instances, as proposed in (Rahm 2011). A virtual document containing weighted terms in a vector representation is

generated, where terms are its dimensions. Then, document similarity is measured by weighting the words in documents using the Term Frequency – Inverse Document Frequency or TF-IDF (Joachims 1996). Finally, the cosine similarity of the documents in a common vector space is calculated as the similarity value of each class pair. A valid example of applying this approach is in RiMOM (Zhang et al. 2016) a state-of-the-art OM system.

To overcome the problem of using an instance-based approach when ontologies lack mutually annotated instances, Wang et al. (2008) proposed a machine learning approach. Their method does not require shared instances across the two ontologies, as machine learning models can be used to infer semantic similarity by training a classification model. Each class is represented as a feature vector using the metadata of its annotated instances. Then cosine similarity is measured between source and target ontology class vectors in order to determine their similarity scores. The classifiers are trained using previously labelled classes annotated by ontology experts. However, a common concern with this approach is that it requires high-quality and well-balanced training data in order to produce good matching results (Castano et al. 2011). This process can be very expensive and time-consuming, particularly for large-scale matching tasks (Wang et al. 2008). Other state-of-the-art systems that follow a similar approach are SAMBO and GLUE.

Model-based Techniques These techniques utilize the semantic interpretation of the ontologies to be matched. The aim of these methods is to infer new mappings between two ontologies by using an initial set of alignments and reasoning techniques. The initial mapping is identified as a set of semantic relations between concepts of the two ontologies to be matched (Castano et al. 2011).

Utilizing Description Logic (DL) for matching is an example of applying this technique. DL belongs to the field of knowledge representation, which offers a different formalization of data to ensure knowledge reusability. Therefore, DL is ideal for intelligent information retrieval systems which are based on reasoning and inferring knowledge. Matching systems that apply this technique are often based on OWL-DL. OWL-DL is similar to OWL, with some characteristics that allow ontologies described with it to benefit from DL techniques. For instance, the T-Box of OWL-DL ontologies has an added component, called role, which describes the relations between ontology concepts (Sánchez-Ruiz et al. 2011).

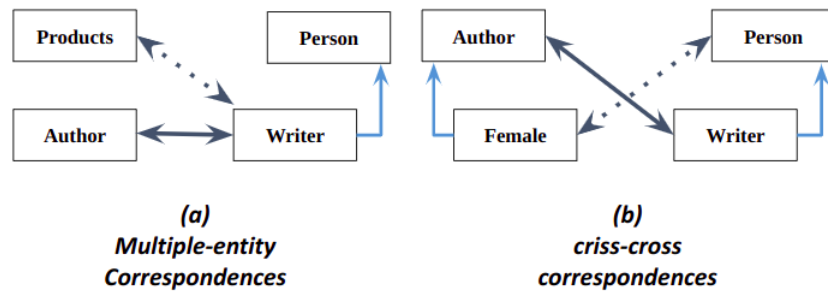


Figure 2.5: Examples of mapping inconsistency patterns used in ASMOV matcher adapted from (Jean-Mary et al. 2009)

Examples of using such a model for matching ontologies are limited to mapping verifications. A number of OM systems use DL reasoning to improve the matching results by discovering and repairing inconstant mappings. For instance, ASMOV is a method that matches ontologies with a semantic verification phase. ASMOV uses DL to ensure that the final alignments do not carry any semantic inconsistency by using different inconsistency patterns. Examples of such patterns are depicted in Figure 2.5 above. In this example, an entity is matched to two different entities in the other ontology with (1:1) cardinality. However, a one-to-one cardinality type specifies that only one entity from the first ontology can be matched to one entity in the second ontology. Therefore, the matching system will eliminate the alignment with the lower confidence value (e.g., **Product** and **Writer**) and keep the other one. In the case of equal confidence values, both alignments will be kept. Other OM systems that apply similar approaches to validate alignments are LogMap (Jiménez-Ruiz & Cuenca Grau 2011), YAM++ (Ngo & Bellahsene 2012), and KOSIMap (Reul & Pan 2010).

2.2.2 Combining Matcher Results

As stated earlier, an OM system is typically a combination of different matching techniques or basic matchers, where each focuses on a specific aspect of the ontology. Ontologies are unique, and therefore, the use of diverse techniques is important to cover all the important aspects of the ontology for better matching results. As illustrated in Figure 2.6, three common workflow approaches are used in the literature: *sequential*, *parallel (or independent)* and *hybrid*.

In the sequential workflow, the alignments from one matcher are used as input for the subsequent matcher. Consequently, the final alignment is an enhanced result of

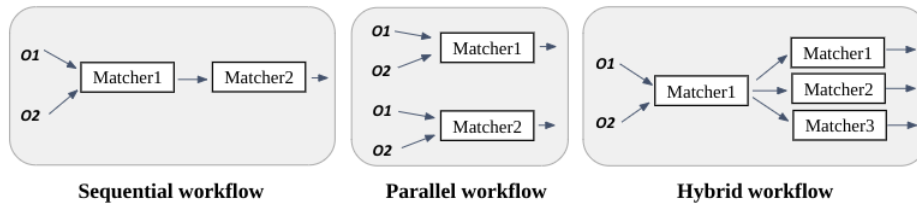


Figure 2.6: Different variations of OM task workflow

a sequence of matchers. A relevant example is to use the output of an element-level matcher as an input to a structure-level matcher, such as in the Cupid approach, and POMap (Laadhar et al. 2017). However, Ochieng & Kyanda (2018) states that this workflow is insufficient due to the long execution time that makes it unsuitable for large-scale tasks. Hence, the sequential workflow is not a common practice among ontology matching systems.

The second matcher workflow is a parallel approach where basic matchers are applied independently to the ontologies. Then, results from all matchers are aggregated in the next phase of the matching process. This is a common practice in the majority of matching systems, in particular, those targeted toward large-scale ontologies. This workflow gives the flexibility of applying different matcher while reducing the execution time since matchers can be applied separately (Ochieng & Kyanda 2018). Examples of applying such a workflow are found in CroMatcher (Gulić et al. 2016), ServOMap (Diallo 2014). In a typical parallel workflow, after the basic matcher’s finish execution, the similarity values resulting from all basic matchers need to be combined. It is often done by an aggregation function such as average, weighted average or sum, harmony weighted average, minimum, maximum, and machine learning. These methods will be discussed in the following subsections.

Finally, a hybrid workflow combines both approaches discussed above. An example of a hybrid workflow is Falcon-AO, which uses a sequential and a parallel approach to combine basic matchers.

2.2.2.1 Weighted Average and Weighted Sum

This method relies on the weight given to each similarity measure, i.e., it does not consider all basic matchers results as equally important. The weight of a matcher can be considered a measure of the quality of its results. However, defining the weight of each matcher is a challenge when using this method. In some systems, the weighting

factor is defined manually based on the expertise of the developers, or based on the previous testing of the system, such as in YAM++. Other matching systems apply an automatic weighing method to assign a weight for each basic matcher. For instance, in CroMatcher, the aggregated similarity is produced by multiplying the similarity values from each matcher with the weight factor produced by their method named AutoWeight++, which can either calculate a weighted average or sum. In the case of a weighted average, such as in CroMatcher, each similarity value will be multiplied by the weight of its specific matcher and divided by the number of basic matchers. In the case of weighted sum, the weighted similarity scores are simply added up, such as in LSSOM (Nguyen & Conrad 2015) and XMap (Djeddi & Khadir 2010).

2.2.2.2 Average

This method can be viewed as a special case of the previous method, except that it assumes that all basic matcher results are equally important (i.e., the weight factor for all matchers is 1). The aggregated similarity is calculated by summing up all the similarity values and then dividing the sum by the total number of matchers. An example of using this method is in COMA (Do & Rahm 2002).

2.2.2.3 Harmony weight

The term harmony is a measure of the reliability and importance of different similarities. Thus, in order to declare that a pair of entities (a, b) is truly matched, their similarity value should be higher than the similarity value of other pairs that contain either a or b , assuming that the alignment cardinality is (1:1). This means that a and b are a perfect match. Prior to calculating the harmony value, the system will aggregate the similarity values of each pair of entities into a similarity matrix. According to Reul & Pan (2010), a similarity matrix has a dimension of $|\mathcal{N} \times \mathcal{M}|$, where \mathcal{N} and \mathcal{M} are the numbers of the entities in the ontology O_1 and O_2 respectively. Generated by a certain matcher K , a similarity matrix represents the similarity values between entities of O_1 and O_2 .

The harmony value is the highest similarity value in its corresponding row and column divided by the number of entities in both ontologies. In the example shown in Figure 2.7 above, only the values in red circles are the higher similarity value in both row and column. For instance, no similarity value is selected in the first row of the metric because the higher value in that row is 0.22, but this value is not the highest

	Composite	Book	Proc.	Monography	Collection
Reference	.11	0	.22	.1	.1
Book	.22	1	.2	.2	.2
Proceedings	.18	.09	36	.09	.18
Monograph	.11	.22	.11	9	.1
Collection	.3	.2	.1	.1	1

$Harmony = 4/5 = 0.8$

Figure 2.7: Example of calculating the harmony weight adapted from Mao et al. (2010)

among those in the same column. Hence, the pair with this value will be omitted. Consequently, the harmony value is 4 divided by the number of entities, which is 5. This harmony number is then used as a weight for the aggregated similarity calculation similar to the previous method but with higher accuracy. This method is also known as *Adaptive Similarity Aggregation* and was developed as part of a matching system named PRIOR+ (Mao et al. 2010).

2.2.2.4 Maximum/Minimum

The maximum method returns the maximum similarity value returned by two basic matchers. Therefore, in the case of conflicting similarity values, it is considered an optimistic way to combine matchers. Alternatively, the minimum method is pessimistic, as it returns the minimum similarity value between two matchers. Both approaches do not require any measuring parameters compared to the previous methods. Moreover, they assume that one of the basic matchers is more important and reliable because of the elimination of other matchers' results. However, a higher similarity value return by a matcher does not necessarily mean that it is reliable. For instance, the two words, `cable` and `table` will have a higher string similarity score compared to their semantic similarity score. Therefore, this method can lead to a false mapping since the description provided in a comment can hold some semantic relations which indicate that they are not supposed to be mapped. Thus, these methods are only useful when the matchers are highly certain and strong (Ngo 2012). Both methods are part of the COMA++ system.

2.2.2.5 Machine Learning

Another form of combining multiple basic matchers is by utilizing machine learning techniques. This method was introduced as a component of the state-of-the-art system

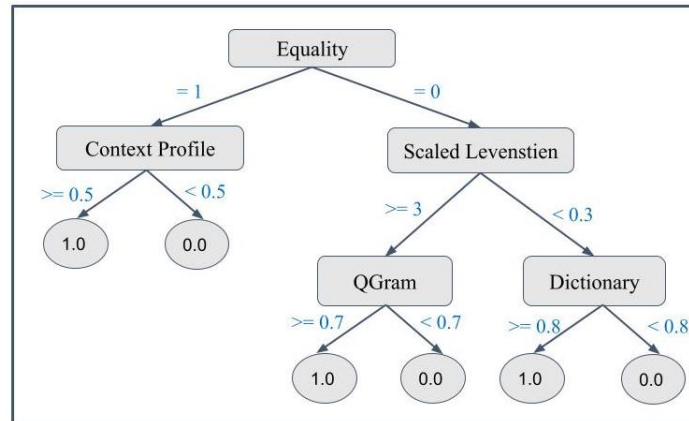


Figure 2.8: Using a decision tree to combine results in YAM++ adapted from (Ngo et al. 2011a)

YAM++. Decision trees were used along with SVM in YAM++ to combine different element-level matchers. The system was implemented with a GUI where the user can choose one of the models. In the decision tree, a non-leaf entity is a similarity metric produced by an element-level matcher such as *Levenshtein distance* while a leaf entity represents the similarity value between $[0.0 - 1.0]$. As Figure 2.8 depicts, the system takes a pair of entities from the source and target ontology and starts from the root of the tree. At the non-leaf node, the system will calculate the similarity value of each pair and compare the value with the conditions on the ongoing edges to decide the path of the process. The same process continues until a leaf node (i.e., decision) is reached (Ngo et al. 2011a).

2.2.3 Final Alignments Selection

After combining results from basic matchers, the next phase of the matching process is to produce the final alignment. This part of the process is a key aspect of OM process, as it determines the final results of the matching system. Alignment selection methods can be applied in order to filter incorrect and inconstant alignments after the matching phase. It is also worth mentioning that these methods can be used during the matching process by basic matchers to eliminate any incorrect mappings to be sent as input to the subsequent matcher. For instance, in POMap, a threshold is used to filter element-level matchers' alignments before processing the structure-level matcher. However, these methods are often introduced after the matching phase to select the final mapping candidate for the user. This section will discuss current selection methods

introduced in the literature such as *threshold*, *Max-N*, *machine learning MaxDelta*, *Rules*, and an *Iterative* approach.

2.2.3.1 Threshold Filter

A threshold is a value used by OM systems to filter invalid mappings from being included in the final alignments. It is a value between 0.0 and 1.0 often identified by the system developer and can be modified in the testing phase in order to obtain better mapping results. To illustrate, any pair of entities aligned with a similarity measure lower than the predefined threshold will be eliminated. The quality of the matching systems depends on the threshold value. Thus, the lower the value of the threshold (e.g., 0.1) the higher is the chance of including incorrect pairs, and the higher the threshold (e.g., 0.9) the less is the quality of the produced mappings (Gulić et al. 2016). It is also worth mentioning that the majority of state-of-the-art systems use a threshold filter. For instance, XMap (Djeddi & Khadir 2010) uses a predefined threshold and produces two sets of alignments, the first one represents the final alignment and the second is for the user to evaluate. The mappings selected by the user are then added to the final alignment set. Other examples of using a threshold filter are LSSOM, AgreementMaker and Falcon-AO.

2.2.3.2 Max-N

This selection method allows only n of the greatest correspondences of each entity to be included in the final alignments. When n is equal to 1, this method is referred to as Max-1 which means that only the correspondence with the highest confidence value of a specific entity is selected for the final alignment. For example, assuming that a matching system has resulted with the following correspondences for the pairs of entities that include the concept **Chair**: $\langle id1, Chairman, Chair, \equiv, 0.88 \rangle$, $\langle id2, Chair, Author, \equiv, 0.33 \rangle$, $\langle id3, Reviewer, Chair, \equiv, 0.12 \rangle$. Only the alignment with the highest confidence value (i.e., 0.88) will be included in the final alignments, while others will be excluded. The Max-1 method was first used in COMA and COMA++. This method is ideal for granting 1:1 alignments cardinality if required by the user. However, if n is greater than 1, the final alignments can carry up to an n alignment of one entity which can also lead to false-positive mappings, i.e., when correct mappings are eliminated by the system.

2.2.3.3 Max Delta

This method selects the correspondence with the greatest confidence value, and the one with a confidence value that differs by less than delta from the greatest value correspondence, where delta is a pre-defined tolerance value (Do & Rahm 2002). Hence, this method carries issues similar to the threshold filter method in terms of including more than one alignment of one entity in the final results. Therefore, for tasks with only equivalent mappings, this method can result in a high number of false-positive mappings. This method was introduced earlier in COMA and was not included in any later systems to the best of our knowledge.

2.2.3.4 Machine Learning

The idea of using a machine learning approach in this context is based on building a gold standard dataset. The latter consists of pairs of previously aligned ontologies, often built by domain experts, and then using this dataset to build a classifier. The classifier then can be used to identify whether a pair of entities are a match or not based on their similarity score. Different machine learning techniques are used in this domain, for example, Decision Trees, Support Vector Machine (SVM), and Neural Networks.

Ichise (2008) proposed Malfom-SVM, where an SVM approach was introduced to map two ontologies by predicting the similarity of entity pairs based on a machine learning approach. Pairs are classified into two classes: positive and negative. Positive indicates a match, while negative means the entities do not match. The training data when applying such a model in ontology matching is a set of entity pairs classified as correctly or not correctly mapped, based on a specific similarity measure. In general, an SVM model represents data in a two-dimensional field in two categories separated with a wide gap (Nezhadi et al. 2011). Then, a new pair of mapped entities can be tested by the SVM model to predict their category (i.e., correct match, incorrect match) based on their similarity score. POMap++ also adopts a similar strategy to classify candidate pairs without using any reference alignments, i.e., training data are generated automatically by the matcher.

2.2.3.5 Rule-Based

Some OM systems introduce additional rules along with other filters in order to restrict the alignment selection. For example, Falcon-AO, is a system that combines two basic

matchers, an element-level and a structural one. The system uses rules along with a threshold filter to determine final alignments. One of the rules states that if both element-level and structure-level matcher result in high similarity scores, the element-level correspondences will be directly included in the final alignments. However, the structure-level correspondence will only be included if it is higher than a threshold. The second rule emphasizes that if the element-level correspondences are low and the structure-level correspondence is high, then only the structural correspondences are included in the final alignments. Moreover, if the element-level similarity score for a pair is below a very low threshold and their structural similarity is also very low, the pair will be immediately eliminated from the final alignment. Falcon-AO assumes that the element-level matcher is more reliable than the structural matcher. However, no further justification for the determination of high and low measures was provided by the authors.

2.2.3.6 Iterative alignment selection

This automated alignment selection approach was proposed by Gulić et al. (2016). It requires alignments to be organized in a similarity matrix, as depicted in Figure 2.9. The first iteration of this approach is similar to the Max1 method. While the Max1 approach only considers correspondences with the maximum scores, this iterative approach also considers the second to the highest values as long as two conditions are satisfied. On each iteration, the method will add correspondences to the final alignments if they have (i) a similarity score higher than the identified threshold and (ii) an entity that has not been added to the final alignments yet. To illustrate, the first iteration in Figure 2.9 adds all correspondences highlighted in red as they hold the highest similarity values in their columns and rows. Their similarity values are also higher than the allocated threshold, i.e., 0.22. For example, although the pair $\langle C_5, C'_5 \rangle$ has the highest similarity value (0.19), it is eliminated from the final alignment as it is lower than the threshold. The following iteration in the process starts by eliminating all entities that have already been represented in the final alignments from the matrix. For example, after the first iteration, the classes $C_1, C'_1, C_2, C'_2, C_4,$ and C'_4 will be removed from the matrix, while the remaining classes can be further considered. Then, the method continues until the two conditions cannot be satisfied anymore.

	C'_1	C'_2	C'_3	C'_4	C'_5	C'_6
C_1	0.84	0.55	0.16	0.34	0.04	0.12
C_2	0.33	0.94	0.13	0.22	0.12	0.19
C_3	0.40	0.23	0.21	0.12	0.14	0.02
C_4	0.02	0.6	0.07	0.77	0.03	0.04
C_5	0.13	0.04	0.04	0.09	0.19	0.15
C_6	0.0	0.72	0.05	0.02	0.03	0.71

1st iteration:
 Pairs with higher score in both row and column
 and greater than 0.22
 Final alignments $\{(C'_1, C_1), (C'_2, C_2), (C'_4, C_4)\}$

	C'_3	C'_6
C_3	0.21	0.02
C_6	0.05	0.71

2nd iteration:
 Pairs with high score + with a concept not in final
 alignment greater than 0.22
 Final alignments $\{(C'_1, C_1), (C'_2, C_2), (C'_4, C_4), (C'_6, C_6)\}$

Figure 2.9: Example of the iterative final alignment selection approach

2.3 Ontology Matching Evaluation

The Ontology Alignment Evaluation Initiative ⁶ (OAEI) is an international effort devoted to assessing ontology matching systems by organizing an annual campaign for evaluating matching systems. The initiative provides over ten benchmark datasets in different tracks for different matching systems to be evaluated. Examples of main tracks are Anatomy, Conference, Complex matching, large biomedical, and knowledge graphs. Here, we introduce evaluation techniques and benchmarks generally used within OAEI and the semantic web community for ontology matching tasks. It is also worth mentioning that since 2004, OAEI has been the leading evaluation platform for all matching tasks related to the semantic web. Further, all the state-of-the-art matching systems have either participated in OAEI campaigns or have been tested on their benchmark datasets.

2.3.1 Alignment Evaluation

Typical ontology matching systems output an alignment, i.e., a set of correspondences. To evaluate the performance of the system, this alignment is compared to a reference alignment. Correspondences can be classified as follows: (i) *True Positives*: which are the correct correspondences discovered by the system, (ii) *False Positives*: the

⁶<http://oaei.ontologymatching.org/>

incorrect correspondences discovered by the system, and (iii) *False Negatives*: these are the correct alignment that the matching system has missed, or was unable to discover (Djeddi & Khadir 2010). Thus, the quality of an alignment set is calculated by measuring its completeness (*recall*), and correctness (*precision*) according to a reference alignment set (Euzenat et al. 2007). The reference alignment is a set of benchmark mappings between two ontologies.

Precision, recall, and f-measure are also used in OAEI to measure the performance of participating systems. Precision measures the number of the correct correspondences, particularly true positives, over the total number of all the produced correspondences, see Equation 2.1. If a system has low precision, it indicates that the alignment contains a high number of false positives. Therefore, precision is used to assess the correctness of the system alignments.

$$precision = \frac{\text{correct correspondences}}{\text{total number of correspondences}} \quad (2.1)$$

On the other hand, as shown in Equation 2.2, recall is the fraction of the correctly discovered correspondences over the total number of correct correspondences in the reference alignment. The higher the number of false-negative mappings (i.e., undiscovered mappings), the lower the recall of the system. Thus, recall is used to evaluate the completeness of the system (Xue & Pan 2018).

$$recall = \frac{\text{correct correspondences}}{\text{total number of expected correspondences}} \quad (2.2)$$

Finally, f-measure is another measure that is often used in order to combine precision and recall by calculating their harmonic mean (Ochieng & Kyanda 2018), as shown in Equation 2.3. It is a common practice to calculate the f-measure by assuming that both measures are equally important. Nonetheless, in other systems, a weight value w can be added to justify the importance of precision and recall depending on the expert view or the matching task. Examples of systems that apply this method are CroMatcher, and an Evolutionary Algorithm (EA) proposed by Xue & Pan (2018).

$$f - \text{measure} = \frac{2 * (\text{precision} * \text{recall})}{(\text{precision} + \text{recall})} \quad (2.3)$$

2.3.2 Benchmark Datasets

The OAEI campaigns usually consist of different tasks, and each task may have one or more tracks. In recent campaigns, main matching tasks were divided into three categories as follows: schema matching tasks which evaluate systems on the task of matching classes and/or properties, instance matching or link discovery tasks which focus on mapping the individuals of two ontologies, and the third task combines mapping both schema and instance levels. Each category has several specific tracks, each evaluating matching systems on a specific dataset, aiming to solve a particular problem. Moreover, most tracks have different test cases, where each test case has its own datasets with different specifications.

Benchmark datasets are based on OWL or RDF/XML ontology format. The schema matching task has over ten tracks, each has different characteristics to test matching systems. Moreover, the majority of tracks are domain-specific, e.g., Anatomy, Large biomedical ontology, and Biodiversity and Ecology. Another popular track is the ‘Conference’ track, which combines different ontology defining knowledge from the conference organization domain. Each test case has a different reference alignment produced either by domain experts or via crowdsourcing applications (Pour et al. 2021). Thus, participating systems are evaluated for each track using these versions of benchmarks based on precision, recall, and f-measure.

2.4 Knowledge Graphs Matching

2.4.1 Knowledge Graphs on the Web

In recent years, KGs have been gaining increasing attention in the semantic web community. The term Knowledge Graph was invented in 2012 when Google referred to the connected facts used to support their search engine as a knowledge graph (Heist et al. 2020). A KG is a set of interlinked facts extracted to form a large network that can be utilized for various tasks that require knowledge inference (Ehrlinger & Wöß 2016). KGs are often compared to ontologies since both are representing knowledge with semantic relations between different concepts, i.e., classes. Because of their similarities, various definitions of KGs have been proposed in the literature (Ehrlinger & Wöß 2016). According to Paulheim (2017), “A knowledge graph (i) mainly describes real-world entities and their interrelations, organized in a graph, (ii) defines possible

classes and relations of entities in a schema, (iii) allows for potentially interrelating arbitrary entities with each other and (iv) covers various topical domains". This definition provides a list of characteristics that distinguishes KGs from other types of knowledge representations. Different from conventional ontologies, KGs are known for their rich instance level, i.e., A-Box, in comparison to their schema level, i.e., T-Box. Moreover, the schema level in KGs is not as well-annotated as the instance level. Meaning that most of the classes lack long textual descriptions such as comments found in formal ontologies (Paulheim 2017).

Besides proprietary KGs that are often utilized by large companies such as Microsoft, Facebook, and Google, Many common KGs have been created over the last decade. Common KGs such as YAGO (Suchanek et al. 2007), DBpedia (Lehmann et al. 2015), Never-Ending Language Learner or NELL (Carlson et al. 2010) and Wikidata (Erxleben et al. 2014) are multi-domain publicly available KGs that, despite being heterogeneous, can be highly complementary too. Such KGs are used in a wide range of downstream applications such as query answering, search engines, as well as recommendation and reasoning systems (Lecue 2020, Heist et al. 2020). These KGs have resulted from different information extraction practices such as crowdsourcing, and fully or semi-automated approaches. For example, DBpedia is semi-automatically constructed from the structured knowledge embedded in Wikipedia's articles, with a crowdsourcing effort to map its entities. On the other hand, NELL is an automatically constructed KG built by extracting facts from semi-structured/unstructured content on web pages (Carlson et al. 2010).

2.4.2 Current Knowledge Graphs Matchers

In the last decade, the problem of OM has been a well-studied domain in the semantic web community. However, matching KGs have only been a topic of interest since 2018 as a new track for matching KGs was added to the annual OAEI event. Given the nature of KGs having denser data at the instance level, the majority of methods focus on mapping their instances. Nonetheless, recent studies have highlighted that matching the schema of large KGs remains a challenging task that needs further research (Hertling & Paulheim 2020b, Rahm & Peukert 2019). In this section, we review matching systems and methods that focus on KGs. Furthermore, we particularly focus on those that have participated in the OAEI KG tracks. This because OAEI leads research in different matching tasks and methods participating in the KG tracks are

the most relevant to our work.

The first category of systems relies on using element-level matchers, such as utilizing string-based techniques. However, it is important to note that, given the nature of the matching task for KG entities, all matching systems employ different variations of string similarity techniques such as exact string matching, n-gram, and edit distance. For example, AgreementMakerLight or (AML) is an OM system proposed by Faria et al. (2013) as an improved and more flexible version of the state-of-the-art AgreementMaker matcher. AML uses a total of nine matching techniques, including different string-based methods. It has been a consistent participant in many matching tasks, including matching KGs. A recent participant in OAEI is LSmatch (Sharma et al. 2021) which utilizes Levenshtein string similarity as one of its main matching components.

As mentioned earlier, element-level techniques are often combined with structure-level techniques in conventional OM systems. The latter exploit structural data present in typical well-formed ontologies such as disjoint axioms to refine the results of element-level techniques. For example, ATBox (Hertling & Paulheim 2020a) is a system that applies a similar technique in order to fine-tune mappings initially discovered by an element-level matcher. However, while such detailed structural specifications are available in a typical ontology, most large-scale and common KGs lack this level of schematic richness. This is due to them being automatically generated from unstructured data, and the lack of capacity in text mining methods to extract fine-grained relationships.

Using background knowledge resources is another common strategy of many systems to find semantically similar KG entities. For example, AML and LogMap are two leading OM systems that, despite being originally designed to handle biomedical ontologies, can participate in KG matching tracks. Both systems utilize background resources such as WordNet and UMLS lexicons. LSmatch also exploits *thesaurus.com* as a background knowledge resource for its synonym matching component. Besides general thesaurus and domain-specific background resources, the Wiktionary system (Portisch et al. 2019) and ALOD2Vec (Portisch & Paulheim 2018) utilize different types of background knowledge compared to state-of-the-art methods. Wiktionary for instance is named after an online collaboratively made lexical resource known as *Wiktionary*⁷ which the system uses for its synonym matching component. *Wiktionary* is known to have a larger coverage of terms compared to WordNet in addition to having monolingual capacity (Portisch et al. 2019). During the matching process, the system starts

⁷<https://www.wiktionary.org/>

by mapping entity labels to a concept or synonym in *Wiktionary* first, and then align entities whose labels share a link with the same *Wiktionary* concept. ALOD2Vec is another example of using background knowledge for KGs matching. It uses *WebisALOD*, an automatically constructed RDF dataset of hypernym relations (Hertling & Paulheim 2017). The system trains a neural network to represent each concept in the background dataset as a vector using the *word2vec* approach. Next, given two entities, the system starts by finding if their labels exist in the *WebisALOD* dataset. Then, the embedding vectors of the two linked *WebisALOD* concepts are retrieved, and their cosine similarity is calculated and will be considered as the similarity value of their equivalent classes in the source and target KGs.

Another method to match KG entities is by using word embeddings. This method represents each word in a unique fixed-length vector, where semantically similar words are represented closer in a Vector Space Model (VSM) (Raghavan & Wong 1986). A well-established word embedding technique is *word2vec* (Mikolov et al. 2013), a neural-network-based model that is able to learn generic, numerical representations of words based on their occurrences and usage in a large corpus of text. Hertling & Paulheim (2019) introduced DOME, a matching system that exploits larger text descriptions in ontology and KGs. For the class matching task, the matcher generates a vector representation for each class. Then, the cosine similarity between vectors representing all classes is measured in the embedding space. OTMapOnto (An et al. 2021) is another embedding-based system that transfers both source and target ontology into a vector space with the use of a pre-trained word embedding model known as Fast-Text (Bojanowski et al. 2017). The system then proceeds to measure the optimal way to transfer entities from source to target ontologies. Entities are coupled by measuring their *Wassertein* distance, a metric for measuring the optimal distance to transport one vector representation into another (Kolouri et al. 2017). With their increasing popularity in different natural language processing tasks, transform models have been recently introduced to ontology and KG matching tasks. For instance, Transformers for Ontology Matching (TOM) (Kossack et al. 2022) uses a pre-trained sentence-BERT model to map entity pairs along with basic string-based matching methods. Fine-TOM (Knorr & Portisch 2022) is the version of TOM that fine-tune the model according to the datasets being matched.

While all previously discussed systems are either OM systems that are able to match large-scale KGs, some systems were particularly designed to align KGs. One

example is FCAMap-KG (Chang et al. 2019) which is a specific variant of a former OM system FCAMap. The latter was mainly designed to map large and complex biomedical ontologies (Zhao & Zhang 2016). FCAMap-KG uses *Formal Concept Analysis*, a mathematical model for clustering instances and structuring classes, to compare the hierarchical structure of the source and target KGs. After initially aligning entities with an element-level matcher, FCA is mainly used to map KG instances by examining their structural connection with classes and properties across the two KGs. LogMapKG (Jiménez-Ruiz 2020) is another example of a system that targets KGs matching. It is based on the state-of-the-art LogMap with the ability to map instances and properties. However, in later versions of LogMap (Jiménez-Ruiz 2021) the two matcher’s functionalities are combined as LogMap is currently able to map all KGs entities.

2.5 Synthesis of literature

Although many matching systems have been introduced to the field, there are some gaps in the literature with regard to mapping large-scale, semi-automatically constructed, and domain-independent KGs.

First, the majority of the state-of-the-art systems are not scalable enough to handle larger datasets such as KGs. This issue has been raised in many studies (Otero-Cerdeira et al. 2015, Rahm & Peukert 2019). In another survey that investigates large-scale ontology matching systems, Ochieng & Kyanda (2018) argue that even though matching systems have lately shown some progress in terms of scalability, many aspects of mapping large-scale ontologies remain to be investigated. While improvements have been shown in the area of biomedical ontologies with more specialized systems and datasets being developed, research on matching large-scale and cross-domain KGs remains less explored. KGs, particularly publicly available ones, are significantly large-scale due to orders of magnitude larger number of instances. Many downstream tasks are impacted by general-purpose KGs, including query answering, recommendation systems, and semantic search (Obraczka et al. 2021). However, despite the growth of such large-scale KGs, a common concern is the quality and integrity of the data produced by semi/fully automated processes. Thus, considerable efforts have been dedicated to improve KG entity resolution (Papadakis et al. 2020) and completeness, i.e., increasing their coverage (Paulheim 2017). Note that both tasks require mapping and aligning the schema

of different KGs. For these reasons, we argue that matching approaches that target large KGs are imperative to further investigations.

Second, current matching systems largely depend on utilizing string-based and structure-based similarity measures. These techniques usually require well-structured datasets similar to OWL-based ontologies used by state-of-the-art systems. Furthermore, many of those systems are designed to work with formal ontologies as well as XML and relational schemas (Ardjani et al. 2015). According to Zhang et al. (2017), state-of-the-art methods can produce high-quality results for well-structured and well-defined ontology. However, these techniques will not perform as well when applied to less well-structured datasets, such as large-scale, automatically curated, and cross-domain KGs. A major issue with such datasets is that they lack the longer textual descriptions and structural organization found in formal ontologies.

Third, studies in this area are mainly focused on mapping domain-specific ontologies. Although the goal of ontology matching is to solve the semantic heterogeneity problem across ontologies from one domain, current downstream applications, such as reasoning and query answering, require integrating data from different cross-domain KGs. Heist et al. (2020) state that the majority of established links between current public KG entities are made where linking is trivial. These KGs are often built semi-automatically from various resources across the web using different knowledge extraction methods such as data mining and machine learning tools, or collaboratively built through crowdsourcing tools. As a result, they have different characteristics, such as the lack of long and descriptive metadata and structural consistency. Therefore, they bring different challenges and need further studying (Hertling & Paulheim 2020b).

In summary, although the research in the field of ontology matching has proposed a significant number of matching methodologies, the literature shows a lack of studies that target mapping large general-purpose KGs. Such KGs are imperative for a variety of tasks, but carry different characteristics and challenges that are yet to be studied in KG matching. This research will fill this gap by further investigate mapping techniques that aimed to map large, automatically constructed, incomplete, and less well-structured KGs.

2.6 Conclusion

Mapping the schema of large-scale, cross-domain, and automatically constructed KGs is far from a trivial task. With their high level of heterogeneity and large-scale search spaces, it is important for KGs matching systems to balance efficiency and effectiveness. Briefly, in Section 2.2.1 of this chapter, we reviewed current ontology matching techniques utilized by the state-of-the-art systems. Further, Section 2.2.2 and Section 2.2.3 discussed other components used together with basic matching techniques to form complex, end-to-end ontology matching systems. In Section 2.3, we reviewed the process of evaluating ontology matching systems as well as different benchmarks used in the semantic web community. In Section 2.4, we highlighted the difference in matching KGs compared to ontologies. Our aim is to give an overview of current matching systems, particularly those designed to map KGs entities. Finally, Section 2.5 synthesized state-of-the-art and current ontology matching systems. We conclude that current systems (1) still suffer from scalability issues, (2) they are highly reliant on string-based techniques, and (3) they are primarily constructed to map well-formed domain-specific ontologies. While the latter often comes with rich lexical and structure data, the majority of common KGs lack such characteristics. To address these issues, the following chapters will introduce our data-driven approach that targets mapping the schema of KGs with hundreds of classes and millions of instances, capable of handling KGs with unbalanced class populations.

Chapter 3

Creating Gold-Standard Mappings

Contents

3.1	Introduction	47
3.2	Related work	48
3.3	The New NELL-DBpedia Dataset	49
3.3.1	Overview	49
3.3.2	Generating Candidate Pairs	50
3.3.3	Candidate Filtering	51
3.3.4	Dataset Annotation	55
3.4	The Expanded Yago-Wikidata Dataset	57
3.5	Discussion	58
3.6	Conclusion	62

3.1 Introduction

The problem of semantic heterogeneity has been thoroughly studied in the Semantic Web community, with a good deal of ontology matching systems being developed and surveyed. Furthermore, matching systems are annually evaluated through the Ontology Alignment Evaluation Initiative (OAEI). However, as we discussed earlier in Chapter 2, majority of OAEI datasets are domain-dependent and mainly focused on ontology matching tasks. For instance, while a new track for KG matching has been introduced to OAEI’s annual campaign since 2018, the challenges of aligning large-scale KGs remain significant (Hertling & Paulheim 2020b). In terms of the KGs nature and the size of alignment, the current gold standard datasets are not well representative of real-world KGs. Such KGs are known for sharing complementary facts about real-world entities such as people and places, while current OAEI datasets are predominantly domain-dependent. Further, the size of the existing gold standard does not accurately represent the complexity of matching large-scale KGs that imply a significantly larger search space due to orders of magnitude larger number of classes.

In this chapter, we aim to answer our first research question, **RQ1**: *What are the current KG matching benchmarks, and how can we build one that is more representative of the large KG matching problem?* As a result, we propose two gold standard datasets for matching the classes of large, automatically constructed, inadequately structured, and domain-independent KGs. The first introduced benchmark is based on DBpedia and NELL. Both KGs are widely used in semantic web research and can be considered highly influential, but they are yet to be consolidated. Despite that, the majority of LOD cross-domain datasets, including KGs, are interlinked to DBpedia which serves as a central link to many LOD datasets. According to Ringler & Paulheim (2017), NELL is considered as the most complementary KG to other larger KGs such as DBpedia with an average of 10% gain of instances, while merging other large KGs can only lead to a 5% gain. Therefore, we believe they are the best candidates for a gold standard dataset for aligning large cross-domain KGs. The second dataset aligns classes from YAGO and Wikidata. Both are well-known KGs that are integrated into different downstream tasks.

This chapter is organized as follows: Section 3.2 reviews the related work and the current KGs matching benchmarks. Then, we describe the process of building the new proposed dataset of NELL and DBpedia in Section 3.3, while section 3.4, details

the process of expanding the current links between YAGO and Wikidata classes to be adapted for the matching task. In Section 3.5, we analyze the differences between current KG matching benchmarks and those we introduce in this chapter. Finally, Section 3.6 concludes this chapter.

3.2 Related work

Ontology Matching has been a well-studied topic that centers around discovering semantically similar entities across two distinct ontologies (Euzenat et al. 2011). In the last decade, many matching systems have been developed and evaluated annually at the annual OAIE’s events. The initiative provides a variety of matching tracks for matching systems to be evaluated on different tasks. Examples of main tasks are Anatomy, Conference, Complex Matching, and Large Biomedical ontologies. KGs are often compared to ontologies, as both are used for data representation purposes. Different from conventional ontologies, general-purpose KGs are large-scale, multi-domain, and less well-formatted compared to ontologies. However, similar to ontologies, KG entities also suffer from semantic heterogeneity, where the same real-world entities can be described using different terminologies.

While there have been many well-established matching methods for OAIE’s different matching tasks, the need for KG matching techniques remains an open area of research (Hertling & Paulheim 2020b). Research in this domain has only been established since 2018 when OAIE introduced a new track for this particular task. Therefore, participating OM systems have been evaluated on the provided benchmark, and a few dedicated KG systems have participated in the later campaigns. Although matching KGs has been a promising area of research recently, the need for gold-standard datasets that represent diverse KGs remains remarkable.

In 2020, the only benchmark used to evaluate OM systems on the task of matching KGs entities is constructed from the DBkWik project (Hertling & Paulheim 2018). It integrates multiple KGs each created from a wiki-hosting platform. The individual KGs from the DBkWik project were used to create the gold standard datasets for the OAIE KG track. The track consists of five test cases and each test case is aimed at matching classes, properties, and instances of two KGs. While the schema correspondences were built by ontology experts, the instance correspondences were crowdsourced. To the best of our knowledge, this gold standard is the only benchmark available for

the KGs matching task. Nonetheless, the number of mapped classes in this dataset is considerably small, i.e., less than 50 across the five test cases (Pour et al. 2021). Indeed, this number does not give an accurate representation of the complexity of matching real-world KGs where hundreds of classes can be matched.

Multiple domain-independent KGs have been published according to the semantic web standards. Those are either based on Wikipedia, such as YAGO and DBpedia, or semi/fully-automatically constructed, such as WebISALOD and NELL. For example, DBpedia is a knowledge graph constructed from structured data embedded in Wikipedia’s articles (Lehmann et al. 2015). DBpedia also involves crowdsourcing communities to maintain the quality of the mapping between Wikipedia’s articles and the structured knowledge in the KG. In contrast, NELL is an automatically constructed and learned KG under the Never-Ending Language Learner project. The latter uses machine learning tools to infer and extract knowledge from web text to continuously evolve its seed KG (Carlson et al. 2010). Since its launch in 2010, NELL has grown to a KG containing around 50 million facts. While the schemas of the majority of Wikipedia-based KGs cover multiple classes and properties, NELL graph schema is very basic. It does not contain as many relations between instances, i.e., properties (Ringler & Paulheim 2017). WebIsALOD (Hertling & Paulheim 2017) is another KG with a taxonomy structure. It is an LOD dataset version that contains around 400 million hypernym relations extracted from the WebIsA ¹ dataset. This KG is linked to DBpedia and YAGO schemas, however, since it only covers hypernym relations, it does not distinguish classes from instances.

3.3 The New NELL-DBpedia Dataset

3.3.1 Overview

The proposed gold standard is based on DBpedia 2016-10 version ² and NELL 1115th iteration ³. The DBpedia dataset is extracted from its english version as NELL does not support other languages. In terms of DBpedia, we use its SPARQL query endpoint to return all schema and instance information. As for NELL, since a query endpoint is not available, we obtain all its metadata by parsing its most recent dump file, which

¹<http://webdatacommons.org/isadb/>

²<http://downloads.dbpedia.org/wiki-archive/>, visited on 14-2-2020

³<http://rtw.ml.cmu.edu/rtw/resources>, iteration number 1115, visited on 22-2-2020

contains every fact learned by the project so far. As a result, DBpedia has over 750 classes, while NELL has around 290 classes. Assuming that \mathcal{P} is the set of pair-wise classes across the two KGs, then the number of all possible pairs is 218,660. Since the aim is to have pairs manually annotated later, a greedy approach will lead to an excessive number of pairs that is expensive to annotate and likely to be overwhelmed with negative pairs. Therefore, we first apply a **Blocking Strategy** to manually generate a set of candidate pairs \mathcal{C} which is a subset of \mathcal{P} with a significantly reduced number of negative class pairs. Next, we perform a **Candidate Filtering Strategy** by applying two similarity measures to each pair in \mathcal{C} to further reduce the search space for human annotators. After the filtering stage, we perform another screening to ensure that none of the discarded classes had a potential match in the corresponding graph. Finally, for **Dataset Annotation**, we ask human annotators to determine the alignment of the resulting class pairs to construct the gold standard dataset.

3.3.2 Generating Candidate Pairs

Given \mathcal{P} , which is a set of all possible class pairs from the two KGs, we apply a **Blocking Strategy** which includes manually screening the class structure of the two KGs. As a result, we obtain \mathcal{C} , which is a set that is supposed to eliminate as many true negatives as possible while maintaining as many (if not all) true positives. To explain the complexity of this task, the two classes named `School` in both KGs refer to different types of schools. For instance, in the DBpedia class structure, `School` is a subclass of `EducationalInstitutions`, while being a super class of `HighSchool` and `University` in NELL. Given this structural inconsistency issue, a preliminary study aimed at aligning the higher level of concepts across the two KGs was necessary.

We manually created two subsets named \mathcal{A} and \mathcal{B} , where the first is a set of classes in NELL that have a possible corresponding class in DBpedia. Similarly, the second set contains classes from DBpedia in which they have a possible corresponding class in NELL. The process of creating these two sets was done in two phases. First, we start by comparing the common root classes across the two KGs, e.g., `Person` or `Place`. Then, all of their non-root (descendant) classes are added to \mathcal{A} , and \mathcal{B} respectively. For instance, all the descendant classes of $Person_{nelli}$ and $Person_{dbp}$ are added to each of \mathcal{A} and \mathcal{B} respectively. Second, we examine other possible classes in which their root classes do not share an overlap of words, i.e., they are not selected in the first step. A valid example is the pair of the two classes $AcademicSubject_{dbp}$ and its possible

equivalent class *AcademicField_{nell}*. While the former is a subclass of `TopicalConcept`, the second is a subclass of `everypromotedthing`. Note that the latter is the root class of the KG taxonomic structure, i.e., the equivalent of `OWL:Thing` in DBpedia. Therefore, our second screening phase is aimed at all descendant classes in both KGs whose name values share overlapping words while their super classes do not necessarily share overlapping words.

As a result of this blocking strategy, a total of 18,492 candidate pairs were generated in \mathcal{C} as the product of \mathcal{A} and \mathcal{B} . We believe this blocking strategy will not incorrectly discard any true positives because we have examined all discarded classes to identify any possible match in the opposite KG. The number of distinct classes from DBpedia and NELL is 138 and 134 respectively. Nonetheless, over 18,000 candidate pairs remain expensive for an annotation task. Therefore, we proceed by applying a candidate filtering strategy to further reduce the number of pairs that need to be annotated while maintaining the gold standard completeness.

3.3.3 Candidate Filtering

Here, we apply two similarity measures to class pairs generated in the previous phase. We utilize a string-based and an instance-based similarity measure combined with a very low threshold to maximize the chance of retaining all true positives. The **String-based Similarity** measure is applied to class names only, as NELL does not offer other metadata descriptions of classes. However, since the two KGs use different vocabulary, depending on a name similarity only will not assure retaining all true-positive pairs. Therefore, we apply an **Instance-based Similarity** measure to capture any possible true positive pairs, where string similarity could have failed to discover them. To the best of our knowledge, a matching approach that is able to handle a substantial number of instances, such as in the case of KGs, is yet to be established. Thus, in this section, we discuss the implementation of our preliminary instance-based approach used in this process. We believe that combining both measures can ensure a high (if not full) recall of true positive pairs. It is worth mentioning that due to the structural irregularity in both KGs, and the lack of structural data, structural-based similarity measures were excluded.

3.3.3.1 String-based Similarity Measure

We apply the *Levenshtein* (Levenshtein et al. 1966) edit distance approach. This method has shown improvements over alternative string-based measures, particularly for matching classes (Cheatham & Hitzler 2013). Here, the edit distance between class names in each candidate pair is measured. This value is then normalized by dividing the value by the length of the longer string, i.e., class name, to produce a value between $[0.0, 1.0]$. For this task, we only retain a pair if the similarity score of the two class names exceeds 0.4. State-of-the-art matching systems that utilize an edit distance approach often apply a higher threshold, which can be up to 0.8, to eliminate the number of false positive alignments (Anam et al. 2015). Nonetheless, in order to capture as many true positive pairs as possible, we use a threshold that is twice lower than the state-of-the-art methods.

3.3.3.2 Instance-based Similarity Measure

This method casts the matching process based on the principle of free-text index and search, which scales to large-scale datasets. In a typical index/search scenario, a collection of resources (i.e., documents) is indexed in a vector space where documents are represented with weighted vectors of their text content. A weighting approach, such as TF-IDF, is then used to weight term occurrences in documents. A query given to a search engine will also be converted into a vector representation and then matched against all the vectors stored in the generated index. The matching is then done by calculating the cosine similarity value, where a ranked list of top \mathcal{K} documents related to the query is retrieved.

Similarly, we propose to treat both KGs as a collection of documents, where each document corresponds to a class in a KG and each term corresponds to the name of an instance. To map similar classes, a query is built by sampling instance names from a source KG’s class, and matching them against the index of the target KG. The equivalent class is determined based on the search result, which is a ranked list of classes whose instance names overlap with those in the query. We exploit *Apache Solr*⁴, a state-of-the-art tool for free-text indexing and search. The pseudocode for the entire similarity measure is illustrated in Algorithm 1.

⁴<https://lucene.apache.org/solr/>

Algorithm 1 Computing the instance-based similarity using Solr search

Require: $source \leftarrow$ a list of classes in $Source$ KG $target \leftarrow$ a list of classes in $Target$ KG

```

1: for Class  $a_n$  in  $source$  do
2:    $count = 1$ 
3:    $candidate = []$ 
4:   while  $count \leq 30$  do
5:      $query \leftarrow$  a concatenation of 20 instance names of class  $a_n$ 
6:      $results \leftarrow search(query, target)$  in the  $target$  index
7:     for  $b_n$  in  $results$  do
8:        $candidate.append(b_n)$ 
9:     end for
10:     $count++$ 
11:  end while
12:   $candidate\_pairs \leftarrow$  Top three frequent classes in  $candidate$  paired with  $a_n$ 
13: end for

```

During the **indexing** process, an index is created for the source and target KG separately. Classes are represented in documents that contain the concatenation of their instance names. Document contents are indexed using the standard *Solr* indexing process, including tokenization, stemming, lemmatization, lower casing, and term-weighting. For the task at hand, an index is needed for NELL and DBpedia to perform the instance-based matching. Thus, we run the following query to obtain all instance names for each DBpedia class:

```

SELECT ?name
WHERE{ ?entity a <http://dbpedia.org/ontology/%ClassName>.
      ?entity rdfs:label ?name.
      Filter (lang(?name)="en")}

```

After each query, a new document representing that class from DBpedia is created and indexed in the designated DBpedia index collection. Similarly, an index is created for NELL, which contains indexed documents of instance names parsed from the NELL dump file mentioned earlier.

In terms of the actual matching process, NELL and DBpedia were treated as source and target, respectively. Consequently, queries are generated by sampling instance names from NELL’s classes. This process can be performed in the opposite direction; however, some of DBpedia’s classes have missing instances. This implies that a query cannot be created from such empty classes. For example, classes such as **State**, **Zoo**, **Profession** are all leaf classes and are supposed to be populated with individuals but the links between the class name and its instances are missing in the KG. A case in point is **California**⁵ and **Florida**⁶: both are defined in the KG with classes (i.e., `rdf:type`) other than **State**. This problem was encountered in 20 classes from the 138 classes selected from DBpedia. With DBpedia being the center of the LOD datasets in mind, many options can be explored in order to fulfill this gap. Other options include using instances from *SKOS* concepts or another KG that already has an established mapping with DBpedia, such as Wikidata or WebIsALOD (Hertling & Paulheim 2017). Nonetheless, we believe that performing a one-way search is sufficient for capturing all positive pairs for the annotation task.

In terms of the **Search** process, we aim to discover class pairs that share a significant number of overlapping instance names across two KGs. Our empirical test on a smaller sample of the dataset showed that two key factors can highly impact the search (matching) result. The first one is the number of instance names to be used in the query string. Using either a too-large or too-small number of instance names to generate a query can result in zero similar documents (classes) being retrieved or false positive results. This is due to these KG instances being automatically extracted, and to a large number of instances per class in typical KGs. The second factor impacting the search result is the number of searches (iterations) performed on each class to determine its equivalent class. Because of the restriction of the query length, concatenating the names of all class instances is not feasible. Moreover, by using a sample of instance names, different results can be retrieved depending on the sample. Our experiment has shown that we can obtain the maximum number of true positive pairs when concatenating 20 instances per query and performing 30 iterations per class.

To demonstrate, for a class a_n in NELL, 20 random instances are obtained and concatenated to form a query string. That query is then matched against all documents (classes) in the target index, i.e., DBpedia. Consequently, a list of classes whose instances overlap with those in the query is retrieved. For example, if the following

⁵<http://dbpedia.org/page/California>

⁶<http://dbpedia.org/page/Florida>

results were retrieved when sampling instances from class $Airport_{nell}$ in the source KG:

$$\begin{aligned} \text{Iteration}_1 &\rightarrow \{Airport_{dbp}, City_{dbp}, Port_{dbp}\} \\ \text{Iteration}_2 &\rightarrow \{City_{dbp}, Port_{dbp}, Airport_{dbp}\} \\ \text{Iteration}_3 &\rightarrow \{Airport_{dbp}, Port_{dbp}\} \\ \text{Iteration}_{n-1} &\rightarrow \{Airport_{dbp}, City_{dbp}\} \\ \text{Iteration}_n &\rightarrow \{Airport_{dbp}, City_{dbp}, Street_{dbp}\} \end{aligned}$$

By the end of the 30th iteration, we added three pairs of candidate alignments for class $Airport_{nell}$. Only the three most frequently retrieved classes among all iterations are added as positive pairs, with a non-zero similarity score. For the above example, the following pairs will be added: $(Airport_{nell}, Airport_{dbp})$, $(Airport_{nell}, City_{dbp})$, and $(Airport_{nell}, Port_{dbp})$. Notice that $Airport_{nell}$ is not matched to, $Street_{dbp}$ as the latter only appeared once during the search process.

3.3.3.3 Combining Similarity Measures

Our goal for this particular task is to discover all potentially true positives to be annotated by humans later. Thus, we aim to ensure a high (if not full) recall, which can be achieved by combining the two similarity measures. We applied the above-mentioned similarity measures to the 18,492 class pairs obtained in the prior phase. Only pairs that obtained a similarity score higher than 0.4 by the string-based measure, or a non-zero value by the instance-based measure, were considered for the annotation task. Following the automated approach explained above, we performed another manual screening to discover the remaining equivalent classes from NELL and DBpedia that were not included in the potential pairs. By inspecting all pairs discarded by the filtering process, we were able to identify and recover 8 pairs. As a result, a total of 596 pairs were created for the human annotation task.

3.3.4 Dataset Annotation

In order to create a gold-standard dataset for matching KG classes, we ask human annotators to determine the alignment for the previously discovered pairs. We then aggregate their interpretations by the majority votes, as human annotators can have

different interpretations of correspondence classes. We also perform a study of the inter-annotator agreement (IAA). The participants were provided with guiding instructions to complete the task. Several labels were allowed to annotate pairs which are a **match**, **not a match**, **more general**, and **more specific**. The latter two options are often used in the ontology domain to label subsumption or complex relation between entities. The reason we gave the annotators this option is that it can be possible in a few cases. For example, while DBpedia has two separate classes for **State** and **Province**, NELL has one class named **StateOrProvince** which combines both. For each pair, participants were given class names, URIs, and a sample of instance names. While in the majority of cases annotators can make their decision based on the class names, in some occasions more context is necessary. Therefore, participants were advised to use the provided URIs to read more about each class context when a name does not give a clear indication of its context. For example, NELL has classes named **MusicArtist**⁷ and **Musician**⁸ which can be ambiguous to some extent. In similar cases, the description provided in NELL’s knowledge browser or DBpedia can be useful. The dataset was annotated by twenty research students and validated by two computer scientists. Each participant annotated around 50 pairs on average. In order to observe (IAA), 400 random pairs are duplicated among 12 annotators, such that each pair is annotated by three different annotators. The average IAA for this task was measured using Cohen’s kappa (Banerjee et al. 1999) based on a sample of the dataset, and it was 0.83.

Two experts then validated the dataset to ensure that the subsumption relations were used properly. Therefore, a subsumption relation was only added to the dataset if there was an agreement by the experts. The gold standard mapping resulting from this annotation task is publicly available as two test cases⁹. The small test case includes a few instances per class, while the full test case contains the full A-box information for the included classes. The latter can be used to benchmark instance-based matching systems. The size of the gold standard is 129 equivalent class pairs with 24 non-trivial matches, i.e., not an exact matching string of class labels. Currently, the larger dataset in OAEI’s KG track carries only 15 class matches, while the maximum number of non-trivial matches is 10. This makes the proposed dataset the largest domain-independent gold standard for matching KG classes. Finally, this gold standard is considered as a

⁷<http://rtw.ml.cmu.edu/rtw/kbbrowser/pred:musicartist>

⁸<http://rtw.ml.cmu.edu/rtw/kbbrowser/pred:musician>

⁹https://github.com/OmaimaFallatah/KG_GoldeStandard

partial gold standard since some classes in both KGs have no equivalent class in the corresponding KG.

3.4 The Expanded Yago-Wikidata Dataset

Contrary to the NELL-DBpedia dataset which we constructed, this dataset is built to expand on pre-existing links between schema.org and Wikidata schema, which were originally created by Krauss (2017). Different from the original dataset, which only includes the schema alignments, we refer to this dataset as YAGO-Wikidata because we retrieved the instances of Schema.org classes from YAGO ¹⁰. YAGO is another open-source KG that is automatically generated from facts shared on Wikipedia. It is considered as one of the largest KGs as its recent version YAGO4 (Pellissier Tanon et al. 2020) contains around 2 billion facts and 64 million entities. Wikidata on the other hand is a KG that is generated via collaborative editing that currently has around 1 billion facts ¹¹.

The original gold standard mappings include over 500 links, however, not all of them represent equivalence relations. Further, it includes links that map the schema level of both KGs by including class and property alignments. For the purpose of our task, we filter the links to keep only those that align classes. So as a first step, we only include mappings annotated with the relationship `equivClass`. However, we observed that it does not guarantee an equivalence relation between the two classes. For example, The class named `BarOrPub` in YAGO has two `equivClass` links with two classes in Wikidata which are `Bar` and `Pub`, and this indicates a one-to-many alignment. Those alignments were also removed, given that the majority of studies on mapping KGs only consider one-to-one mapping. As a result, the new dataset contains 304 equivalent class pairs.

Since the original dataset does not include any instances, we needed to retrieve instances data for all the mapped classes. In terms of Wikidata, since its entities are represented by their Q indices, e.g., Q1234, we use the Wikidata python API to query their URIs in order to retrieve their labels. The following SPARQL query was used in order to retrieve Wikidata labels:

```
SELECT *  
WHERE {
```

¹⁰YAGO 4 <https://yago-knowledge.org/downloads/yago-4>

¹¹https://www.wikidata.org/wiki/Wikidata:Main_Page

```

    %Qindex rdfs:label ?label .
    FILTER (langMatches( lang(?label), "EN" ) )
}

```

The same API is then used to retrieve instances of each class using the following query, where P31 is the Q index of the property that represents `instanceOf` relation in Wikidata.

```

SELECT ?label
WHERE {
    ?name wdt:P31 wd:%ClassName .
    ?name rdfs:label ?label .
    FILTER (langMatches( lang(?label), "EN" ) )
}

```

However, some classes resulted in an error indicating that the Q index was no longer available. This is due to the original dataset being old and the fact that Wikidata is constantly edited with collaborative efforts. In that case, we used the class label to retrieve its current URI, which often contains the Q index. Then, we use the retrieved data to generate a subgraph of Wikidata that includes all 304 classes and their annotated instances. Similarly, we use YAGO's SPARQL query endpoint to retrieve all schema and instances metadata related to the 304 classes included in the original mapping dataset.

```

SELECT ?name
WHERE {
    ?entity a <http://schema.org/%s> .
    ?entity rdfs:label ?name.
    Filter (lang(?name)="en")
}

```

The new expanded dataset is made publicly available at ¹². This includes the two subgraphs in RDF/XML format and the alignments file according to OAEI's standards. With 304 class alignments, including 92 non-trivial links.

3.5 Discussion

Here, we discuss the importance of creating benchmarks for matching the schema of large KGs that resemble real-world KG characteristics. Current benchmarks do not

¹²<https://github.com/OmaimaFallatah/YagoWikidata>

Table 3.1: The number of classes and instances in the two proposed datasets

Dataset	#Classes	#Instances	Avg # instance
DBpedia	138	631,461	4,576
NELL	134	1,184,377	8,905
YAGO	304	5,149,594	33,691
Wikidata	304	2,158,547	12,576

Table 3.2: Statistics of the number of classes and instances in the eight KGs from the OAEI Knowledge graph track

Knowledge graph	#classes	#instances	Avg #instance
Star Wars	269	145,033	539
The Old Republic	101	4,180	41
Star Wars Galaxies	67	9,634	144
Marvel Database	186	210,996	1,134
Marvel Cinematic Universe	55	17,187	312
Memory Alpha	181	45,828	253
Star Trek Expanded Universe	283	13,426	47
Memory Beta	240	51,323	214

resemble the challenging task of mapping cross-domain KGs in terms of size, imbalance class distribution, topics, and matching focus.

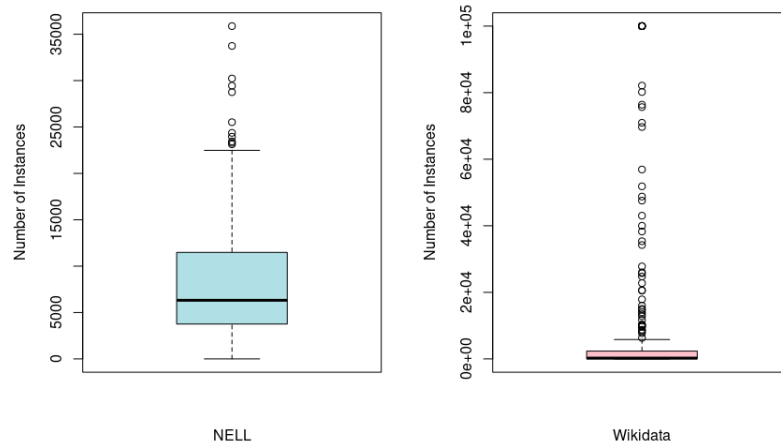
Regarding the *dataset size*, common KGs often have hundreds of classes and millions of instances. Table 3.2 details the number of classes, instances, and the average number of instances in the current OAEI KGs. This dataset consolidates 8 KGs from the DBkWik project (Hertling & Paulheim 2018), forming 5 benchmarks used to evaluate matching systems on mapping classes, properties, and instances. The two largest KGs in terms of the number of classes are the **Star Wars** and **Star Trek Expanded Universe**, which have 269 and 283 classes, respectively. However, both KGs have a relatively small number of instances compared to their class counts. Furthermore, DBkWik KGs do not represent common KGs in terms of the size of instances data. For example, the **Marvel Database** is the largest out of the eight OAEI KGs in terms of the number of instances, with a little over 200,000 instances. This number is far from the reality of large-scale and domain-independent KGs shared on the web as depicted in Table 3.1.

Another aspect that distinguishes common KGs from the OAEI KGs is the ***distribution of instances*** across classes. Figure 3.1 illustrates the differences between the *MemoryAlpha-MemoryBeta* class distribution compared to two common KGs (i.e., NELL and Wikidata). We can notice that while having an imbalanced class distribution is common in both datasets, the smaller number of instances in OAEI KGs has also impacted the distribution. In both KGs in Figure 4.1b, more classes have relatively small and comparable numbers of instances with one outlier class, while common KGs have far more outliers in Figure 4.1a. Further, common KGs have a high variance in terms of the number of instances. Also, the classes that fall into the middle 50% are significantly larger than those in the OAEI KGs. This also shows that with common KGs, in addition to the imbalanced classes issue, they also carry enormous instances with possible data redundancy which may impact some matching methods.

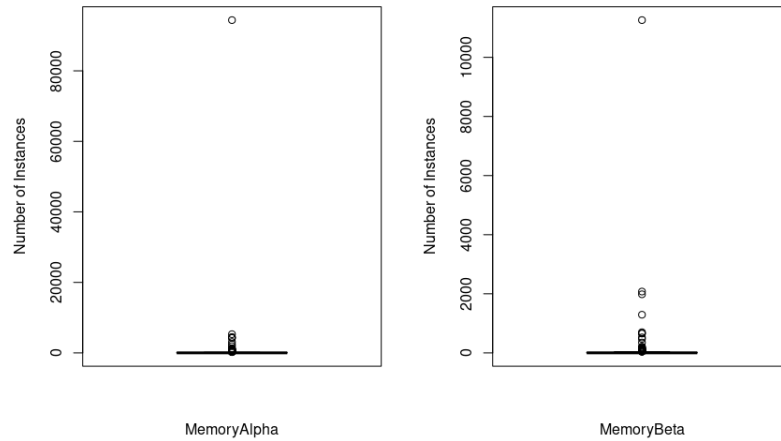
The ***range of topics*** covered by OAEI KGs is also very different from real-world matching tasks. Common KGs are often cross-domain and cover data about real-world entities such as places, organizations, and events. However, this is not the case with the current OAEI KGs that are based on the entertainment domain. The latter was generated from wiki platforms where the top five topics covered are in the domains of games, lifestyle, comic books, and movies (Hertling & Paulheim 2018).

Another limitation of the current benchmarks is that they are more ***focused on mapping KG instances***. To be more specific, the KG track has 15,129 instance-level alignments across the five benchmarks, compared to less than 50 for class-level alignments. Despite that the majority of these KGs include a significantly large number of classes, the total number of class alignments across the 5 benchmarks is 49. In fact, the largest benchmark in the current dataset only includes 15 class alignments (Hertling & Paulheim 2020b), which indeed does not well represent the problem of matching large cross-domain KGs with hundreds of complimentary classes.

The two benchmarks we propose earlier accurately represent the challenge of matching large-scale KGs as both of them contain mappings between hundreds of classes. Moreover, they are generated from common and domain-independent KGs with high impact on the field. Both benchmarks have classes with imbalanced distribution as the case in many common KGs. This issue is typical due to the (semi)automated nature of the way such large KGs are created, resulting in inconsistent class/property completeness and coverage (Ramadhana et al. 2020). For instance, while some KGs are generated by crowd editing efforts such as Wikidata, NELL is extracted by continu-



(a) Common Knowledge Graphs



(b) OAEI Knowledge Graphs

Figure 3.1: The class distribution of two KGs from the two proposed datasets (NELL and Wikidata), compared to the distribution of instances in two of the largest OAEI KGs (MemoryAlpha and MemoryBeta)

ous ‘machine reading’ of web-based free texts (Heist et al. 2020). Previous work has discussed its affect on different tasks relying on large KGs, such as biased results on recommendation systems (Fu et al. 2020), and KG completion (Ji et al. 2016). In the following chapters we discuss how this issue can impact the results of instance-based matching methods as well.

3.6 Conclusion

In this chapter, we proposed two gold standard benchmarks for matching the classes of large KGs. The datasets are based on four highly influential KGs, and one of them is yet to be linked to the LOD, i.e., NELL. In Section 3.2, we highlighted the lack of datasets that represent real-world KGs which are often large-scale, domain-independent, and less well-structured compared to current KG datasets. Section 3.3 details our approach to building a gold standard mapping between NELL and DBpedia which are two well-acknowledged common KGs that are yet to be linked. In Section 3.4, we expanded the current links between YAGO and Wikidata classes to generate a second dataset for common KGs matching. The two datasets proposed in this chapter are considered the largest domain-independent benchmarks for the KG schema matching task. Both datasets are currently used as evaluation benchmarks for OAEI’s new KG matching track named the *Common Knowledge Graphs* track¹³. The track has been introduced as of the 2021 campaign, where 9 out of 17 systems were able to participate. Finally, as discussed in Section 3.5, with proposing the two datasets, we argue that matching large, domain-independent and automatically constructed KGs has significant utility and therefore, future work should be devoted further to this area.

Related Publication

- Fallatah, Omaima., Zhang, Ziqi., & Hopfgartner, Frank. A gold standard dataset for large knowledge graphs matching. *In Ontology Matching 2020: Proceedings of the 15th International Workshop on Ontology Matching co-located with the 19th International Semantic Web Conference (ISWC 2020)* (Vol. 2788, pp. 24-35). CEUR Workshop Proceedings.

¹³<https://oaei.ontologymatching.org/2022/commonKG/index.html>

Chapter 4

Training Knowledge Graph Classifiers

Contents

4.1	Introduction	64
4.2	Related Work	65
4.2.1	Entity Classification for Ontology Population	65
4.2.2	Imbalance in Text Classification	67
4.3	KG Instances Classification	70
4.3.1	KG Instances Resampling	70
4.3.2	Building the KG Instance Classifier	74
4.4	Experiments	76
4.4.1	Experiment Settings	76
4.4.2	Evaluation Metrics	77
4.4.3	Results on Common KG Datasets	78
4.4.4	Results on the DBkWik dataset	81
4.4.5	Results on the Web Data Commons dataset	84
4.5	Conclusion	88

4.1 Introduction

In the previous chapter, we have introduced two benchmarks for mapping common KGs that brings more challenges to the matching task. Combining different matching techniques for ontology and KG matching is a common approach. Hence, the majority of state-of-the-art matching systems depend on a set of pre-selected techniques. For example, *Element-level* matchers discover similar entities by utilizing the textual annotations defined in the ontology’s entities, e.g., URIs, labels, and comments. *Structural-level* matchers are also utilized, to exploit structural information like disjoint axioms to refine element-level alignments. Nonetheless, while current tools are able to produce high-quality results for well-formed ontologies, such techniques are not as well-performing when applied to KGs that lack longer textual descriptions. Further, while some ontology matching systems utilize structural knowledge available in well-structured ontologies, they can be difficult to apply in the case of automatically constructed KGs lacking thorough schematic information.

There have been a smaller number of instance-based or *Extensional* methods compared to terminological and structural matching methods. This is due to conventional ontologies sharing a significant amount of terminological similarities, such as prefixes and suffixes. In contrast, KGs, particularly those shared on the web, are unique and annotate numerous instances at the instance level compared to their schema level. Further, they are often domain-independent and cover data about real-world entities described with different terminologies. This makes the *instance-based* matcher more suitable for matching KG classes. The conventional way of utilizing such a method implies measuring the overlap of instances across classes. However, it is more challenging to measure the extension of such an overlap given the large number of instances in cross-domain and highly imbalanced KGs as shown in Chapter 3.

This chapter aims to answer our second research question, **RQ2**: *Given the large yet unbalanced number of instances in KG classes, how can we make use of them effectively in learning?* Here, we propose a novel method of using a KG instances to train a model that classifies instances into classes in a KG, and we refer to this as *KG instance classifier*. The trained classifiers are used as a key component of our proposed instance-based approach that we will discuss in the following chapter. Here in this chapter, we focus on introducing our method of self-training this KG classifier using instances and classes from a given KG. Moreover, we present an in-depth analysis of

the class imbalance problem found common in large KGs, and describe our approach to handling this problem in the context of KG matching. Nevertheless, our method is highly generic, and we expect it to be useful in scenarios of KG population that are key for other KG tasks such as KGs completion and KGs entity refinement.

Section 4.2 discusses some current and related studies. Then, in Section 4.3, we detail our approach of training KG classifiers. Finally, we examine the ability of the trained classifiers to classify instances from distinct datasets in Section 4.4.

4.2 Related Work

We discuss related work in two areas: those that utilize entity classification for ontology population and those that study methods for handling imbalanced data in text classification.

4.2.1 Entity Classification for Ontology Population

Ontology Population (OP) is defined as the task of introducing new entities to an ontology, this includes adding instances, properties, or classes (Faria et al. 2014). New entities can be identified manually either by domain experts or human annotators, which can be both a time-consuming and complex task. Instead, new ontology entities can be also discovered automatically from a text corpus. For example, adding new instances to the ontology by utilizing fine-grained entity typing tools that extract entities from a large corpus of text. Several approaches are used in the literature for ontology population: rule-based, statistical approaches, and machine learning methods (Lubani et al. 2019). In this section, we focus on state-of-the-art and recent entity classification approaches used in OP, as they are the most relevant to our work.

Some earlier machine learning-based methods for this task extract new entities from unstructured or semi-structured text. This is done by using methods from the domain of Named Entity Recognition (NER) (Mohit 2014) in order to determine instances relevant to each class in the target ontology. However, evaluating such methods requires a huge amount of labelled data, and often requires human validation. For instance, FIGER (Ling & Weld 2012) trains a multi-label classifier for entity labeling. The dataset used in this work consists of sentences from Wikipedia labeled based on 112 types (i.e., classes) extracted from Freebase (Bollacker et al. 2008). Classifiers are trained using lexical features such as tokens, part-of-speech tags, and word depen-

dency and patterns. Similarly, HYENA (Yosef et al. 2012) also trains a multi-label hierarchical classifier, however, entities here can be classified into multi-level classes, i.e., a superclass and a subclass. This approach also classifies entities based on types extracted from YAGO utilizing lexical features like sentence surrounding mentions, the index of the paragraph with a mention, and part-of-speech tags.

With the increasing popularity of word and KG embeddings (Dai et al. 2020), more work has been done using such resources for OP tasks. Yogatama et al. (2015) and Ren et al. (2016) introduce embeddings-based approaches that represent entity types, text features, and entity mentions in a low-dimensional space where semantically similar features and types have similar representations. Similar to earlier work in (Ling & Weld 2012), both methods generate training data using types from Freebase. However, incorporating embeddings allows for less noisy results as compared to lexical features, word embeddings are more efficient at capturing the meanings of words considering their context. In addition to utilizing fine-grained entity recognition models to populate an ontology with additional instances, embedding methods are also utilized to populate ontology schemas. For instance, Zafar et al. (2016) introduced an approach that induces the taxonomy of an ontology using word embeddings. Given a benchmark consisting of a corpus from a specific domain, and a gold-standard taxonomy, the goal of this method is to generate a taxonomy that is close to the gold standard using the input corpus only. The method utilizes the word2vec model (Mikolov et al. 2013) to represent words in a low-dimensional model in order to expand the taxonomy using hypernym relations. Similarly, Ristoski et al. (2017) proposed TIEmb a word-embeddings-based approach to expand the class subsumption (*is-a*) taxonomy (e.g., a **Basketball Player** *is-a* **Athlete**). This method also utilizes vector space embeddings to represent classes of such a relation. In such a model, instances of the more specific class (**Basketball Player**), will be positioned inside the large cluster that represents the more general class (**Athlete**). Therefore, given a knowledge base such as DBpedia and WebIsADB (Hertling & Paulheim 2017), the method represents instances using word embeddings where each class will naturally generate a cluster. Then, the class subsumption relations are extracted from the formed clusters to be added to the taxonomy of the knowledge base.

Some recent work have explored using deep learning models for ontology population. Clarkson et al. (2018) introduced an approach that aligns instances that are manually defined by a user to an ontology schema. It trains a hierarchical classification model

using instances from the target ontology, i.e., the ontology to be populated. Classifiers are trained using word embeddings trained using a domain-specific text corpus of patient reports related to drug reactions. The trained classifier is then applied to instances that were previously annotated by users, and the classifier then determines the classes an instance can be mapped to. The user’s validation is required to populate the class based on the computed probabilities. The approach we introduce here is different from (Clarkson et al. 2018) first in terms of the size of the dataset – their target ontology contains less than 100,000 instances. Second, their work does not discuss any issues related to class imbalance, as their dataset was manually created. Further, due to the nature of their task being based on the medical domain, their approach requires consistent user involvement, i.e., to label the data and verify the classification outcomes. More recently, Seo et al. (2021) proposed a KG schema expansion that trains a deep learning model using a large text corpus, without any user annotation. The model is trained using entity mentions and word position as features, and it aims to generate new classes and properties. Their model also incorporates active learning, as the user’s feedback is required to decide whether a certain class should be added to the target KG schema. The dataset used to evaluate this method is also based on Freebase types, similar to other state-of-the-art OP tools.

4.2.2 Imbalance in Text Classification

Imbalanced distribution in datasets can contribute to many problems in supervised machine learning. Typically, machine learning models tend to be biased towards majority classes, i.e., highly represented classes. This is due to algorithms being built to improve the learning accuracy by minimizing errors where possible (Laadhar et al. 2019). Learning from highly imbalanced datasets has been a thoroughly studied topic, where many solutions have been developed (Padurariu & Breaban 2019, Sahare & Gupta 2012). Solutions can be divided into two different categories. The first category operates on the dataset itself by aiming to re-sample the data prior to training the models. Without altering the training data, the second category of solutions aims to introduce changes in classification algorithms themselves (Liu et al. 2020). In the following, we provide an overview of the state-of-the-art and common data-balancing methods used for classification-like tasks.

In terms of re-sampling methods, they are used to balance the class distribution by either using undersampling or oversampling techniques. Undersampling is performed

mainly on the majority class(es). A common way to perform such a task is by randomly eliminating samples from such classes to reduce their size to match the minority class(es). On the other hand, oversampling operates on the minority classes by generating more positive samples to match the size of the majority class. This can either be done by randomly duplicating samples in the minority class(es) or by using algorithms that generate syntactic data (Mascardi et al. 2009). A random approach would likely lead to a loss of important data in undersampling or generate noisy data samples in oversampling. For this reason, more complicated approaches have been introduced in the literature.

The Edited Nearest Neighbor (ENN) (Fayed & Atiya 2009) uses the k-means method to remove instances with more neighbors from a different class. The One-Sided Selection (Kubat et al. 1997) removes instances close to the decision border. Tomek Links targets the majority classes by removing all overlapping instances to ensure that instances from different classes are fairly distanced (Tomek 1976). Similarly, the Synthetic Minority Over-sampling Technique or SMOTE (Chawla et al. 2002) is the most common oversampling method applied in the literature to handle imbalanced data. It randomly oversamples the minority classes by generating syntactic data for each minority class. The algorithm uses the K -nearest neighbors to current samples in a minority class to introduce new synthetic samples from neighboring samples. Different variations of SMOTE have also emerged later, such as Borderline-SMOTE (Han et al. 2005), Safe-Level SMOTE (Bunghumpornpat et al. 2009) and Adaptive Synthetic Sampling Approach (ADASYN) (He et al. 2008). However, all of these methods are based on the K-Nearest Neighboring (KNN) algorithm (Altman 1992), which has some limitations. Liu et al. (2020) state that this algorithm can be easily affected by large and noisy data. Further, the cost of running this algorithm significantly grows on large-scale datasets.

Lu et al. (2017) study different sampling techniques on 18 datasets, where undersampling methods generated better results compared to oversampling in a task involving imbalanced binary classification data. In a more recent study, Padurariu & Breaban (2019) compared different data balancing strategies on the task of classifying short phrases of text from 17 classes in the financial sector. Their work included studying different forms of representing data, e.g., TF/IDF, Bag of words, and word embeddings, along with various sampling algorithms. They have concluded that oversampling techniques including random oversampling have shown an improvement in

the results of most classifiers. Similarly, Glazkova (2020) studied the impact of different SMOTE variations on the task of topic classification of biographical data extracted from the Russian version of Wikipedia. The results have shown an improvement using oversampling techniques with KNN and SVM classifiers compared to neural networks.

A notable limitation in these studies is that they either focused on binary classification or on rather small datasets. Khurana & Verma (2022) introduced a method that overcomes the limitations of current oversampling techniques that generate syntactic data, however, the dataset size and the distribution of classes are far from the KG datasets' size. Moreover, the largest class in the dataset used in (Padurariu & Breaban 2019), has 1329 instances while the smallest has 22. It can be intuitive to apply sampling techniques on binary classification tasks, as it can be easy to determine the majority and minority classes. However, the larger the dataset, e.g., KGs, the more complicated the balancing process can be. This is because when multiple classes are largely imbalanced, improving the performance of a certain class can potentially worsen the performance of another (Krawczyk 2016).

The other category of data balancing strategies works on the machine learning algorithm instead of the data (Elkan 2001, Liu & Zhou 2006). This is often done by adjusting the weights of classes to better distinguish samples from minority classes. Such methods assign higher weights to the minority classes and lower weights to the majority classes in order to improve the classifier bias toward large classes. In addition to these methods requiring inputs from domain experts to study the dataset and assign specific class weights, they are known to have little to no impact on batch-training models such as neural networks (Liu et al. 2020). This is because the weights are only applied to the validation data, while the issue with imbalanced training data remains.

Although many solutions for dataset imbalance have been introduced in the literature, we identify that there is no research on addressing the class imbalance issue in KG matching or even ontology matching in general. Previous research in the context of classification tasks has shown that there is often no one-size-fits-all method, and thus previous findings may not generalize to this task. Combined with the increasing research in KG matching and the popularity of instance-based techniques, we argue that it is imperative to further investigate, and develop methods to address the issue of imbalanced distribution in the KG matching task.

4.3 KG Instances Classification

Given the nature of the KG hierarchy which organizes instances based on classes, we propose to follow a *self-supervised* machine learning approach, which is a form of supervised learning that automatically generates training samples from the dataset itself (Hu et al. 2014). Here, classifiers can be trained using instances as data points (samples) and classes as their classification labels. Thus, given a KG (e.g., \mathcal{KG}) with a set of classes $\mathcal{KG} = \{C_0, C_1, C_2, C_3, \dots, C_n\}$, our aim is to train a multi-class classifier, e.g., $CL_{\mathcal{KG}}$. Particularly, it is a KG instance classifier trained using \mathcal{KG} instance data where each class has a set of instances, $\mathcal{C}_n = \{i_0, i_1, i_2, \dots, i_n\}$. Thus, given a new, unseen instance name, $CL_{\mathcal{KG}}$ will be able to classify it into one of \mathcal{KG} classes, e.g., \mathcal{C}_n .

In a supervised text classification task, a classifier is trained with labelled data such as positive and negative emotions in a simple sentiment analysis task. Then, the model can be tested with new unseen data and predict its appropriate class. As for any machine learning task, we divide the KG dataset into two parts. First, a training dataset is to be used during the process of building and training the classifiers. Second, a testing dataset is used to evaluate the performance of the classification and feature extraction processes. Here, we divide instances data into 85% for training and 15% for the purpose of testing the classifiers.

4.3.1 KG Instances Resampling

As we have discussed so far, large KG classes are highly imbalanced. This implies that our KG classifier training needs to handle the class imbalance problem. To illustrate, Figure 4.1 depicts the distribution of instances in the four KGs related to our datasets presented earlier in Chapter 3. In the following, we introduce six different sampling strategies that we implement in order to re sample KG instances. Since we are dealing with imbalanced data, we can resort to popular methods used for dealing with imbalanced training data in classification. As the goal of the data balancing technique is to decrease the bias of classifiers towards majority classes at the expense of the minority classes, we need to define both in the context of KGs. However, as discussed earlier in Section 4.2, that there is a lack of consensus on how majority/minority classes are defined in multi-classification settings particularly with this large number of classes. Here, we adopt an approach where we first calculate the average number of instances per class within a KG, and then classes with fewer instances than this number are

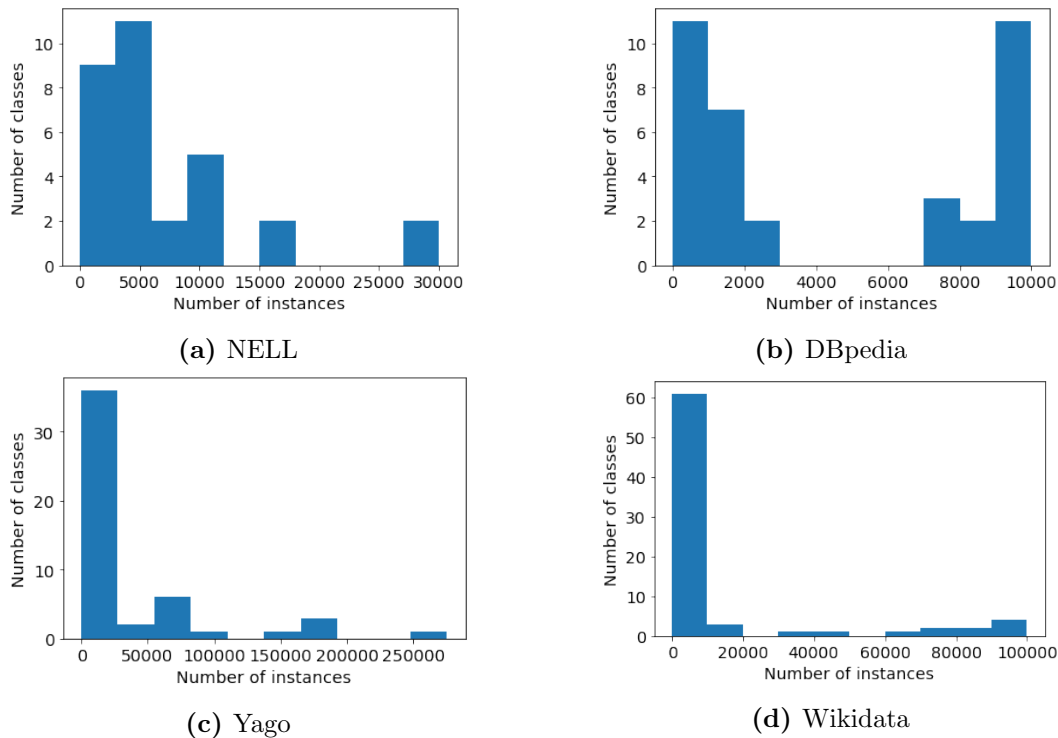


Figure 4.1: The class distribution of the four KGs in the two used datasets after applying the exact name filter.

treated as minority classes and those above it as majority classes. While we acknowledge that this threshold can be automatically decided in the future, we believe that this threshold is considered satisfactory for the task in hand. Later in this chapter, we will demonstrate that the identified threshold strikes a balance by not overly excluding or including an excessive number of samples from the minority or majority classes.

4.3.1.1 Random Undersampling

In binary classification, random undersampling randomly excludes data samples from the majority class to match the size of the minority class. This is a common strategy seen in the literature (Liu et al. 2008). Similarly, in multi-class classification tasks, this method is independently applied to each class by randomly sampling an equal sample size of all classes. The sample size matches the size of the class with the least data samples (Lemaître et al. 2017).

4.3.1.2 TF-IDF Undersampling

As opposed to random undersampling, which randomly discards instances from the majority classes and can result in losing potentially useful samples, this method uses TF-IDF (Ramos et al. 2003) to measure the ‘importance’ of samples and select them based on this score. As a result of being fully/semi-automatically created, common KGs often suffer from noisy and redundant information in their instances (Hertling & Paulheim 2018). Thus, by introducing data resampling, we aim to balance KGs instance population by undersampling the majority classes. Our goal is to obtain a smaller yet indicative instance sample in order to limit the effect of the imbalanced class distribution on the training/learning process. We chose to apply TF-IDF as it is a commonly used measure to evaluate the relevance of words to a collection of documents by weighting their occurrences. Similar to applying TF-IDF for information retrieval tasks, we treat each KG class as a ‘document’ while the concatenation of the labels of its instances is treated as its content. Hence, the TF/IDF value of a token represents its relevance to a particular class in a KG compared to other classes. The TF/IDF scores of all the tokens extracted from each majority class are calculated and ranked. The highest ranked k words per class are then used to undersample instances in the majority classes by discarding instance names that do not contain any of these words. To illustrate, assuming that \mathcal{C}_n is a majority class in \mathcal{KG} , and $W = \{w_0, w_1, w_2..w_{k-1}\}$, is a list of tokens with the top k TF-IDF score in \mathcal{C}_n . Then, we discard instance names that do not contain one of the words in W . As a result, we have a more balanced dataset which can be used to train a KG instance classifier.

4.3.1.3 SMOTE

Here, we apply SMOTE (Chawla et al. 2002), a state-of-the-art algorithm that oversamples the minority classes by automatically generating syntactic data using neighbouring instances. This technique is considered an alternative to random oversampling, which is a non-heuristic approach that balances classes by duplicating the samples in the minority classes to match the size of the largest majority class (Lemaître et al. 2017). However, random oversampling can often lead to model overfitting (Krawczyk 2016). Another reason for excluding random oversampling from being applied to the original KG dataset is the severity of the class imbalance problem. For instance, some small classes in YAGO have less than 10 instances while other classes have over 100,000 instances as depicted in Figure 4.1. Random oversampling will produce overwhelmingly

redundant instances for small classes, making the overfitting problem much worse.

4.3.1.4 TF-IDF + Oversampling

Combining undersampling and oversampling strategies is another approach for handling imbalanced datasets. Following such a hybrid strategy has been shown to improve the results of several classification tasks (Krawczyk 2016, Feng et al. 2021). While earlier work already experimented with other variations of this idea, here we propose a new method that combines TF-IDF undersampling with random oversampling. We aim to maintain a trade-off between handling the imbalance issue in both majority and minority classes. After applying the TF-IDF undersampling to the majority classes, we apply oversampling to make each class equal-size in terms of their instances. This includes creating repeated samples from minority classes.

4.3.1.5 TF-IDF + SMOTE

This strategy is similar to the previous one. However, instead of random oversampling, here, SMOTE is applied as an oversampling technique to handle the minority classes.

4.3.1.6 Cost-based Learning

All previous strategies belong to the category of data-level methods, often applied to the datasets prior to training a model. Another type of strategy (i.e., ‘algorithm level’) aims to modify existing machine learning models in an effort to reduce their bias towards the majority classes (Liu et al. 2020). A common algorithm-level approach is cost-sensitive learning (Elkan 2001) which modifies the class weights by assigning larger weights to minority class(es) and smaller weights for majority class(es) to be used during the model learning process. In this work, we evaluate a state-of-the-art approach (King & Zeng 2001), which gives each class a weight that is equal to its total number of instances divided by the distribution of instances across all classes as depicted in Equation 4.1, where $dict$ is a dictionary of classes and their assigned weights, $bincount(\mathcal{C})$ is the number of instances in the class \mathcal{C} .

$$Class_Weight_{dict} = n_{samples} / (n_{classes} * bincount(\mathcal{C})) \quad (4.1)$$

4.3.2 Building the KG Instance Classifier

After applying a specific sampling strategy from the previous section, the goal here is to build a machine-learning model using the resampled training data.

4.3.2.1 KG Instances Feature Representation

For any text classification task, the dataset needs to be represented in a numerical format in order for the machine learning algorithm to process it. For example, in the context of document classification, each document needs to be represented using its containing text. In this task, instance names need to be represented in a numerical format to learn such models. Further, in order to obtain a cleaner version of the datasets, standard text preprocessing techniques are applied. All instance labels are transferred into lowercase, and all stopwords and non-alphanumeric characters are removed. Different feature extraction techniques are proposed in the literature.

- Bag of Words (BoW) is a technique widely employed in different natural language processing and information retrieval tasks. A major limitation of this technique is that it only considers words, disregarding their sequence or position in a sentence or phrase (Altmel & Ganiz 2018). This makes it unsuitable for the task at hand, as the majority of real-world entities are described using long phrases, where the order of words is important to its context.
- TF-IDF is another common technique used to examine the importance of words in a document. It combines two parts: (i) Term Frequency (*tf*) which is the frequency of each term in the document and (ii) Inverse Document Frequency (*idf*) is the number of times this term appears in all documents in the document collection (i.e., other classes in the same KG). Hence, the weight value of a term here represents the importance of a token in a particular class in comparison to other classes in the KG.
- Word Embeddings technique has become a common way of numerically representing text data. It uses neural-network-based models that are able to learn generic, numerical representations of words based on their occurrences and usage in a large corpus. As a result, they are capable of catching the meaning of words that are often found in similar contexts. For example, **apple** and **orange** will be closely represented in the embeddings space. *GloVe* (Global Vector for Word

Representation) (Pennington et al. 2014) and *word2vec* (Mikolov et al. 2013) are two examples of state-of-the-art word embeddings models.

- Language models such as the Bidirectional Encoder Representation from Transformers or BERT consists of a multi-layers encoder based on attention mechanism (Vaswani et al. 2017). This deep architecture of stacked layers allows the model to capture both semantic and syntactic aspects of words (Peeters et al. 2020). Unlike directional models such as word2vec and GloVe, where each word can only have one vector representation, BERT generates context-specific vector representation of words which allows better capturing the meaning of a word in different contexts. Therefore, using a language model such as BERT, the vector representing the word `apple` the fruit will be further away from the one representing `Apple` the technology company.

Recent studies report that transformer models and pre-trained language models has significantly improved the state-of-the-art results in many natural language processing tasks, including text classification (Altinel & Ganiz 2018, Minaee et al. 2021). While training a language model using KGs is an option, this process can be time and resource-consuming. Thus, we opt to use pre-trained language models as they have great capability of preserving and capturing words meaning compared to other approaches.

4.3.2.2 Training the KG Classifiers

Our classification task is considered a multi-class classification task. Here, we explore the task at hand using three different state-of-the-art classifiers.

- A Logistic Regression (LR) classifier. Although this algorithm is commonly used for binary classification tasks, it can be used as a multi-class classifier with the *One-vs-Rest* or *One-vs-All* technique (Ma & Zhang 2015). This technique implies training multiple binary classifiers, where each represents a single class in a KG using instances from that class as positive samples and instances from other classes as negative samples.
- A Deep Neural Network (DNN). This is a deep learning model that trains a multi-layer sequential network. For this task, we built a DNN model with: an

input layer of a pre-trained language model trained on Google News ¹, followed by four fully connected hidden layers with 128, 128, 64, 32 rectified linear units. A dropout layer of 0.2 is added between each pair of dense layers for regulation. Finally, a *softmax* layer for multi-class classification, taking the total number of classes in the KG we are training a classifier for. The architecture of this model is inspired by previous high-performing models in different NLP tasks, such as in (Minaee et al. 2021) and (Collobert et al. 2011).

- A pre-trained BERT-based model (Maiya 2020). As we stated earlier, BERT is a language model that obtains state-of-the-art results in different classification tasks, particularly on large-scale tasks. The pre-trained model takes as input KG classes as classification labels with a list of their annotated instances. Instance names are then transferred into vector representations using the BERT preprocessor. Note that BERT requires the training and testing data to be encoded in a specific way, therefore, previously mentioned feature representation methods may not be suitable for this model.

4.4 Experiments

4.4.1 Experiment Settings

Our choices of machine learning models include using deep learning models, which often require more resources. Therefore, our experiments of training the KG instance classifiers were conducted on ShARC (Sheffield Advanced Research Computer), the University of Sheffield’s high-performance computing system. ShARC is open for the University of Sheffield staff and research students, and it is constructed of 12160 GiB of memory, a cluster of 2024 CPU cores, and 40 GPUs. This platform has facilitated running multiple experiments simultaneously for such a large task. For the following experiments, we use the two OAEI datasets proposed earlier in Chapter 3 as well as the OAEI KG track benchmark. In addition to the KG datasets, we evaluate our classifiers on the Web Data Commons (WDC) datasets, specifically the WDC web table dataset ². It consists of millions of relational tables describing real-world entities, annotated using the Schema.org vocabulary.

¹<https://tfhub.dev/google/tf2-preview/nlm-en-dim128/1>

²<http://webdatacommons.org/structureddata/schemaorgtables/>

4.4.2 Evaluation Metrics

Earlier in Chapter 2, we only discussed the way in which precision, recall, and f-measure are calculated in the context of matching tasks. Here, we provide a brief overview of how they are used for evaluating classification-like tasks.

These measures are based on the total number of true positives, false positives, and false negatives. Thus, given a set of instance names predicted to belong to a particular class by a KG classifier, (i) true positives are instance names with a correctly predicted class label, (ii) false positives are instance names with incorrectly predicted class labels, and (iii) false negatives are the instances that the matching system has missed. Precision is used to assess the ability of the KG classifier to correctly predict samples (Equation 4.2), while recall is about measuring the completeness of the predictions (Equation 4.3).

$$precision = \frac{true\ positives}{true\ positives + false\ positives} \quad (4.2)$$

$$recall = \frac{true\ positives}{true\ positives + false\ negatives} \quad (4.3)$$

F-measure is another measure that is often used in order to measure the trade-off between precision and recall by calculating their harmonic mean (Equation 4.4).

$$recall = \frac{2 * (precision * recall)}{(precision + recall)} \quad (4.4)$$

Other important measures for multi-class classification tasks are **macro average** and **micro average** f-measures. Both are used to evaluate the overall performance of multi-class classifiers such as those in our case. Micro average or otherwise known as accuracy is calculated by combining all predicted instances, then measuring precision, recall, and f-measure. The macro average is the mean of the f-measure of all the classes learned by the classifier. Micro average is the most used measure, however, it can be influenced by classes with majority samples (Liu et al. 2009). Therefore, in the case of imbalanced class distribution, the macro average gives more insight into the classifier’s ability to predict the minority classes too.

Table 4.1: The macro average results of the three trained machine learning models on the common KG dataset.

Dataset	LR			DNN			BERT		
	P	R	F1	P	R	F1	P	R	F1
NELL	0.21	0.18	0.16	0.78	0.75	0.76	0.86	0.84	0.85
DBpedia	0.16	0.14	0.11	0.47	0.45	0.45	0.61	0.57	0.58
YAGO	0.18	0.13	0.12	0.40	0.33	0.34	0.55	0.45	0.47
Wikidata	0.10	0.09	0.09	0.32	0.24	0.25	0.46	0.34	0.36

4.4.3 Results on Common KG Datasets

First, we evaluate our KG classifiers using common KG datasets including YAGO, NELL, DBpedia, and Wikidata. These are the four KGs from the two datasets proposed earlier in Chapter 3 and used for the common KGs track at OAEI. Table 4.1 shows the macro average results of the three classification models detailed earlier in Section 4.3.2: the Logistic Regression, the Deep Neural Network, and the BERT classifier on the testing datasets. One can clearly notice that the BERT classifier significantly outperforms the two other models. Therefore, in the following experiments, we show only the results of the BERT classifier.

We explained earlier in Section 4.4.2 that the macro average can be a better indicator of the ability of classifiers to correctly predict instances from the minority classes instead of biasing towards the majority classes. Table 4.2 illustrates the macro precision, recall and f-measure of the BERT classifier using each of the six different data balancing strategies discussed in Section 4.3.1 above. The table highlights in bold that the best classification results are obtained by the classifier trained using the TF/IDF undersampling with oversampling on all 4 datasets. When this strategy is applied, a notable increase in the macro average precision, recall, and f-measure was observed. For example, in terms of f-measure, the NELL dataset obtains a minimum increase of 10% (from 85% to 95%), while the Wikidata dataset yields the maximum increase of f-measure with 23% (from 36% to 0.59%). Figure 4.2 illustrates the macro f-measure when using our best-performing strategy, i.e., TF/IDF with oversampling, compared to the original datasets. The increase in macro average scores is largely attributed to improvements in the performance of the minority classes prediction.

In terms of undersampling, we apply two different strategies, random undersampling and using words with high TF/IDF scores for undersampling. We can conclude

Table 4.2: The macro average results for the KG classifier using different data balancing strategies on the four common knowledge graph dataset

Knowledge Graph	Sampling Strategy	Precision	Recall	F-measure
NELL	w/o resampling	0.86	0.84	0.85
	SMOTE	0.86	0.84	0.85
	Random Undersampling	0.13	0.17	0.13
	TF/IDF Undersampling	0.91	0.90	0.90
	TF/IDF with oversampling	0.95	0.95	0.95
	Cost-based learning	0.79	0.75	0.74
	TF/IDF + SMOTE	0.91	0.91	0.91
DBpedia	w/o resampling	0.61	0.57	0.58
	SMOTE	0.59	0.56	0.57
	Random Undersampling	0.13	0.18	0.11
	TF/IDF Undersampling	0.70	0.67	0.68
	TF/IDF with oversampling	0.74	0.74	0.73
	Cost-based learning	0.41	0.39	0.32
	TF/IDF + SMOTE	0.67	0.64	0.65
YAGO	w/o resampling	0.55	0.45	0.47
	SMOTE	0.57	0.59	0.58
	Random Undersampling	0.08	0.14	0.09
	TF/IDF Undersampling	0.64	0.50	0.52
	TF/IDF with oversampling	0.64	0.62	0.62
	Cost-based learning	0.20	0.24	0.16
	TF/IDF + SMOTE	0.67	0.54	0.56
Wikidata	w/o resampling	0.46	0.34	0.36
	SMOTE	0.49	0.38	0.37
	Random Undersampling	0.04	0.13	0.06
	TF/IDF Undersampling	0.60	0.51	0.53
	TF/IDF with oversampling	0.60	0.59	0.59
	Cost-based learning	0.32	0.29	0.23
	TF/IDF + SMOTE	0.45	0.38	0.40

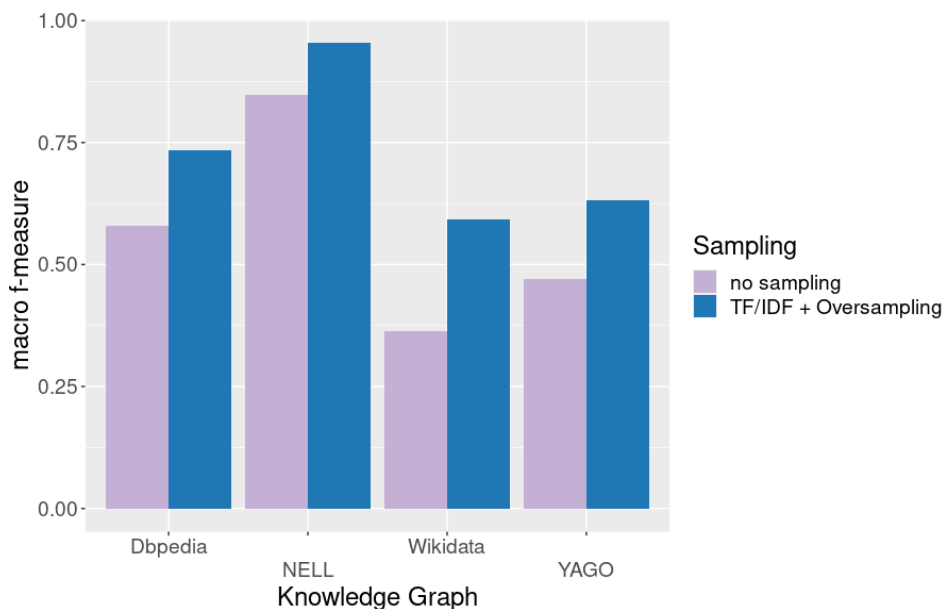


Figure 4.2: The macro f-measure of the BERT classifiers using the TF/IDF with oversampling compared to no sampling strategy on the common KG datasets

that while random undersampling notably reduces the instances per class, it has a detrimental impact on results due to its aggressive elimination process. Therefore, it seems to lower the accuracy of all classifiers compared to using no data balancing strategy. On the other hand, using our proposed TF/IDF undersampling method yields a notable performance improvement across the 4 KGs. For instance, training the BERT classifier on the Wikidata dataset without any sampling obtains 36% macro f-measure, while using TF/IDF undersampling increases that up to 53%. However, since this method only targets the majority classes, the classifier performance on the minority classes remains the same.

In regard to oversampling, utilizing the state-of-the-art SMOTE algorithm on KG instances also improves the macro f-measure of the classifiers compared to using no sampling. However, the improvement is not as significant as other strategies. Moreover, given the severity of the data imbalance issue in common KGs, combining undersampling of the majority classes with oversampling of the minority classes can be a more well-balanced approach. For example, the table shows that combining the TF/IDF undersampling with SMOTE results in a macro f-measure score closest to the best-performing strategy, i.e., TF/IDF and oversampling. In particular, on the NELL dataset, TF/IDF combined with SMOTE scored 4 percentage points lower (0.91 macro

Table 4.3: The macro average for the KG classifiers using the best-performing sampling strategy on the eight KGs from the DBkWik project

Knowledge Graph	Precision	Recall	F-measure
Star Wars	0.49	0.48	0.42
Star Wars Galaxies	0.03	0.06	0.03
Memory Alpha	0.36	0.39	0.33
Memory Beta	0.21	0.30	0.20
Star Trek Expanded Universe	0.42	0.44	0.36
Marvel Cinematic Universe	0.52	0.53	0.52
Marvel Database	0.24	0.32	0.24
The Old Republic	0.30	0.39	0.28

f-measure) than the best-performing configuration, i.e., TF/IDF with oversampling (0.95 macro f-measure). Further, generating syntactic samples with SMOTE does not seem to have the same effect on other KGs such as YAGO and Wikidata. This is due to the size of both KG datasets being twice the size of the NELL and DBpedia datasets in terms of both schema and instances data.

Finally, we implement cost-based learning, which applies different class weights to handle the biased classification results. While this method aims to maintain the training data, it results in a decrease in the macro f-measure across all 4 KGs. For instance, compared to not using class weights on DBpedia, the f-measure decreases from 58% down to 32%. As we mentioned earlier in Section 4.2, this can be due to this approach only being applied to the testing data and tends to over-penalize the classifier for biasing towards the majority classes.

4.4.4 Results on the DBkWik dataset

Table 4.3, illustrates the macro precision, recall, and f-measure of the KG classifiers trained on the 8 KGs in the DBkWik dataset. The results are based on using the BERT classifier along with the best-performing sampling strategy, which combines TF/IDF undersampling with oversampling. We can observe that in terms of macro f-measure, the results are not as high as what we have seen on the common KGs dataset. The highest results are observed with this dataset were on the **Marvel Cinematic Universe** with 52% precision, 53% recall, and an f-measure of 52%.

We discussed earlier in Section 3.5 how the current KG dataset, i.e., DBkWik,

has very different characteristics compared to real-world KGs. The differences are in terms of the number of instances, instance distribution, and the nature of the dataset being based on the entertainment domain. Here, we further discuss how these characteristics can also impact the performance of KG classifiers. First, machine learning algorithms, particularly neural-network-based techniques such as BERT, need large and well-balanced datasets for better results (Padurariu & Breaban 2019). However, as shown in Table 3.2 the number of instances in DBkWik is significantly smaller compared to its large schema. Second, this relatively small number of instances is not well-distributed across the KG classes. For example, Star Wars is one of the largest KGs in DBkWik in terms of its A-Box data size, it has over 145,000 instances. Nonetheless, over 100,000 of its instances belong to one class, while the rest are distributed across the other 241 classes.

The other factor that could impact the performance of KG classifiers is the range of topics covered by a KG. Figure 4.3 depicts the classification reports of twenty classes from YAGO and NELL compared to the DBkWik Marvel Cinematic Universe and Memory Alpha KGs. We can observe the difference between the classification results based on the topic of the classes. In YAGO and the marvel datasets, for instance, we can observe that classes such as `actor`, `song`, `episode`, `comicStory`, have a lower f-measure score in both KGs. As we mentioned earlier, the OAEI KGs are from the domain of TV, comics, and Gaming. Moreover, without a certain level of background knowledge, instances of these classes can be highly ambiguous and difficult to differentiate even for human annotators. Recent research has also studied the problem of classifying entities from such domains, particularly in the field of Named Entity Recognition (NER). For example, Sikdar et al. (2018) conducted an annotation task using Twitter data for NER task, and they report that annotating movies, TV shows, and songs were among the most challenging tasks for participants. Malmasi et al. (2022) also introduced a dataset for classifying ‘complex’ real-world entities and evaluated some state-of-the-art pre-trained language models on the task. These models have shown low performance when classifying entities of `Creative Work` without any external knowledge source compared to classifying other classes like `Organization` and `Product`.

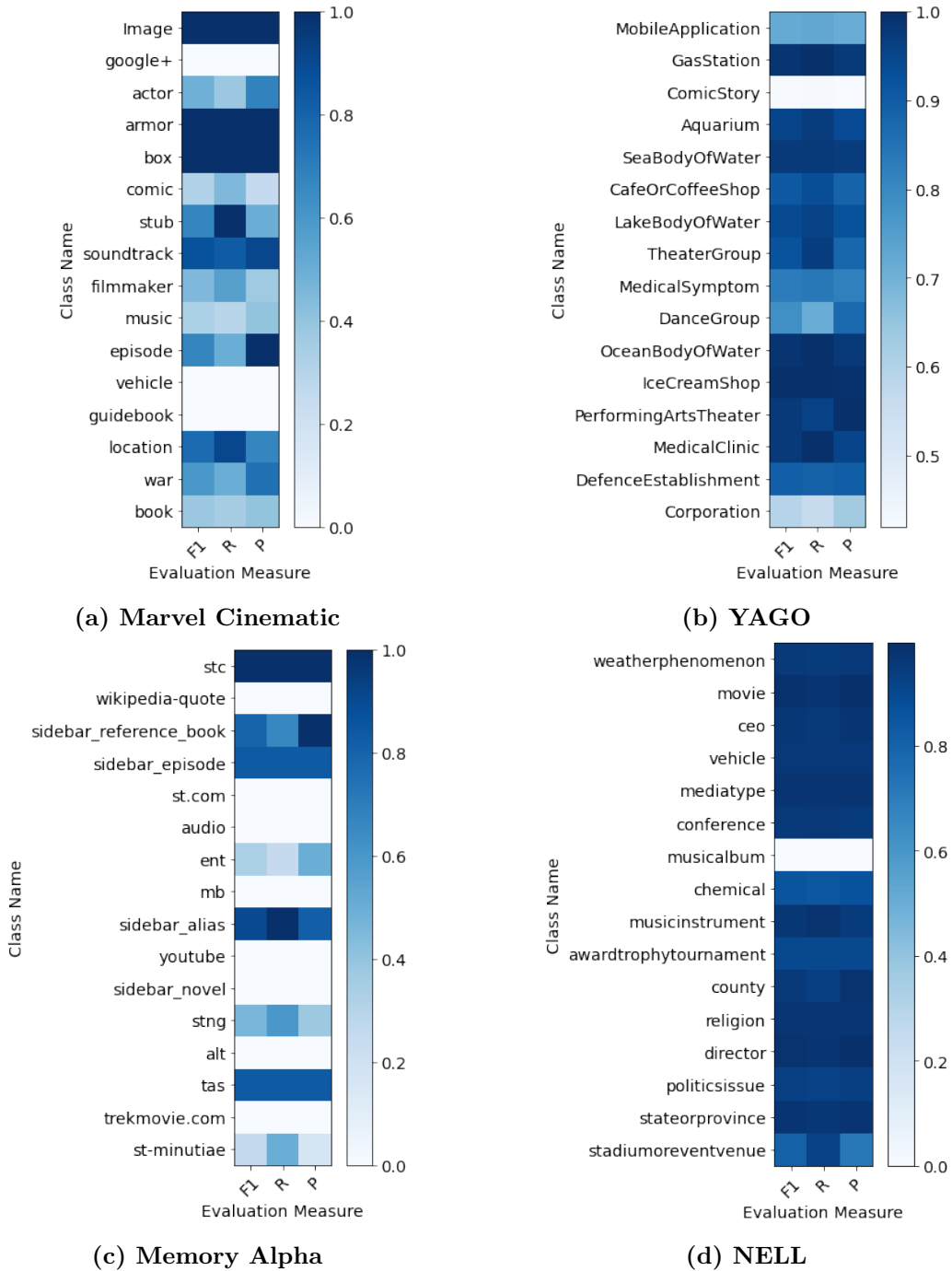


Figure 4.3: The classification reports of a 20 randomly sampled classes from the OAEI KG Marvel Cinematic Universe and Memory Alpha compared to two common KGs: YAGO and NELL.

4.4.5 Results on the Web Data Commons dataset

Over the last decade, many websites have started to embed structured data within their web pages using markup standards such as Microdata, JSON-LD, RDFa, and Microformats (Bizer et al. 2019). Among these adopted standard vocabularies, `schema.org`³ is one of the largest vocabularies describing real-world entities such as Organization, Person, Product, Place, and Event. Embedding structured data in webpages through adopting the `schema.org` vocabulary has been widely supported by world-leading search engine companies such as Google, Yahoo, and Microsoft Bing (Meusel et al. 2015). In addition to the embedded data being used by search engines for search result optimization, they are also recognized as valuable sources of data for a wide range of language-related tasks such as entity linking (Primpeli et al. 2019, Suhara et al. 2022) and KG construction (Ritze et al. 2016, Bizer et al. 2019). Moreover, this data has become publicly available as a result of the *Common Crawl Foundation* efforts. The Common Crawl⁴ is an open repository of regularly published web crawled corpora that can be accessed and analyzed by anyone. The October 2022 version⁵ of the Common Crawl contains 3.15 billion webpages. Further, the Web Data Commons (WDC) is another initiative that extracts and publishes structured data from the Common Crawl corpora, and so far, it has published 11 releases (Meusel et al. 2014).

These releases of structured data are potentially valuable resources for evaluating our KG classifier. This is because the dataset annotates real-world entities using their `schema.org` classes along with annotation of their properties. For instance, entities of the class `Hotel` can be described with properties like *name*, *address*, *description*, and *price range*, while `Product` entities can have other properties, such as *image* and *brand*. In particular, here we use the WDC `schema.org` table corpus⁶, which contains 4.2 million relational tables generated by obtaining `schema.org` data from the Common Crawl and grouping the data into separate tables by class/host combinations. Among the 43 `schema.org` classes available from this dataset, there is a table for each web host from which entities were extracted for each class. Each table details a list of columns, which represent the properties of entities used in a certain host. To illustrate, Figure 4.4 shows a sample of the table extracted from *Netflix* website for the class `Movie` which has the following columns: *name*, *description*, *director*, *date created*, *actor* and *genre*.

³<https://schema.org/>

⁴<https://commoncrawl.org/>

⁵<https://commoncrawl.org/2022/10/sep-oct-2022-crawl-archive-now-available/>

⁶<http://webdatacommons.org/structureddata/schemaorgtables/>

row_id	name	description	director	datecreated	actor	genre	page_url
0	Septien	Writer-director	{'name': 'Michael Tully'}	2011	{'name': 'Rachel Korine'}, {'name': 'Mark Darb Comedy		https://dvd.netflix.com/Movi
1	The Finest Hours	Recounting one	{'name': 'Craig Gillespie'}	2016	{'name': 'Eric Bana'}, {'name': 'John Ortiz'}, {'n Action &a		https://dvd.netflix.com/Movi
2	United	A devastating pl	{'name': 'James Strong'}	2011	{'name': 'Tim Healy'}, {'name': 'Sam Claflin'}, {Drama		https://dvd.netflix.com/Movi
3	The Haunting in Con	In this supernat!	{'name': 'Peter Cornwell'}	2009	{'name': 'Erik J. Berg'}, {'name': 'D.W. Brown'}, Thrillers		https://dvd.netflix.com/Movi

Figure 4.4: Example table describing movies extracted from *Netflix.com*

Table 4.4: The distribution of instances across classes in the WDC datasets

Place		LocalBusiness		CreativeWork	
Class	#Instances	Class	#Instances	Class	#Instances
Airport	510	Hospital	3,456	Book	254,127
CollegeOrUniversity	10,705	Hotel	142,453	Dataset	151,193
LakeBodyOfWater	373	Library	1,775	Movie	125,973
LandmarksOrHistoricalBuildings	2,929	RadioStation	22,481	MusicAlbum	51,881
Mountain	4,893	Restaurant	30,008	MusicRecording	71,723
Museum	3,819	ShoppingCenter	5,441	Painting	9,734
Park	2,797	SkiResort	697	Recipe	226,973
RiverBodyOfWater	954	StadiumOrArena	152	TVEpisode	69,407
School	39,429	TelevisionStation	10		
Total	66,409	Total	206,473	Total	961,011
Average	7,378	Average	22,941	Average	120,126

While entities of the same class will intuitively share similar properties as explained earlier, movies extracted from another host, such as *IMDB* could have slightly different properties. Moreover, different data publishers can have different guidelines and rules which can impact the data quality.

Here, we propose to use this corpus to create a benchmark for evaluating our KG classifier. Specifically, we propose to select only the classes that are considered as leaf nodes in the schema.org class hierarchy (i.e., those with specific annotated entities). Taking the 43 schema.org classes, we first create a hierarchical tree by mapping them to the current schema.org full hierarchy ⁷. Then, within each branch of the tree, we select the leaf classes and ignore intermediary classes. For instance, for the high-level class *Place*, we ignore intermediate nodes such as *BodyOfWater* and *CiviStructure*, but we include their leaf classes such as *RiverBodyOfWater* and *Airport* respectively. We also exclude subclasses that can be easily populated by using a controlled list such as *Country*, *Continent*, *City*, and *Language*. By applying this process, we obtain 26 leaf classes for three major schema.org classes, shown in Table 4.4.

⁷<https://schema.org/docs/full.html>

After determining the classes we will evaluate on, we then collect their instances from the WDC schema.org table corpus. As mentioned before, each table describes entities of one schema.org class from an entire web domain, e.g., *imdb.com*, and depending on the web domain, the table may have different columns. Here, to be consistent with the KG datasets we have used for evaluation before, we propose to only use the *'name'* column, and we merge entities of the same class from multiple web domains into a single table. Thus, the column named `name_t` is used to represent the KG instances used as training data. Then, the column named `schemaorg_class` is the name of the classes for which we are aiming to classify. Table 4.4 depicts the distribution of instances across the classes in the three datasets. We can notice that for `Place` and `Local Business`, their subclasses are very imbalanced in terms of their number of instances. For instance, `Hotel` is the largest class with over 140,000 instances, while `TelevisionStation` is the smallest class which has only 10 instances. On the other hand, subclasses of `Creative Work` are more well distributed.

Table 4.5 depicts the precision, recall, and f-measure results for all the classes in the three WDC datasets as well as the macro average for the three higher level classes. The presented results are based on the KG classifier using the best sampling strategy, i.e., TF-IDF with oversampling. From that table, we can notice that classifying entities of `Place` yields the highest results, with 0.92 f-measure. In this category, we can notice that the classifier is performing well in all classes, where the class `Museum` has the lowest f-measure (0.79). On the `LocalBusiness` dataset, which has an overall macro f-measure of 0.83, the classifier was not able to predict any correct instances from `TelevisionStation`. In addition to this class being the smallest class with only 10 instances, those instances are also very noisy. For instance, while some of those instances contain random numbers, others are describing arbitrary entities, e.g., `Country Roads Magazine` and `Sunshine Pages`. In terms of classifying `CreativeWork` entities, it seems to be more challenging than the two previous categories, with a macro f-measure of 0.68. This is largely due to low performance on the classes `Book`, `Recipe`, `Dataset`, and `Movie`. This is also consistent with our earlier findings in the KG datasets, that is classifying real-world entities can be very challenging as many can be very ambiguous.

Table 4.5: Results of the KG classifiers on the WDC datasets

Class Name	Precision	Recall	F-measure
Place			
Airport	0.85	0.87	0.86
CollegeOrUniversity	0.95	0.98	0.97
LakeBodyOfWater	0.94	0.96	0.95
LandmarksOr HistoricalBuildings	0.99	0.96	0.98
Mountain	0.91	0.98	0.94
Museum	0.74	0.84	0.79
Park	0.93	0.95	0.94
RiverBodyOfWater	0.96	0.99	0.97
School	0.92	0.96	0.98
Macro Avg	0.91	0.93	0.92
Local Business			
Hospital	1.00	1.00	1.00
Hotel	0.99	0.98	0.99
Library	0.98	0.99	0.98
RadioStation	0.91	0.99	0.95
Restaurant	0.97	0.97	0.97
ShoppingCenter	0.97	0.95	0.96
SkiResort	0.71	0.77	0.74
StadiumOrArena	0.77	0.99	0.87
TelevisionStation	0.00	0.00	0.00
Macro Avg	0.81	0.85	0.83
Creative Work			
Book	0.34	0.63	0.44
Dataset	0.62	0.58	0.60
Movie	0.65	0.58	0.61
MusicAlbum	0.67	0.55	0.60
MusicRecording	0.99	1.00	1.00
Painting	0.82	0.93	0.87
Recipe	0.24	0.52	0.33
TVEpisode	0.98	0.99	0.99
Macro Avg	0.66	0.72	0.68

4.5 Conclusion

In this chapter, we introduced our KG instance classifiers, multi-class classifiers that can predict the classes of a given KG instance name. We utilized BERT, a leading language model that has been showing remarkable results in different classification tasks. In order to cope with the KG class imbalance issue, our approach incorporated six different data-balancing strategies. Among these, two strategies were newly introduced in this work: an undersampling strategy that utilizes the TF/IDF weight to resample the majority classes in a KG, and another approach that combines the TF/IDF undersampling with random oversampling the minority classes. We evaluated our classification method on various KG evaluation datasets. The results with regard to the sampling have shown an improved classification result when combining TF/IDF with oversampling. We showed that our KG classifier can be also used for ontology population tasks by evaluating them on the WDC dataset. The latter is a large-scale and automatically constructed dataset largely used for different tasks in the semantic web, such as KG construction. The results have shown that the entity classification task can be more challenging in some domains such as entertainment compared to other real-world entities typically shared in common KGs. Finally, the approach introduced in this chapter has demonstrated promising results in training KG instance classifiers. However, there is still ample room for further research and exploration in this area.

Related Publication

- Fallatah, Omaima., Zhang, Ziqi., & Hopfgartner, Frank. The Impact of Imbalanced Class Distribution on Knowledge Graphs Matching. *In Proceedings of the 17th International Workshop on Ontology Matching (OM 2022) co-located with the 21st International Semantic Web Conference (ISWC 2022)*. CEUR-WS.

Chapter 5

Matching Knowledge Graph Classes

Contents

5.1	Introduction	90
5.2	Related Work	91
5.2.1	Knowledge Graphs Matching Overview	91
5.2.2	Limitations of Current Methods	93
5.3	Approach	93
5.3.1	Overview	93
5.3.2	Pre-processing	94
5.3.3	Instance-based Matcher	95
5.3.4	Name Matcher	98
5.3.5	Final Alignment Selection	98
5.4	Evaluation	99
5.4.1	Experimental Settings	99
5.4.2	Datasets	99
5.4.3	KGMatcher+ Results and Discussion	100
5.4.4	The Impact of Different Matching Component	104
5.5	Conclusion	109

5.1 Introduction

The previous chapter introduced a method for training KG instance classifiers that can be utilized to match their classes. While the problem of ontology matching has been well studied in the last decade, mapping large-scale KGs remains a challenging task. Similar to ontologies, KG entities are highly heterogeneous, since many real-world entities can be described using different vocabulary. Nevertheless, while ontologies primarily focus on modeling the schema of a specific domain, cross-domain KGs are known for describing numerous instances. Due to their nature of being largely generated in a semi-automated manner, KGs are less well-formed compared to manually created and well-designed ontologies. Current matching methods are mainly focused on well-formed ontologies, while the problem of matching automatically curated and large KGs remains unsolved. As highlighted in Chapter 2, the majority of the state-of-the-art methods are highly dependent on string and structural-based techniques, common KGs often lack the longer textual descriptions, e.g., comments, required by such methods. In terms of structural-based similarity measures, despite that some KGs lack the schematic information required by such methods, they are often used to verify or support element-level matchers.

This chapter proposes a matching system named KGMatcher+, a novel domain-independent method for mapping classes in large KGs. This approach combines two matching techniques: a string-based method with an instance-based approach. The latter only uses annotated instance names to generate similar class pairs, building on the KG classifiers detailed in Chapter 4. Our method utilizes the large number of instances typically annotated in large and common KGs and is able to cope with the imbalanced population of KG classes. This is to answer our third research question, **RQ3**: *How can we effectively use instances to match KG classes while addressing the unbalanced population issue?*

This chapter is organized as follows: Section 5.2 discusses some current KG matching methods and summarizes their limitations. In Section 5.3, we introduce our hybrid matching approach, while Section 5.4 evaluates the proposed method. Finally, Section 5.5 concludes this chapter.

5.2 Related Work

In Chapter 2, we detailed the state-of-the-art matching techniques, as well as the current ontology matching methods. Here, we give a brief overview of the most recent approaches utilized for KGs matching in particular (Section 5.2.1), then we highlight the limitation of these methods, particularly on the task of matching large-scale and less-well-formed KGs in Section 5.2.2.

5.2.1 Knowledge Graphs Matching Overview

As mentioned earlier, that in order to achieve better matching results, most systems combine several existing matching techniques. While Section 2.2 provided a comprehensive review of various ontology matching techniques, this section aims to provide a high-level overview of the most recent matching systems. The purpose of this overview is to establish a context for comparing our own matching approach later in this chapter. By providing this overview, we aim to demonstrate the relevance and significance of our research by contextualizing it within the existing literature and showcasing its potential contributions to the field of ontology and KG matching.

Given the nature of the task, the most common matching technique across all tools is to determine equivalent class pairs based on terminological similarity. This is one of the element-level techniques discussed in Section 2.2.1.1, which discovers similar entities by utilizing the textual annotations defined in the KG entities, e.g., URIs, labels, and comments. It is often done by applying a variety of string-similarity metrics such as string equality, edit distance, and n-gram. To name a few, terminological techniques are the base of Matcha (Faria et al. 2022) formally known as AML, and LogMapLite, a light weight version of LogMap that only applies essential (efficient) string matching techniques (Jiménez-Ruiz 2020). Another common element-level technique used for KGs schema matching is language or semantic-based, which depends on utilizing background knowledge (Portisch et al. 2021). For example, Wiktionary system (Portisch & Paulheim 2022b) uses *Wiktionary* while LSMatch (Sharma et al. 2021) utilizes *WordNet*. Another form of incorporating background knowledge is by using another knowledge base or a KG, e.g., *WebisALOD* a KG with hypernym relations that is used in the ALOD2Vec system (Portisch & Paulheim 2022a). Such matching techniques align candidate class pairs in the source and target KGs by mapping them to terms or entities in the background knowledge resource first. Recent systems, such as TOM (Kossack

et al. 2022), Fine-TOM (Knorr & Portisch 2022), and OTMapOnto (An et al. 2021), have also explored using transformer models as background knowledge given their remarkable role on different NLP tasks.

Determining equivalent schema entities by studying the structure of the two input KGs is another strand of matching techniques (Section 2.2.1.2). They exploit structural information available in well-formed ontologies like disjoint axioms, such as in AML (Faria et al. 2019) and LogMap (Jiménez-Ruiz 2020), or by using mathematical models to analyze KG structures, such as in FCAMap-KG (Chang et al. 2019). Structural-based techniques can be error-prone therefore, they are mainly used to refine element-level alignments. A useful example is ATBox/ATMatcher (Hertling & Paulheim 2020a), a matching system that uses neighboring entity information to filter mappings initially discovered by its string-based matcher. However, as we discussed in Chapter 3, unlike many well-curated domain-specific ontologies, public and cross-domain KGs often lack such structural information due to their automatically generated nature.

Instance-based techniques align schema entities (i.e., classes or properties) based on the overlap of their instances. Earlier in Section 2.2.1.2, we recognized that due to the lack of instance data in formal ontologies, there have been fewer studies of instance-based methods compared to terminological methods. Similarly, instance-based is the least incorporated technique among KG matching methods, even though typical KGs have an abundance of instances. For example, as discussed before in Chapter 3, DBpedia 2016-04 has over 5 million instances and 754 classes; YAGO consists of a similar number of instances as DBpedia, but has over 500,000 classes; NELL, on the other hand, has around 300 classes and over 3 million instances (Heist et al. 2020, Ringler & Paulheim 2017). This is orders of magnitude larger than domain-specific KGs like the one in the OAEI’s KG track, which have less than 300,000 instances on average (Pour et al. 2021). This number of instances can make measuring the overlap of instances a very challenging task. Nonetheless, we believe that instance-based techniques can be potentially more helpful, and indeed, there has been an increasing number of methods for matching large KGs schema entities, particularly properties, using instance-based techniques (Zhang et al. 2017, Ayala et al. 2021). DOME (Hertling & Paulheim 2019) is one of the recent approaches that incorporate an instance-based matcher for KG class matching. The system aligns KG instances first and then measures the overlap of shared instances to align classes using *dice*, readers can refer to Section 2.2.1.2 for

details on current instance-based methods.

5.2.2 Limitations of Current Methods

From the above analysis of the current KG matching method, the following three patterns were observed. **First**, the majority of current systems are built for well-formed and well-structured ontologies with fewer instances. As a result, they mainly depend on terminological and structural techniques while instance-based ones are less well-studied. While current tools are able to produce high-quality results for well-formed and manually curated datasets, such techniques are not as suitable for real-world large-scale KGs that lack detailed textual and structural descriptions. **Second**, many of the current matching systems are still suffering in terms of scalability to handle large matching tasks. In recent OAEI campaigns, Pour et al. (2021) report that many matching systems face challenges managing the balance between efficiency and effectiveness, particularly for large tasks such as large-scale biomedical ontologies, and KGs matching. **Third**, some matching strategies, particularly state-of-the-art methods, were originally built to handle domain-specific ontologies, such as in the biomedical domain. This is due to many reasons, one being that research in that area has established a need for linking such domain-specific resources. The other reason is that most available datasets were also domain-dependent, e.g., OAEI anatomy, conference, and the current KG track. However, methods used in this single-domain setting may not be applicable for domain-independent and real-world settings, where classes may contain information about real-world entities described with different vocabulary.

5.3 Approach

5.3.1 Overview

Preliminaries. While some notations have been mentioned before, here we repeat some of them to place everything in the context of KGs matching. Given two input knowledge graphs \mathcal{KG} and \mathcal{KG}' , we define the correspondence between two classes $\mathcal{C} \in \mathcal{KG}$ and $\mathcal{C}' \in \mathcal{KG}'$ as the tuple $\langle \mathcal{C}, \mathcal{C}', v \rangle$ where $v \in [0, 1]$ is the similarity value of \mathcal{C} and \mathcal{C}' . Each class in the two KGs has a set of instances, $\mathcal{C}_n = \{i_0, i_1, i_2, \dots, i_n\}$ and $\mathcal{C}'_m = \{i_0, i_1, i_2, \dots, i_m\}$. The following sections describe different modules of KGMatcher+, as illustrated in Figure 5.1.

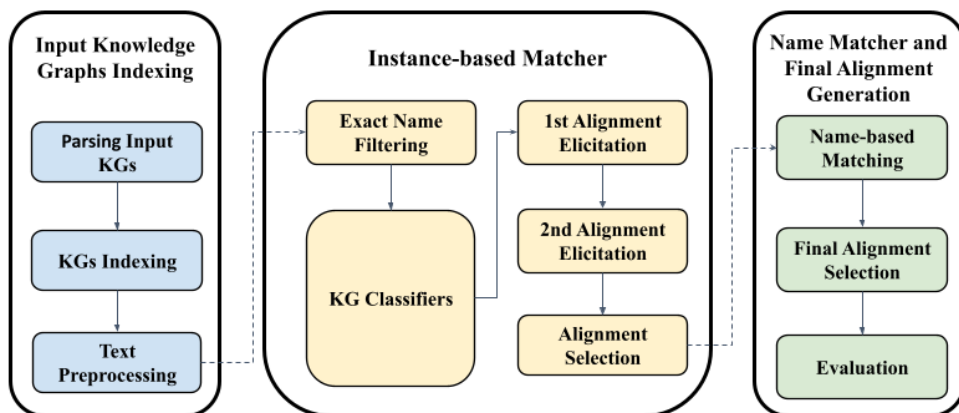


Figure 5.1: The architecture of the proposed system (KGMatcher+)

The system starts with parsing the two input KGs and applying general text preprocessing methods. The second component of the system is the instance-based matcher. It adopts a data-driven approach where a two-way classification technique is followed to map classes from two KGs. Further, it is based on the extent to which the instances of a class in one KG are classified as instances of classes in another KG. First, after removing classes with exact names in order to reduce the search space, the matcher uses the processes of resampling and building KG instance classifiers as described in the previous chapter (Section 4.3.1 and Section 4.3.2). Next, the trained classifiers are then applied to the other KG to classify its instances. Mapping pairs of classes are then derived based on the classification results of the two classifiers. The two stages will result in two directional alignment sets, denoted as $A_{\mathcal{KG} \rightarrow \mathcal{KG}'}$ which is a set of correspondences between classes from \mathcal{KG} and \mathcal{KG}' respectively, and $A_{\mathcal{KG}' \rightarrow \mathcal{KG}}$ which is a set of correspondences in the opposite direction. The two directional alignments are then aggregated in order to obtain one alignment set for this matcher $A_{instance}$. The second matcher is based on string and semantic similarities, which belongs to the element-level matcher category. This matcher uses class labels to generate equivalent class pairs, denoted as A_{name} . Finally, the method ends by generating the final alignments \mathcal{A} .

5.3.2 Pre-processing

The first component of the matcher consists of three steps. It starts by parsing the two input KGs in order to extract and then separately index their lexical annotations. Given a KG, an index of its classes is generated by following the standard free text

indexing approach for search engines. The details of this part of the method, i.e., pre-processing KG instances, have been previously explained in Section 4.3.2.1. Nonetheless, KGs class names are also preprocessed by KGMatcher+. Similarly, KG classes can also be described with multiple words that are often separated with an underscore or concatenated with or without camel case style. For example, `placeofworship` and `ReligiousBuilding`. In terms of the underscore character, it is replaced with a space character. For concatenated labels, a word segmentation process that utilizes a dictionary is applied to infer the spaces between words, while the camel case is also replaced with a white space.

5.3.3 Instance-based Matcher

This matcher maps KG classes based on their shared instances using the KG instance classifiers trained in Chapter 4. The results of that chapter have shown that KGs are similar in principle to other datasets in terms of how unique the imbalanced data issue can be in different KGs. Therefore, in order to capture as many ‘true-positive’ alignments as possible, we perform the matching process in a two-way classification fashion. Each process trains a self-supervised KG classifier using one of the two input KG data. Then, a classifier trained on \mathcal{KG} for instance can be used to predict the class C to which a given ‘instance’ name may belong. The two KG classifiers are later used to elicit candidate class alignments. In the following sections, we detail each step of this matcher as they are depicted in Figure 5.1.

5.3.3.1 Exact name filter

The first component starts by filtering classes with exact names. This works by excluding class pairs that share exactly the same labels from \mathcal{KG} and \mathcal{KG}' . Our goal here is to use the instance-based matcher to leverage the final alignments with class pairs that are likely to not be discovered by simple string matchers. Moreover, typical large KGs can have hundreds of classes to be matched, which can affect the performance of any matching method. Therefore, this step also serves as a blocking strategy that reduces the search space for the matcher. Furthermore, classes with only one instance are eliminated from this process, as the BERT-based classifier is unable to learn from such classes.

5.3.3.2 KG Classifiers

At this part of the matching approach, two KG classifiers are trained for \mathcal{KG} and \mathcal{KG}' using the approach detailed in Chapter 4. KG instances are sampled using the best-performing sampling strategy, i.e., TF-IDF undersampling and oversampling (Section 4.3.1). Then, the two trained classifiers, as detailed in Section 4.3.2, $\mathcal{CL}_{\mathcal{KG}}$ and $\mathcal{CL}_{\mathcal{KG}'}$ will be utilized in the following two steps of the matching system.

5.3.3.3 Alignment Elicitation

The alignment elicitation phase is split into a first and a second process in order to perform the two-way matching as shown in Figure 5.1. Here, we derive class mapping candidates based on the classification results of $\mathcal{CL}_{\mathcal{KG}}$ and $\mathcal{CL}_{\mathcal{KG}'}$. The first alignment elicitation is performed in the direction $\mathcal{KG} \rightarrow \mathcal{KG}'$ to generate mapping candidates, denoted as $A_{\mathcal{KG} \rightarrow \mathcal{KG}'}$. Thus, \mathcal{KG} will be treated as the source KG and \mathcal{KG}' as the target. Subsequently, $\mathcal{CL}_{\mathcal{KG}}$ is applied to all instance names in \mathcal{KG}' and will return a list of classes in which each instance name may belong to in \mathcal{KG} . By taking the class with the highest probability value, each instance in \mathcal{KG}' will have a predicted class in \mathcal{KG} . Then, to generate $A_{\mathcal{KG} \rightarrow \mathcal{KG}'}$, we pair each class \mathcal{C}' with the class \mathcal{C} that receive the majority votes, based on applying $\mathcal{CL}_{\mathcal{KG}}$ to instances of \mathcal{C}' . To illustrate, a correspondence between \mathcal{C}_4 and \mathcal{C}'_2 is added to the candidate alignments if the majority of \mathcal{C}_4 instances were classified as instances of \mathcal{C}'_2 . Then, in order to generate a similarity value between $[0,1]$, we use the percentage of instances that voted for the majority class. As an example, the candidate pair $\langle \mathcal{C}_4, \mathcal{C}'_2, 0.57 \rangle$ indicates that 57% of \mathcal{C}'_2 instances were predicted to be \mathcal{C}_4 when applied to $\mathcal{CL}_{\mathcal{KG}}$. The second elicitation process is done in the opposite direction by reversing the roles of the two input KGs to create $A_{\mathcal{KG}' \rightarrow \mathcal{KG}}$. Further, Candidate alignments are generated based on the output of the classifier $\mathcal{CL}_{\mathcal{KG}'}$.

5.3.3.4 Alignment Selection

The next phase of the instance-based matcher aimed at unifying the two directional alignments $A_{\mathcal{KG} \rightarrow \mathcal{KG}'}$ and $A_{\mathcal{KG}' \rightarrow \mathcal{KG}}$ generated during the elicitation process. In order to select the final alignments for the instance-based matcher, we follow the state-of-the-art iterative approach. This approach was first introduced as part of CroMatcher (Gulić et al. 2016), and it significantly outperforms other alignment selection methods, as we

	\mathcal{C}'_0	\mathcal{C}'_1	\mathcal{C}'_2	\mathcal{C}'_3	\mathcal{C}'_4	\mathcal{C}'_5	\mathcal{C}'_6
\mathcal{C}_0	0.0	0.01	0.15	0.55	0.0	0.22	0.1
\mathcal{C}_1	0.2	0.0	0.0	0.12	0.0	0.13	0.66
\mathcal{C}_2	0.11	0.14	0.72	0.0	0.0	0.0	0.21
\mathcal{C}_3	0.88	0.11	0.0	0.02	0.8	0.0	0.4
\mathcal{C}_4	0.0	0.4	0.55	0.17	0.12	0.71	0.0
\mathcal{C}_5	0.13	0.0	0.02	0.0	0.19	0.09	0.0
\mathcal{C}_6	0.0	0.0	0.0	0.92	0.16	0.0	0.0



 $Final_Alignments = \{(\mathcal{C}_1, \mathcal{C}'_6, 0.66), (\mathcal{C}_2, \mathcal{C}'_2, 0.72),$
 $(\mathcal{C}_3, \mathcal{C}'_0, 0.88), (\mathcal{C}_4, \mathcal{C}'_5, 0.71), (\mathcal{C}_6, \mathcal{C}'_3, 0.92)\}$

Figure 5.2: An example of calculating the final alignment using the method in (Gulić et al. 2016)

reviewed in Section 2.2.3.6. In order to combine both alignment sets, each directional alignment will be first stored into an alignment matrix of a dimension of $\mathcal{N} \times \mathcal{M}$, where \mathcal{N} is the number of classes in \mathcal{KG} and \mathcal{M} the number of classes in \mathcal{KG}' . Then, the tables are populated based on the two-directional alignment sets created earlier. To aggregate the two matrices, we take the average similarity value of each pair. For example, if $(\mathcal{C}_4, \mathcal{C}'_5, 0.68)$ in $A_{\mathcal{KG} \rightarrow \mathcal{KG}'}$ and $(\mathcal{C}'_5, \mathcal{C}_4, 0.73)$ in $A_{\mathcal{KG}' \rightarrow \mathcal{KG}}$ the pair aggregated similarity value will be 0.71. Subsequently, we follow the iterative algorithm in (Gulić et al. 2018) to select final alignment $\mathcal{A}_{instance}$. Given an alignment matrix and a threshold t , this method goes through each row at a time and selects the highest correspondence in each row, if the similarity value for that correspondence is beyond t (e.g., bold text in 5.2). When a class is involved in two correspondences (e.g., \mathcal{C}_3 in rows 1 and 7), only the one with the higher similarity is retained (e.g., $(\mathcal{C}_6, \mathcal{C}'_3, 0.92)$), and the previously selected correspondence is deleted. This process takes place iteratively until all classes are selected within a correspondence and no changes are to be made. In terms of the threshold value for this method, we assign it to 0.22 following the settings in (Gulić et al. 2018, 2016). Section 2.2.3.6 provides some further details on this method and other final alignment selection methods.

5.3.4 Name Matcher

The second matching component in KGMatcher+ belongs to the element-level matcher category. It calculates the similarity of KG classes based on the string and the semantic similarity of their names. Given a set of all possible correspondences between classes in \mathcal{KG} and \mathcal{KG}' , generated with an exclusive pairwise comparison, we measure the word embedding similarity and the edit distance similarity of the two class names. We only apply the two similarity measures to KGs class labels, as not all KGs provide other longer descriptions such as comments. For the edit distance similarity, we calculate the normalized levenshtein distance for each class pair. This method normalizes the edit distance value by the length of the longer string to get a value between $[0.0, 1.0]$. In terms of the word embedding similarity, a Google pre-trained `word2vec` model is used to represent class names and measure their cosine similarities in the Vector Space Model where semantically similar words are represented closer to each other. Following the same approach in Section 5.3.2, concatenated strings such as `awardtrophytournament` are segmented into multiple words. Thus, in the case of a multi-word class name, the matcher aggregates the vector representation of each word composing the class name by taking the element-wise average of the vectors of each composing word. We then choose the maximum of the two similarity measures, if the similarity scores are higher than a threshold t_n . We set t_n to 0.8 which is in line with previous element-level methods that combine multiple similarity measures such as (Hertling & Paulheim 2020a, Nkisi-Orji et al. 2018). To illustrate, assuming that a pair of the two classes `RailwayStation` and `TrainStation` where their word embedding similarity is 0.83 and their edit distance is 0.56, we select the maximum similarity value, i.e., the word embedding similarity which is also higher than the t_n . However, if the two similarity scores of a pair are lower than the t_n , then that pair will not be added to the candidate alignment set. The output of this matcher is a candidate alignment set \mathcal{A}_{name} to be combined with the instance matcher alignments ($\mathcal{A}_{instance}$).

5.3.5 Final Alignment Selection

To generate the final alignments of the matching system, the instance-based alignments are combined with the name matcher alignments. This is done by following the same alignment selection method used earlier to combine the two directional instance-based alignments, as detailed in Section 5.3.3.4. The same state-of-the-art method is followed

by treating \mathcal{A}_{name} and $\mathcal{A}_{instance}$ as directional alignments to generate the final alignment of KGMatcher+ (i.e., \mathcal{A}).

5.4 Evaluation

In this section we aim to evaluate our hybrid matching approach (KGMatcher+) on the task of matching large KGs. We also compare KGMatcher+ results against several matching methods including state-of-the-art OM systems. Since the proposed method incorporates an instance-based method, our evaluation hypothesis is that it will improve the state-of-the-art results on matching common KGs.

5.4.1 Experimental Settings

The proposed matching system has been implemented in python. Then, the *Matching Evaluation Toolkit (MELT)* (Hertling et al. 2019) was used to wrap KGMatcher+ to make it available as an OAEI system and for easier results reproducibility. MELT¹ evaluation allows to test any matcher with different datasets as long as they are in the appropriate KG or ontology format². In terms of evaluation experiments, they have been implemented in Java on a VM with 128GB of RAM, 16 vCPUs (2.4 GHz), which is similar to the OAEI evaluation configurations. All the evaluation experiments were produced using the MELT evaluation platform, which is the official evaluation tool used for the majority of OAEI tasks, including the two KG tracks. MELT calculates precision, recall, and f-measure for the produced alignment compared to the reference alignments provided with each task. Some experiments were also conducted on ShARC (Sheffield Advanced Research Computer), the University of Sheffield’s high-performance computing clusters, detailed in Chapter 4.

5.4.2 Datasets

To evaluate the ability of the proposed method to match the schema of KGs, we use the OAEI benchmarks. As mentioned before, OAEI is an annual evaluation event with the aim to evaluate the performance of matching tools on a variety of matching tasks. We use the KGs from each track as inputs to KGMatcher+, then we compare the

¹<https://github.com/dwslab/melt>

²<https://dwslab.github.io/melt/matcher-evaluation>

alignment produced by KGMatcher+ to those in the reference alignment of each task. Similar to the OAEI standards, we calculate precision, recall, and f-measure using the MELT evaluation tool. We based our evaluation on OAEI datasets, as they are widely recognized and used by current work to evaluate matching systems on mapping KGs.

The *Common Knowledge Graphs track* evaluates the ability of matching systems to map the schema of large-scale, cross-domain, and automatically constructed KGs. This track was introduced to OAEI in 2021, and it consists of two matching benchmarks we proposed earlier in Chapter 3. It has two tasks, where the first one aligns classes from NELL and DBpedia, and the second one maps classes from YAGO and Wikidata. For this task, the total number of class alignments is 129 for the NELL-DBpedia benchmark, and 304 for the YAGO-Wikidata benchmark.

The *Knowledge Graph track*, which consists of five different matching tasks that target matching the schema (classes and properties), and instances of 8 KGs from the DBkWik project. In terms of class alignments, the aggregated number of class alignments for the 5 benchmarks is 49.

5.4.3 KGMatcher+ Results and Discussion

Here, we present the results of our proposed method KGMatcher+ which participated in OAEI 2021 and OAEI 2022 campaigns. We compare KGMatcher+ results with all public OAEI systems from both campaigns. The results are based on the two datasets in the common KG track and the KG track. Note that the latter includes five test cases for matching classes, properties, and instances. Here, we include the aggregated results of matching classes across the five test cases. The following tables illustrate the results obtained by all systems on the three tasks. KGMatcher+ was able to outperform all systems on the task of matching common KGs in 2021 (Pour et al. 2021) and 2022 (Pour et al. 2022). Furthermore, it was the second to the best-performing system in the latest OAEI 2022 campaign on the KG track.

In terms of the common KGs track, we can observe from Table 5.1 and Table 5.2 that KGMatcher+ outperforms all baselines on both datasets, recording the highest recall and f-measure. In terms of the NELL-DBpedia dataset, KGMatcher+ outperforms all systems by a minimum of 6% in the f-measure score and by 11% in terms of recall. Furthermore, the proposed method outperforms AML and LogMap, two state-of-the-art ontology matching systems and leading OAEI participants. For instance, on the task of matching NELL and DBpedia, KGMatcher+ achieves 0.95 f-measure, which is

0.06 higher than AML which achieves 0.89. Although the newest version of AML which has been introduced as Matcha in OAEI 2022 has slightly improved AML results in terms of f-measure (0.90), it is still far from the KGMatchers+ results. On the YAGO-Wikidata dataset, KGMatcher+ surpasses all systems by 4% in terms of recall and 2% in terms of f-measure. One can also notice that all systems score lower on YAGO-Wikidata. This is likely due to the size of YAGO-Wikidata, which is twice the size of NELL-DBpedia. Readers should also note that while all systems were able to generate class alignments when applied to the full-size version of YAGO-Wikidata, some systems were not able to scale to this task. This includes Matcha, AML, and LogMap, which were only able to process the smaller version of the dataset. The smaller dataset has all the schema data, but with a smaller subset of instances per class. However, those systems do not utilize any instances during the matching process. Thus, applying them to a smaller dataset will not affect their class alignment results. It is also worth mentioning that OTMapOnto participated only in OAEI 2021 campaign, and it is not a public OAEI system. In addition, due to its architectural complexity, we are not able to reproduce its environment to test it on the new YAGO-Wikidata dataset, even with the author's appreciated help.

Table 5.3 depicts the results on the OAEI KG track, where we can infer that KGMatcher+ is still among the top-performing systems. AML and ATMatcher are the best-performing systems with an f-measure of 0.89 and 0.87 respectively, and KGMatcher+ achieves an f-measure of 0.80. However, while KGMatcher+ only uses the labels of entities to produce class alignments, other systems that outperform it on this task implement a larger variety of matcher combinations. In addition, they target not only the labels of KG entities, but also consider other entities' metadata. AML, for instance, incorporates nine different basic matchers including structural matchers, and other filters to further improve the quality of the matching results. Moreover, the majority of OAEI systems incorporate multiple string-based techniques such as n-gram, prefixes, and suffixes, while the name matcher component of KGMatcher+ is a fairly basic element-level matcher. For example, ATMatcher implements a variety of element-level matchers, including one that aims to find specific stopwords common in a certain KG or ontology. These could be a set of words that repeatedly appear in entity labels. ATMatcher then considers such words as stopwords to be removed prior to applying any further string-based matching techniques. This allows the system to discover the similarity between pairs like `<sidebar_starship, starship>` and

Table 5.1: Evaluation results on the OAEI *Common Knowledge Graphs* track - NELL-DBpedia task

Matching System	Precision	Recall	F-measure
AML	1.00	0.80	0.89
LogMap	0.99	0.80	0.88
KGMatcher+	1.00	0.91	0.95
ATMatcher	1.00	0.80	0.89
LSmatch	0.96	0.75	0.84
ALOD2Vec	1.00	0.80	0.89
Wiktionary	1.00	0.80	0.89
LogMapLite	1.00	0.60	0.75
DOME	0.99	0.63	0.77
OTMapOnto	0.90	0.84	0.87
Matcha	1.00	0.81	0.90
FCAMap-KG	1.00	0.78	0.87
Fine-TOM	1.00	0.78	0.87
TOM	1.00	0.68	0.81

Table 5.2: Evaluation results on the OAEI *Common Knowledge Graphs* track - YAGO-Wikidata task

Matching System	Precision	Recall	F-measure
AML	1.00	0.80	0.89
LogMap	1.00	0.76	0.86
KGMatcher+	0.99	0.84	0.91
ATMatcher	1.00	0.77	0.87
LSmatch	0.96	0.63	0.76
ALOD2Vec	0.99	0.77	0.86
Wiktionary	1.00	0.74	0.85
LogMapLite	1.00	0.70	0.81
DOME	1.00	0.72	0.83
Matcha	1.00	0.80	0.89
FCAMap-KG	1.00	0.40	0.56
Fine-TOM	1.00	0.71	0.83
TOM	0.99	0.72	0.83

Table 5.3: Evaluation results on the OAEI *Knowledge Graphs* track

Matching System	Precision	Recall	F-measure
AML	0.98	0.77	0.86
LogMap	0.93	0.71	0.81
KGMatcher+	1.00	0.66	0.80
ATMatcher	0.97	0.79	0.87
LSmatch	0.97	0.64	0.78
ALOD2Vec	1.00	0.67	0.80
Wiktionary	1.00	0.67	0.80
LogMapLite	1.00	0.54	0.70
DOME	0.92	0.66	0.77
OTMapOnto	0.73	0.80	0.77
Matcha	0.00	0.00	0.00
FCAMap-KG	1.00	0.70	0.82
Fine-TOM	1.00	0.66	0.80
TOM	1.00	0.71	0.83

$\langle \text{sidebar_novel}, \text{novel} \rangle$, if *sidebar* was a corpus specific stopword. Nonetheless, given the topics of this dataset, such pairs will not be mapped by word embedding models or by an edit-distance similarity. Moreover, depending on the coverage of the entertainment domain in a background knowledge resource, such as WordNet and Wiktionary, it can be also challenging to discover such pairs using semantic methods like in ALOD2Vec, Wiktionary system, and LSMatch.

Further, we analyzed the extent to which different methods are able to recover matching pairs that contain an imbalanced number of instances (to be called ‘imbalanced pairs’). Based on the common KG track, we define a pair $(\mathcal{C}, \mathcal{C}')$ as an imbalanced class pair, if one of the classes is a majority class and the other is a minority class, or if both classes are considered as minority classes. As a result, We observed 40 pairs of imbalanced classes in YAGO-Wikidata and 22 in NELL-DBpedia. Figure 5.3 illustrates the percentages of pairs discovered by the OAEI systems evaluated earlier compared to the proposed method. On both datasets, we can observe that KGMatcher+ was able to discover over 60% of such imbalanced pairs (64% in NELL-DBpedia and 63% in YAGO-Wikidata). On YAGO-Wikidata, for instance, KGMatchers+ discovered 25 out of the 40 imbalanced pairs, while the next best systems (AML and LogMap) dis-

covered only 11 and 9 pairs respectively. On the NELL-DBpedia, KGMatcher+ found 14 pairs while the closest system is OTMapOnto which discovered 10 pairs. The results also show that some matching systems only discovered trivial matches, e.g., LSmatch, FCAMap-KG, TOM, and Fine-TOM. The latest two found only two to one imbalanced class pair in the YAGO-Wikidata dataset.

5.4.4 The Impact of Different Matching Component

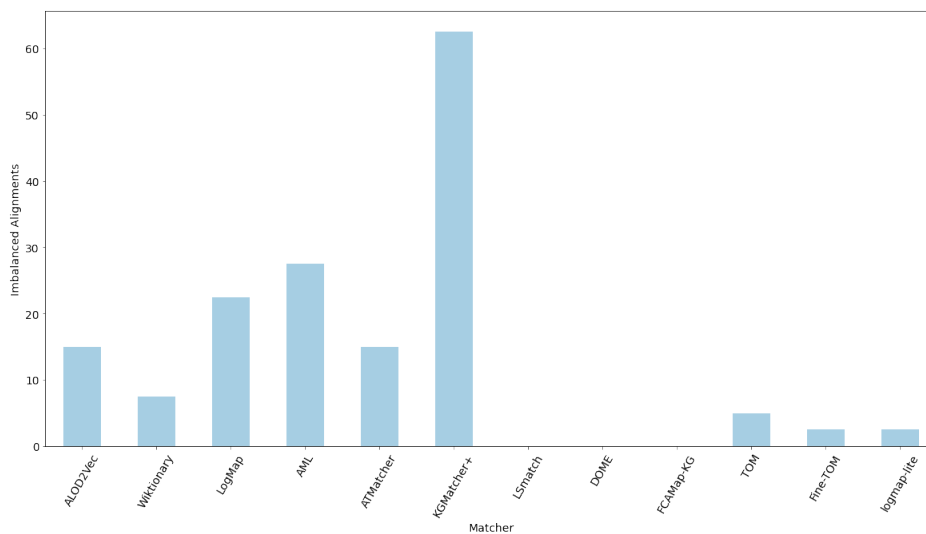
In this section, we study the impact of different matching components of KGMatcher+: the resampling component, the matcher combination, and the number of words used in TF-IDF undersampling (i.e., k). We perform these experiments on the Common KG track of OAEI as the OAEI KGs do not provide a large enough benchmark for an instance-based method.

5.4.4.1 The Impact of Resampling KG Instances

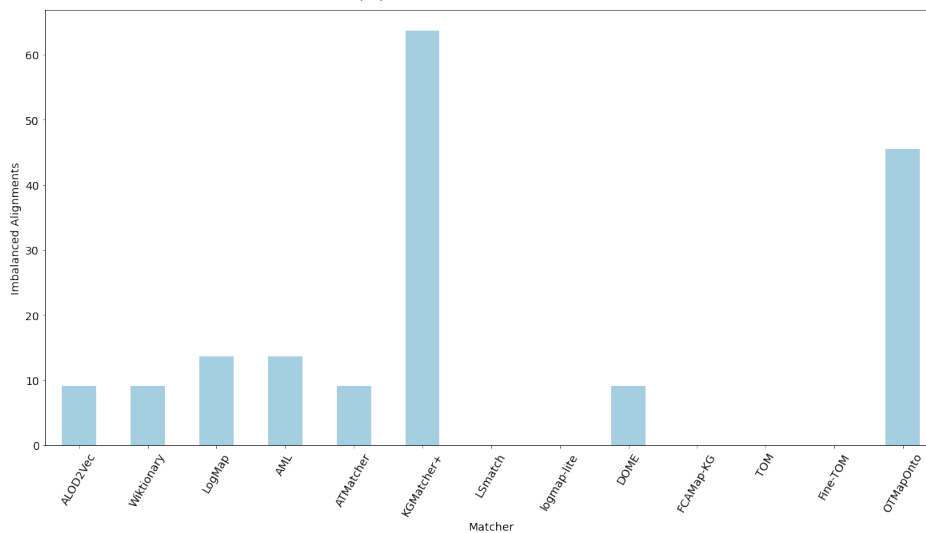
As part of training KG classifiers, we incorporated different data balancing strategies in order to cope with the imbalanced class distribution issue (in Section 4.3.1). In the following, we discuss the impact of these strategies on the accuracy of matching large-scale KGs classes. We implement the 6 strategies as part of the architecture of KGMatcher+ as depicted in Figure 5.1. When each strategy is used instead of others, we denote that version of KGMatcher+ as KGMatcher+{SS}, where SS indicates the corresponding Sampling Strategy.

Table 5.4 shows the precision, recall, and f-measure of KGMatcher+ when using different sampling strategies. As the table shows, in terms of f-measure, KGMatcher+{TF-IDF + oversampling} outperforms all other variations with (f-measure=0.91) on the YAGO-Wikidata dataset and (f-measure=0.95) on the NELL-DBpedia dataset. In terms of undersampling strategies, KGMatcher+ {Random Undersampling} fails to improve the overall results on both datasets compared to the results obtained when no sampling was applied. On the other hand, while KGMatcher+{TF-IDF undersampling} does leave the matching results on both datasets unchanged, compared to no sampling, it maintains the same performance while significantly decreasing the matcher processing time. E.g., from 55 minutes to 29 minutes on the NELL-DBpedia dataset and from 3 hours to 1.5 on the YAGO-Wikidata.

Two different undersampling strategies were implemented, random undersampling and using TF/IDF. The results show a significant gap in the impact of the both strate-



(a) YAGO-Wikidata



(b) NELL-DBpedia

Figure 5.3: The percentage of imbalanced class pairs discovered by different methods compared to KGMatcher+

Table 5.4: Evaluation results of different variations of KGMatcher+ each utilizing a different data balancing strategy (the best results are in **bold**).

Strategy	YAGO-Wikidata			NELL-DBpedia		
	Precision	Recall	F-Measure	Precision	Recall	F-Measure
No Sampling	0.97	0.79	0.87	0.97	0.91	0.94
Random Undersampling	0.98	0.75	0.85	0.98	0.81	0.89
TF-IDF Undersampling	0.96	0.80	0.87	0.97	0.91	0.94
TF-IDF + Oversampling	0.99	0.84	0.91	0.98	0.91	0.95
SMOTE	0.97	0.78	0.86	0.98	0.89	0.93
TF-IDF + SMOTE	0.99	0.80	0.88	0.97	0.91	0.94
Cost-based learning	0.96	0.77	0.85	0.96	0.85	0.90

gies on KGMatcher+ performance. The first method, i.e., random undersampling, includes a random elimination of KG instances. Further, with this strategy, instance samples are reduced to match the size of the class with the least samples, which can be less than 10 instances in some common KGs. This rather aggressive reduction in training data could have a detrimental impact on classifier training. As a result, KGMatcher+{random undersampling} shows the lowest accuracy results. In contrast, using TF-IDF to down sample classes in KGMatcher+{TF-IDF undersampling} does not negatively impact the results, as the elimination process maintains instances with indicative words.

In terms of oversampling, we implemented the state-of-the-art approach known as SMOTE, which includes generating syntactic samples in the minority classes. The results reveal that KGMatcher+{SMOTE} decreases the recall on both datasets, which subsequently affects the f-measure score as well. Then, in terms of combining undersampling with oversampling, we combined the TF/IDF approach with two oversampling methods. Although KGMatcher+ {TF-IDF + SMOTE} shows results similar to the best-performing strategy on the NELL-DBpedia dataset, i.e., KGMatcher+{TF-IDF + Oversampling}, it does not perform as well on the YAGO-Wikidata dataset. This is due to the latter dataset being twice the size of NELL-DBpedia which also impacts the severity of the imbalance population problem. Even though class distribution in the KGs used in the experiments was severe, undersampling the majority classes with TF-IDF helps mitigate this issue. However, generating synthetic data from KG

instances seems to introduce noisy samples to the dataset, as indicated by the result of $\text{KGMatcher}+\{\text{SMOTE}\}$. This seems to be consistent with previously reported findings in text classification tasks (Padurariu & Breaban 2019).

The final data balancing strategy is $\text{KGMatcher}+\{\text{Cost-based learning}\}$, which adapts the BERT model to handle class imbalance by using class weights. Although the main advantage of this method is to maintain the integrity of the datasets, this did not work well as it achieved the worst precision, recall, and f-measure, which are even lower than the model using no data balancing strategies at all. This is due to over-penalizing the classifier for incorrectly classifying instances from the minority classes.

5.4.4.2 The Impact of Matcher Combination

$\text{KGMatcher}+$ is a hybrid matching approach that combines the ability of an element level matcher to discover KG classes with terminological/semantic similarity with an instance-based method. The latter is able to align KG classes with overlapping real-world entities, even when their classes have different names. Our experiments included performing only the name matcher component of $\text{KGMatcher}+$ compared to the hybrid approach that is $\text{KGMatcher}+$. Figure 5.4 shows the precision, recall, and f-measure for each experiment. Overall, the hybrid approach achieves better results than only using the name matcher. On the YAGO-Wikidata dataset, combining instance-based and name matcher archives an f-measure of 0.91 which is higher than using the name matcher (0.85 f-measure). Similarly, on the NELL-DBpedia dataset, the hybrid approach archives 0.95 f-measure compared to the name matcher f-measure (0.89). We can deduce that adopting a hybrid approach that incorporates an instance-based method is beneficial, particularly for matching common KGs classes. Due to the dataset size, and given that the instance-based matcher uses an exact name filter to reduce the search space, we are not able to compare the results to the instance-based matcher only. However, combining both matchers significantly improve the overall results, as the instance-matcher was able to discover additional alignments. Comparing the name matcher component to the full $\text{KGMatcher}+$ method, the latter notably improves the recall on both datasets. The recall on the YAGO-Wikidata has improved by 10% (from 0.74 to 0.84), while the recall on the NELL-DBpedia dataset was improved by 11% (from 0.80 to 0.91). While combining the two matchers increased the results, it managed to maintain a high precision between 0.99 on YAGO-Wikidata and 0.98 on NELL-DBpedia.

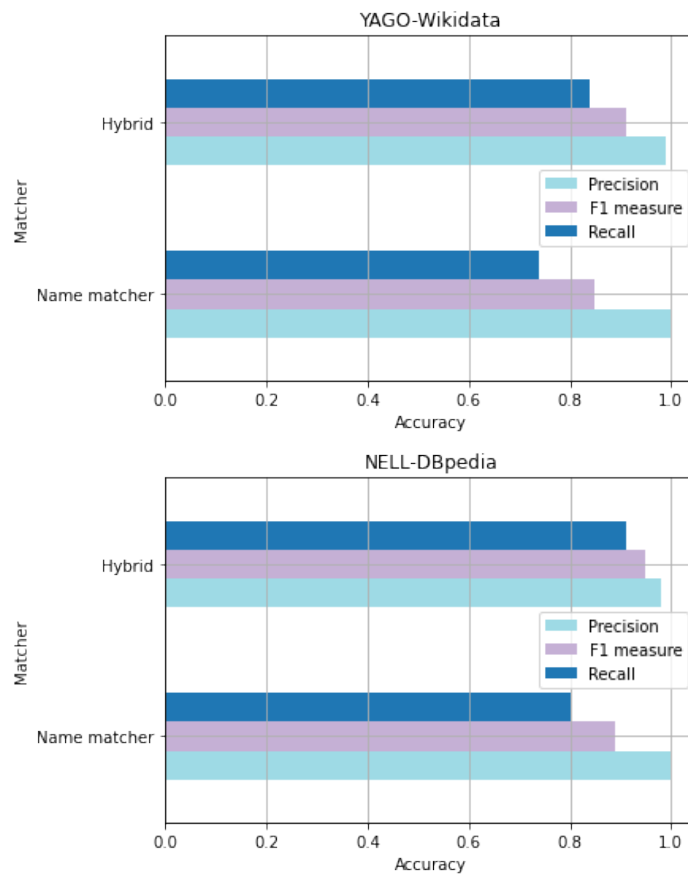


Figure 5.4: Comparing the results of the name matcher component of KGMatcher+ against the full KGMatcher+ method.

Table 5.5: The impact of K value on KGMatcher+

Dataset	K	Precision	Recall	F-Measure	Time
NELL-DBpedia	$K = 10$	1.00	0.91	0.95	02:09:26
	$K = 15$	0.98	0.91	0.94	02:12:04
	$K = 20$	0.98	0.92	0.95	02:20:45
YAGO-Wikidata	$K = 10$	0.99	0.84	0.91	03:45:22
	$K = 15$	0.99	0.82	0.90	04:03:47
	$K = 20$	0.99	0.83	0.90	04:32:03

5.4.4.3 The Impact of the TF/IDF Threshold

Whenever the TF/IDF undersampling technique is applied within KGMatcher+, it requires a value we named as K , discussed earlier in Chapter 4 (Section 4.3.1.2). This method generates a ranked list of words according to their TF/IDF score within a certain KG class, i.e., a majority class. Then, the K value is used to re-sample that class' instances by discarding instances names that do not compose any of the top K words. Table 5.5 compares different values, e.g., 10, 15, and 20 in terms of their impact on the overall matching results and on the run time. The results show that increasing K to higher than 10 leads to decreasing the overall performance on both datasets. This is also accompanied by an increase in the KGMatcher+ running time that varies between the two datasets due to the YAGO-Wikidata being double the size of the NELL-DBpedia. It is also worth mentioning that while this value slightly impacts the matcher results, our method still outperforms all OAEI systems, despite what K value is used, as the previous section presented.

5.5 Conclusion

In this chapter, we proposed KGMatcher+ which has a component that only uses KG instances to generate class alignments. Our method is domain-independent and, to the best of our knowledge, includes the first instance-based matcher for matching KG classes with the ability to handle unbalanced class populations. We presented the evaluation of our method compared to the state-of-the-art methods using the two most representative OAEI matching tracks, i.e., the Common KG track and the KG track. KGMatcher+, outperformed existing systems on the task of matching common KGs and ranked as one of the top methods for the KG track. From the results presented

above, we can conclude that mapping the schema of large and common KGs is far from a trivial task and needs to be handled differently from conventional ontology matching. We have also shown that utilizing an instance-based approach has a significant impact on matching large KGs with an abundance of annotated instances. Furthermore, we have presented an analysis that compares the impact of different sampling strategies incorporated in KGMatcher+. The results have also confirmed that combining under-sampling and oversampling is the best strategy for handling the problem with both majority and minority classes. Besides the impact of resampling, we have also studied the effect of other components of KGMatcher+ such as the name matcher, showing that a ‘hybrid’ approach yields a better matching performance.

Related Publications

- Fallatah, Omaima., Zhang, Ziqi., & Hopfgartner, Frank. A hybrid approach for large knowledge graphs matching. *In Proceedings of the 16th International Workshop on Ontology Matching (OM 2021) co-located with the 20th International Semantic Web Conference (ISWC 2021)* (Vol. 3063, pp. 37-48). CEUR-WS.
- Fallatah, Omaima., Zhang, Ziqi., & Hopfgartner, Frank. The Impact of Imbalanced Class Distribution on Knowledge Graphs Matching. *In Proceedings of the 17th International Workshop on Ontology Matching (OM 2022) co-located with the 21st International Semantic Web Conference (ISWC 2022)*. CEUR-WS.
- Fallatah, Omaima., Zhang, Ziqi., & Hopfgartner, Frank. KGMatcher Results for OAEI 2021. *In Proceedings of the 16th International Workshop on Ontology Matching (OM 2021) co-located with the 20th International Semantic Web Conference (ISWC 2021)* (Vol. 3063, pp. 160-166). CEUR-WS.

Chapter 6

Conclusion and Future Work

In this chapter, we summarize the research presented in the thesis and give some concluding remarks. First, we start by presenting the summary of the contributions delivered by our research in Section 6.1. Then, in Section 6.2, we provide a discussion of the remaining and open issues to be further investigated and considered in future work.

6.1 Summary of the Thesis Contributions

KGs, particularly general-purpose and cross-domain ones, have been increasingly used in different downstream applications. The last decade of research in the ontology matching task has resulted in various matching methods and techniques being developed. While KGs are also means of data representation, they can be very different from conventional well-formed ontologies. Typical cross-domain KGs shared on the web are often large-scale, with hundreds of classes and millions of instances. Therefore, our research was primarily focused on studying the problem of matching the classes in KGs with such characteristics.

We started by studying different matching techniques used to discover semantically similar entities in heterogeneous ontologies in Chapter 2. Then, we further investigated the current state of matching KGs and the techniques and tools that have been introduced for a relatively new task. The literature review has concluded with three main findings: (1) current systems still suffer from scalability issues, (2) they typically make use of the characteristics that are only found in well-formed ontologies, and (3) prior research is mainly domain-specific as the pre-existing gold standard datasets are

predominantly domain-dependent such as those in the OAEI biomedical tasks. This analysis is the first contribution of this thesis, and it assisted us to address our three research questions stated in Section 1.4 as follows.

RQ1: *What are the current KG matching benchmarks, and how can we construct one that is more representative of the large KG matching problem?*

Concerning the lack of diverse and real-world-based datasets, we proposed two gold-standard datasets in Chapter 3, which counts as the second contribution of this thesis. The two benchmarks are based on four KGs with high influence in the semantic web domain, namely, DBpedia, NELL, Wikidata, and YAGO. Further, they are the largest existing benchmarks for the task of matching classes in large-scale KGs. Both datasets have been added to the annual OAEI evaluation events as a new matching track known as the *Common Knowledge Graphs* track, which aims at evaluating matching systems on this particular task. This track has been running since 2021 and both datasets are now largely used as part of research in this area.

RQ2: *Given the large yet unbalanced number of instances in KG classes, how can we make use of them effectively in learning?*

Combining different matching techniques is a common strategy in the majority of state-of-the-art matching systems. However, most of the existing approaches depend on terminological and structural methods compared to instance-based ones due to the nature of typical ontologies – they lack the large number of instances needed for such methods. Nonetheless, large and public KGs are known for having an abundance of instances, which make them ideal for an instance-based method. Given the number of instances shared in typical KGs, measuring the overlap of instances can be a challenging task. In order to overcome this limitation, we proposed a method of training KG instance classifiers in Chapter 4. Those classifiers were built using state-of-the-art language modeling techniques that have shown exceptional results in many text classification tasks. Furthermore, we carried out a study of the imbalanced population issue in common KGs and its impact on the instance classification task. Our solutions for such a problem revealed a significant improvement in the classifier results. The approach of training KG instance classifiers is the third contribution in this thesis.

RQ3 *How can we effectively use instances to match KG classes while addressing the unbalanced population issue?*

We integrated the KG instance classifier as part of our KG matching method. We have proved that a hybrid matching approach with an instance-based matcher

outperforms existing state-of-the-art matching methods. Furthermore, we evaluated our prototype called KGMatcher+ on seven matching benchmarks from two different KG tracks from OAEI 2022 and showcased that using an instance-based method can discover more correct alignments. Additionally, we analyzed the impact of six different data balancing techniques, including two methods that we introduce in this work, on matching large-scale and imbalanced KG classes. Then, we studied the impact of the matcher components of our hybrid method. We showed that using an instance-based method improves recall and f-measure compared to only using an element-level approach. Finally, we studied the impact of the threshold value used by our sampling method. We concluded that increasing the number of words used for undersampling has very little influence on the result, but can impact the matcher’s processing time. This is promising since it shows that KGMatcher+ is capable of generating satisfactory results without further parameter tuning, which has practical benefits. The hybrid KG matching method is our fourth contribution.

6.2 Open Issues and Future Work

Despite the fact that we have delivered a solution for the challenging task of matching large-scale and common KGs, some open issues remain to be considered. In the following, we present some current limitations and areas of opportunity to expand the work in this thesis.

Matcher Combination.

The current matcher combination in KGMatcher+ utilizes both an element-level matcher and an instance-based matcher. While this approach has its advantages, it can potentially limit the system’s performance when dealing with tasks that specifically target domain-specific KGs. This limitation arises from the fact that such tasks often require domain-specific knowledge that may not be adequately covered by general pre-trained word embeddings and language models. Therefore, expanding the current matcher combination could be done by exploring background knowledge resources in order to discover additional matching. Further, another future challenge can be to automatically decide the matchers’ pipeline of our method. The current matching pipeline is built with the assumption that the two input KGs share semantically similar class names and instances, which is typical in common KGs. However, in other cases where the input KGs are from a certain domain where such an assumption does not apply, the

pipeline can be changed accordingly. For example, we can study the distributions of instances across the source and target KG classes and then use the statistical analysis to inform the hybrid matcher whether the instance-based or the background knowledge matcher should be included or not. Another way to decide the matcher combination is to explore supervised machine learning approaches for this task (Hertling et al. 2020, Laadhar et al. 2017). However, this method can bring more challenges as it requires training data to decide the appropriate matcher combination.

Instance Matching. One of the limitations of KGMatcher+ is that it currently includes only a basic instance-level matcher. For the OAEI participation, we have adapted KGMatcher+ to also match the instances of KGs. The current instance matching component generates candidate instance pairs based on the existence of the label in the opposite KG. However, it is important to note that this approach relies solely on terminological matching and may not capture the full semantics and context of the instances. Labels alone may not provide sufficient information for accurate instance matching, especially in cases where instances have similar labels but represent different real-world entities. This component can be further improved in future work by exploring additional techniques. For example, given that KGMatcher+ produces competitive results in terms of class matching, and with common KGs sharing a large number of complementary instances, class correspondences can be used to verify or discover further instance-level alignments. This can be done by matching the schema first and using the same string-based method to match instances. Then, using the results produced by the class matching component to increase the similarity value of instance pairs with ‘matched’ classes (Portisch & Paulheim 2022a, Hertling & Paulheim 2020a). Another approach to explore is to reciprocally match both levels, where class matching results are utilized by the instance matching component. Then, the latter can also be used to revise the class matching results until convergence (Suchanek et al. 2012). However, to the best of our knowledge, a benchmark for matching common KG instances is yet to be built. Thus, the two proposed datasets can be expanded to cover the task of matching common KGs instances in addition to their schemas.

User Interaction. While fully automating the matching process is an important aspect of such a task, in some real-world applications the user’s input and validation of the matching results can be necessary. This is particularly critical in matching tasks in the medical and business domains, which are witnessing a growth of KG applications. As a result, explainability has been an ongoing challenge in ontology matching sys-

tems (Hertling et al. 2019, Portisch et al. 2019). Therefore, a graphical user interface combined with proper visualization techniques can be adapted to allow users to further understand the process of deriving the matches. Further, beyond the confidence value of each correspondence, users can be given explanations of the matching results, such as the specific matching component that has generated a particular alignment.

Multi-source Matching. Current matching approaches and datasets are predominantly focused on pairwise matching. However, recent studies have highlighted the need to move towards multi-source matching, including KGs (Ayala et al. 2021, Primpeli & Bizer 2022). Multi-source matching is when the matching is performed across more than two KGs simultaneously. With the growth of KGs and their applications, some scenarios may require such an approach. For instance, constructing a large KG from distinct KGs by matching them prior to the integration process (Hertling & Paulheim 2022). Thus, a long-term research direction is to develop an approach that performs multi-source KG matching.

Bibliography

- Altinel, B. & Ganiz, M. C. (2018), ‘Semantic text classification: A survey of past and recent advances’, Information Processing & Management **54**(6), 1129–1153.
- Altman, N. S. (1992), ‘An introduction to kernel and nearest-neighbor nonparametric regression’, The American Statistician **46**(3), 175–185.
- An, Y., Kalinowski, A. & Greenberg, J. (2021), Otmappointo: optimal transport-based ontology matching., in ‘Proceedings of the 16th International Workshop on Ontology Matching co-located with the 20th International Semantic Web Conference (ISWC 2021)’, Vol. 3063, pp. 185–192.
- Anam, S., Kim, Y. S., Kang, B. H. & Liu, Q. (2015), ‘Review of ontology matching approaches and challenges’, International Journal of Computer Science and Network Solutions **3**(3), 1–27.
- Ardjani, F., Bouchiha, D. & Malki, M. (2015), ‘Ontology-alignment techniques: Survey and analysis.’, International Journal of Modern Education & Computer Science **7**(11).
- Arnold, P. & Rahm, E. (2014), ‘Enriching ontology mappings with semantic relations’, Data & Knowledge Engineering **93**, 1–18.
- Aumueller, D., Do, H.-H., Massmann, S. & Rahm, E. (2005), Schema and ontology matching with coma++, in ‘Proceedings of the 2005 ACM SIGMOD international conference on Management of data’, pp. 906–908.
- Ayala, D., Hernández, I., Ruiz, D. & Rahm, E. (2021), Towards the smart use of embedding and instance features for property matching, in ‘IEEE 37th International Conference on Data Engineering (ICDE2021)’, IEEE, pp. 2111–2116.

- Banerjee, M., Capozzoli, M., McSweeney, L. & Sinha, D. (1999), ‘Beyond kappa: A review of interrater agreement measures’, Canadian Journal of Statistics **27**(1), 3–23.
- Berners-Lee, T., Hendler, J. & Lassila, O. (2001), ‘The semantic web’, Scientific American **284**(5), 34–43.
- Bizer, C., Primpeli, A. & Peeters, R. (2019), ‘Using the semantic web as a source of training data’, Datenbank-Spektrum **19**(2), 127–135.
- Bojanowski, P., Grave, E., Joulin, A. & Mikolov, T. (2017), ‘Enriching word vectors with subword information’, Transactions of the association for computational linguistics **5**, 135–146.
- Bollacker, K., Evans, C., Paritosh, P., Sturge, T. & Taylor, J. (2008), Freebase: a collaboratively created graph database for structuring human knowledge, in ‘Proceedings of the 2008 ACM SIGMOD international conference on Management of data’, pp. 1247–1250.
- Bunkhumpornpat, C., Sinapiromsaran, K. & Lursinsap, C. (2009), Safe-level-smote: Safe-level-synthetic minority over-sampling technique for handling the class imbalanced problem, in ‘Pacific-Asia conference on knowledge discovery and data mining’, Springer, pp. 475–482.
- Carlson, A., Betteridge, J., Kisiel, B., Settles, B., Hruschka, E. R. & Mitchell, T. M. (2010), Toward an architecture for never-ending language learning, in ‘Twenty-Fourth AAAI conference on artificial intelligence’.
- Castano, S., Ferrara, A., Montanelli, S. & Varese, G. (2011), Ontology and instance matching, in ‘Knowledge-driven multimedia information extraction and ontology evolution’, Springer, pp. 167–195.
- Chang, F., Chen, G. & Zhang, S. (2019), Fcamap-kg results for oaei 2019, in ‘Proceedings of the 14th International Workshop on Ontology Matching co-located with the 18th International Semantic Web Conference (ISWC2019)’, Vol. 2536, pp. 138–145.
- Chawla, N. V., Bowyer, K. W., Hall, L. O. & Kegelmeyer, W. P. (2002), ‘Smote: synthetic minority over-sampling technique’, Journal of artificial intelligence research **16**, 321–357.

- Cheatham, M. & Hitzler, P. (2013), String similarity metrics for ontology alignment, in ‘Proceedings of the International Semantic Web Conference (ISWC2013)’, Springer, pp. 294–309.
- Clarkson, K., Gentile, A. L., Gruhl, D., Ristoski, P., Terdiman, J. & Welch, S. (2018), User-centric ontology population, in ‘The 15th European Semantic Web Conference (ESWC 2018)’, Springer, pp. 112–127.
- Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K. & Kuksa, P. (2011), ‘Natural language processing (almost) from scratch’, Journal of machine learning research **12**, 2493–2537.
- Cruz, I. F., Antonelli, F. P. & Stroe, C. (2009), ‘Agreementmaker: efficient matching for large real-world schemas and ontologies’, Proceedings of the VLDB Endowment **2**(2), 1586–1589.
- Cruz, I. F. & Sunna, W. (2008), ‘Structural alignment methods with applications to geospatial ontologies’, Transactions in GIS **12**(6), 683–711.
- Dai, Y., Wang, S., Xiong, N. N. & Guo, W. (2020), ‘A survey on knowledge graph embedding: Approaches, applications and benchmarks’, Electronics **9**(5), 750.
- Diallo, G. (2014), ‘An effective method of large scale ontology matching’, Journal of Biomedical Semantics **5**(1), 1–19.
- Djeddi, W. E. & Khadir, M. T. (2010), Xmap: a novel structural approach for alignment of owl-full ontologies, in ‘The International Conference on Machine and Web Intelligence 2010’, IEEE, pp. 368–373.
- Do, H.-H. & Rahm, E. (2002), Coma—a system for flexible combination of schema matching approaches, in ‘Proceedings of the 28th International Conference on Very Large Databases’, pp. 610–621.
- Doan, A., Madhavan, J., Domingos, P. & Halevy, A. (2004), Ontology matching: A machine learning approach, in ‘Handbook on ontologies’, Springer, pp. 385–403.
- Ehrlinger, L. & Wöß, W. (2016), ‘Towards a definition of knowledge graphs’, SEMANTiCS (Posters, Demos, SuCCESS) **48**(1-4), 2.

- Elkan, C. (2001), The foundations of cost-sensitive learning, in ‘International joint conference on artificial intelligence’, Vol. 17, Lawrence Erlbaum Associates Ltd, pp. 973–978.
- Erxleben, F., Günther, M., Krötzsch, M., Mendez, J. & Vrandečić, D. (2014), Introducing wikidata to the linked data web, in ‘The 13th International Semantic Web Conference (ISWC 2014)’, Springer, pp. 50–65.
- Euzenat, J., Meilicke, C., Stuckenschmidt, H., Shvaiko, P. & Trojahn, C. (2011), Ontology alignment evaluation initiative: six years of experience, in ‘Journal on Data Semantics’, Vol. 15, Springer, pp. 158–192.
- Euzenat, J., Shvaiko, P. et al. (2007), Ontology matching, Vol. 18.
- Faria, C., Serra, I. & Girardi, R. (2014), ‘A domain-independent process for automatic ontology population from text’, Science of Computer Programming **95**, 26–43.
- Faria, D., Pesquita, C., Santos, E., Palmonari, M., Cruz, I. F. & Couto, F. M. (2013), The agreementmakerlight ontology matching system, in ‘OTM Confederated International Conferences On the Move to Meaningful Internet Systems’, Springer, pp. 527–541.
- Faria, D., Pesquita, C., Tervo, T., Couto, F. M. & Cruz, I. F. (2019), Aml and amlc results for oaei 2019, in ‘Proceedings of the 14th International Workshop on Ontology Matching co-located with the 18th International Semantic Web Conference (ISWC2019)’, Vol. 2536, pp. 101–106.
- Faria, D., Silva, M. C., Cotovio, P., Eugénio, P. & Pesquita, C. (2022), ‘Matcha and matcha-dl results for oaei 2022’.
- Fayed, H. A. & Atiya, A. F. (2009), ‘A novel template reduction approach for the k -nearest neighbor method’, IEEE Transactions on Neural Networks **20**(5), 890–896.
- Feng, H., Qin, W., Wang, H., Li, Y. & Hu, G. (2021), A combination of resampling and ensemble method for text classification on imbalanced data, in ‘International Conference on Big Data’, Springer, pp. 3–16.

- Fu, Z., Xian, Y., Gao, R., Zhao, J., Huang, Q., Ge, Y., Xu, S., Geng, S., Shah, C., Zhang, Y. et al. (2020), Fairness-aware explainable recommendation over knowledge graphs, in ‘Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval’, pp. 69–78.
- Giunchiglia, F., Autayeu, A. & Pane, J. (2012), ‘S-match: an open source framework for matching lightweight ontologies’, Semantic Web Journal **3**(3), 307–317.
- Glazkova, A. (2020), ‘A comparison of synthetic oversampling methods for multi-class text classification’, arXiv preprint arXiv:2008.04636 .
- Gruetze, T., Böhm, C. & Naumann, F. (2012), ‘Holistic and scalable ontology alignment for linked open data’, WWW conference 2012 Workshop on Linked Data on the Web, LDOW **937**, 1–10.
- Gulić, M., Vrdoljak, B. & Banek, M. (2016), ‘Cromatcher: An ontology matching system based on automated weighted aggregation and iterative final alignment’, Journal of Web Semantics **41**, 50–71.
- Gulić, M., Vrdoljak, B. & Vuković, M. (2018), ‘An iterative automatic final alignment method in the ontology matching system’, Journal of Information and Organizational Sciences **42**(1), 39–61.
- Han, H., Wang, W.-Y. & Mao, B.-H. (2005), Borderline-smote: a new over-sampling method in imbalanced data sets learning, in ‘International conference on intelligent computing’, Springer, pp. 878–887.
- He, H., Bai, Y., Garcia, E. A. & Li, S. (2008), Adasyn: Adaptive synthetic sampling approach for imbalanced learning, in ‘IEEE international joint conference on neural networks (IEEE world congress on computational intelligence 2008)’, IEEE, pp. 1322–1328.
- Heist, N., Hertling, S., Ringler, D. & Paulheim, H. (2020), ‘Knowledge graphs on the web – an overview’, Tiddi, I., Lécué, F., Hitzler, P. (eds.) Knowledge Graphs for eXplainable Artificial Intelligence: Foundations, Applications and Challenges, Studies on the Semantic Web **47**.

- Hertling, S. & Paulheim, H. (2017), Webisalod: providing hypernym relations extracted from the web as linked open data, in ‘The Semantic Web - ISWC 2017 - 16th International Semantic Web Conference’, Vol. 10588, Springer, pp. 111–119.
- Hertling, S. & Paulheim, H. (2018), Dbkwik: A consolidated knowledge graph from thousands of wikis, in ‘IEEE International Conference on Big Knowledge (ICBK2018)’, IEEE, pp. 17–24.
- Hertling, S. & Paulheim, H. (2019), Dome results for oaei 2019, in ‘Proceedings of the 14th International Workshop on Ontology Matching co-located with the 18th International Semantic Web Conference (ISWC2019)’, Vol. 2536, pp. 123–130.
- Hertling, S. & Paulheim, H. (2020a), Atbox results for oaei 2020, in ‘Proceedings of the 15th International Workshop on Ontology Matching (OM 2020), co-located with the 19th International Semantic Web Conference (ISWC 2020)’, Vol. 2788, CEUR-WS, pp. 168–175.
- Hertling, S. & Paulheim, H. (2020b), The knowledge graph track at oaei, in ‘The Semantic Web - the 17th International Conference, ESWC 2020’, Vol. 12123, Springer, pp. 343–359.
- Hertling, S. & Paulheim, H. (2022), ‘Gollum: A gold standard for large scale multi source knowledge graph matching’, arXiv preprint arXiv:2209.07479 .
- Hertling, S., Portisch, J. & Paulheim, H. (2019), Melt-matching evaluation toolkit, in ‘Proceedings of The Power of AI and Knowledge Graphs - 15th International Conference, SEMANTiCS 2019’, Vol. 11702, Springer, pp. 231–245.
- Hertling, S., Portisch, J. & Paulheim, H. (2020), Supervised ontology and instance matching with melt, in ‘Proceedings of the 15th International Workshop on Ontology Matching co-located with the 19th International Semantic Web Conference (ISWC 2020)’, Vol. 2788, pp. 60–71.
- Hu, F., Shao, Z. & Ruan, T. (2014), ‘Self-supervised chinese ontology learning from online encyclopedias’, The scientific world journal **2014**.
- Hu, W., Jian, N., Qu, Y. & Wang, Y. (2005), Gmo: A graph matching for ontologies, in ‘Proceedings of K-CAP Workshop on Integrating Ontologies’, pp. 41–48.

- Hu, W. & Qu, Y. (2008), ‘Falcon-ao: A practical ontology matching system’, Journal of Web Semantics **6**(3), 237–239.
- Ichise, R. (2008), Machine learning approach for ontology mapping using multiple concept similarity measures, in ‘Seventh IEEE/ACIS International Conference on Computer and Information Science (ICIS 2008)’, IEEE, pp. 340–346.
- Jain, P., Hitzler, P., Sheth, A. P., Verma, K. & Yeh, P. Z. (2010), Ontology alignment for linked open data, in ‘Proceedings of the 9th International Semantic Web Conference (ISWC2010)’, Springer, pp. 402–417.
- Jean-Mary, Y. R., Shironoshita, E. P. & Kabuka, M. R. (2009), ‘Ontology matching with semantic verification’, Journal of Web Semantics **7**(3), 235–251.
- Ji, G., Liu, K., He, S. & Zhao, J. (2016), Knowledge graph completion with adaptive sparse transfer matrix, in ‘Thirtieth AAAI conference on artificial intelligence’.
- Jiménez-Ruiz, E. (2020), Logmap family participation in the oaei 2020, in ‘Proceedings of the 15th International Workshop on Ontology Matching (OM 2020), co-located with the 19th International Semantic Web Conference (ISWC 2020)’, Vol. 2788, CEUR-WS, pp. 201–203.
- Jiménez-Ruiz, E. (2021), Logmap family participation in the oaei 2021, in ‘Proceedings of the 16th International Workshop on Ontology Matching co-located with the 20th International Semantic Web Conference (ISWC 2021)’, Vol. 3063, pp. 175–177.
- Jiménez-Ruiz, E. & Cuenca Grau, B. (2011), Logmap: Logic-based and scalable ontology matching, in ‘Proceedings of the 10th International Semantic Web Conference (ISWC 2011)’, Springer, pp. 273–288.
- Joachims, T. (1996), A probabilistic analysis of the rocchio algorithm with tfidf for text categorization., Technical report, Carnegie-mellon univ pittsburgh pa dept of computer science.
- Jurafsky, D. & Martin, J. (2018), ‘Wordnet: Word relations senses and disambiguation’, Speech and Language Processing .
- Kachroudi, M., Moussa, E. B., Zghal, S. & Ben, S. (2011), ‘Ldoa results for oaei 2011’, Proceedings of the 6th International Workshop on ontology matching, co-located with the 10th International Semantic Web Conference (ISWC 2011) **814**, 148.

- Khurana, A. & Verma, O. P. (2022), ‘Optimal feature selection for imbalanced text classification’, IEEE Transactions on Artificial Intelligence .
- King, G. & Zeng, L. (2001), ‘Logistic regression in rare events data’, Political analysis **9**(2), 137–163.
- Kirsten, T., Thor, A. & Rahm, E. (2007), Instance-based matching of large life science ontologies, in ‘International Conference on Data Integration in the Life Sciences’, Springer, pp. 172–187.
- Knorr, L. & Portisch, J. (2022), Fine-tom matcher results for oaei 2021, in ‘Proceedings of the 16th International Workshop on Ontology Matching co-located with the 20th International Semantic Web Conference (ISWC 2021)’, Vol. 3063, pp. 144–151.
- Kolouri, S., Park, S. R., Thorpe, M., Slepcev, D. & Rohde, G. K. (2017), ‘Optimal mass transport: Signal processing and machine-learning applications’, IEEE signal processing magazine **34**(4), 43–59.
- Kossack, D., Borg, N., Knorr, L. & Portisch, J. (2022), Tom matcher results for oaei 2021, in ‘Proceedings of the 16th International Workshop on Ontology Matching co-located with the 20th International Semantic Web Conference (ISWC 2021)’, Vol. 3063, pp. 193–198.
- Krauss, P. (2017), ‘schemaorg-wikidata-map’, <https://github.com/okfn-brasil/schemaOrg-Wikidata-Map>.
- Krawczyk, B. (2016), ‘Learning from imbalanced data: open challenges and future directions’, Progress in Artificial Intelligence **5**(4), 221–232.
- Kubat, M., Matwin, S. et al. (1997), Addressing the curse of imbalanced training sets: one-sided selection, in ‘Icml’, Vol. 97, Nashville, USA, p. 179.
- Laadhar, A., Ghozzi, F., Bousarsar, I. M., Ravat, F., Teste, O. & Gargouri, F. (2017), Pomap: An effective pairwise ontology matching system, in ‘IC3K 2017: 9th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management’, pp. 161–168.
- Laadhar, A., Ghozzi, F., Megdiche, I., Ravat, F., Teste, O. & Gargouri, F. (2019), The impact of imbalanced training data on local matching learning of ontologies, in ‘International Conference on Business Information Systems’, Springer, pp. 162–175.

- Lambrix, P. & Tan, H. (2006), ‘Sambo—a system for aligning and merging biomedical ontologies’, Journal of Web Semantics **4**(3), 196–206.
- Lecue, F. (2020), ‘On the role of knowledge graphs in explainable ai’, The Semantic Web Journal **11**(1), 41–51.
- Lehmann, J., Isele, R., Jakob, M., Jentzsch, A., Kontokostas, D., Mendes, P. N., Hellmann, S., Morsey, M., Van Kleef, P., Auer, S. et al. (2015), ‘Dbpedia—a large-scale, multilingual knowledge base extracted from wikipedia’, Semantic web **6**(2), 167–195.
- Lemaître, G., Nogueira, F. & Aridas, C. K. (2017), ‘Imbalanced-learn: A python toolbox to tackle the curse of imbalanced datasets in machine learning’, The Journal of Machine Learning Research **18**(1), 559–563.
- Levenshtein, V. I. et al. (1966), Binary codes capable of correcting deletions, insertions, and reversals, in ‘Soviet Physics Doklady’, Vol. 10, Soviet Union, pp. 707–710.
- Ling, X. & Weld, D. S. (2012), Fine-grained entity recognition, in ‘Twenty-Sixth AAAI Conference on Artificial Intelligence’.
- Liu, X.-Y., Wu, J. & Zhou, Z.-H. (2008), ‘Exploratory undersampling for class-imbalance learning’, IEEE Transactions on Systems and Cybernetics (Cybernetics) **39**(2), 539–550.
- Liu, X.-Y. & Zhou, Z.-H. (2006), The influence of class imbalance on cost-sensitive learning: An empirical study, in ‘Sixth International Conference on Data Mining (ICDM’06)’, IEEE, pp. 970–974.
- Liu, Y., Loh, H. T. & Sun, A. (2009), ‘Imbalanced text classification: A term weighting approach’, Expert systems with Applications **36**(1), 690–701.
- Liu, Z., Cao, W., Gao, Z., Bian, J., Chen, H., Chang, Y. & Liu, T.-Y. (2020), Self-paced ensemble for highly imbalanced massive data classification, in ‘IEEE 36th international conference on data engineering (ICDE2020)’, IEEE, pp. 841–852.
- Lu, W., Li, Z. & Chu, J. (2017), ‘Adaptive ensemble undersampling-boost: a novel learning framework for imbalanced data’, Journal of systems and software **132**, 272–282.

- Lubani, M., Noah, S. A. M. & Mahmud, R. (2019), ‘Ontology population: Approaches and design aspects’, Journal of Information Science **45**(4), 502–515.
- Ma, L. & Zhang, Y. (2015), Using word2vec to process big text data, in ‘2015 IEEE International Conference on Big Data (Big Data)’, IEEE, pp. 2895–2897.
- Madhavan, J., Bernstein, P. A. & Rahm, E. (2001), Generic schema matching with cupid, in ‘proceedings of the 27th International Conference of Very Large Data Bases(VLDB)’, Vol. 1, pp. 48–58.
- Maiya, A. S. (2020), ‘ktrain: A low-code library for augmented machine learning’, arXiv preprint arXiv:2004.10703 .
- Malmasi, S., Fang, A., Fetahu, B., Kar, S. & Rokhlenko, O. (2022), ‘Multiconer: a large-scale multilingual dataset for complex named entity recognition’, arXiv preprint arXiv:2208.14536 .
- Mao, M., Peng, Y. & Spring, M. (2010), ‘An adaptive ontology mapping approach with neural network based constraint satisfaction’, Journal of Web semantics **8**(1), 14–25.
- Marie, A. & Gal, A. (2008), Boosting schema matchers, in ‘OTM Confederated International Conferences On the Move to Meaningful Internet Systems’, Springer, pp. 283–300.
- Martinez-Gil, J., Navas-Delgado, I. & Aldana-Montes, J. F. (2012), ‘Maf: An ontology matching framework’, Journal of Universal Computer Science **18**(2), 194–217.
- Mascardi, V., Locoro, A. & Rosso, P. (2009), ‘Automatic ontology matching via upper ontologies: A systematic evaluation’, IEEE Transactions on knowledge and data engineering **22**(5), 609–623.
- Megdiche, I., Teste, O. & Trojahn, C. (2016), An extensible linear approach for holistic ontology matching, in ‘The 15th International Semantic Web Conference (ISWC 2016)’, Springer, pp. 393–410.
- Meusel, R., Bizer, C. & Paulheim, H. (2015), A web-scale study of the adoption and evolution of the schema.org vocabulary over time, in ‘Proceedings of the 5th International Conference on Web Intelligence, Mining and Semantics’, pp. 1–11.

- Meusel, R., Petrovski, P. & Bizer, C. (2014), The webdatacommons microdata, rdfa and microformat dataset series, in ‘Proceedings of the 13th International Semantic Web Conference (ISWC2014)’, Vol. 8797, Springer, pp. 277–292.
- Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S. & Dean, J. (2013), ‘Distributed representations of words and phrases and their compositionality’, Advances in neural information processing systems **26**, 3111–3119.
- Miller, G. A. (1995), ‘Wordnet: a lexical database for english’, Communications of the ACM **38**(11), 39–41.
- Minaee, S., Kalchbrenner, N., Cambria, E., Nikzad, N., Chenaghlu, M. & Gao, J. (2021), ‘Deep learning–based text classification: a comprehensive review’, ACM Computing Surveys (CSUR) **54**(3), 1–40.
- Mohammadi, M., Atashin, A. A., Hofman, W. & Tan, Y. (2018), ‘Comparison of ontology alignment systems across single matching task via the mcnemar’s test’, ACM Transactions on Knowledge Discovery from Data (TKDD) **12**(4), 1–18.
- Mohit, B. (2014), Named entity recognition, in ‘Natural Language Processing of Semitic Languages’, Springer, pp. 221–245.
- Nezhadi, A. H., Shadgar, B. & Osareh, A. (2011), ‘Ontology alignment using machine learning techniques’, International Journal of Computer Science & Information Technology **3**.
- Ngo, D., Bellahsene, Z. & Coletta, R. (2011a), A flexible system for ontology matching, in ‘International Conference on Advanced Information Systems Engineering’, Springer, pp. 79–94.
- Ngo, D., Bellahsene, Z. & Coletta, R. (2011b), A generic approach for combining linguistic and context profile metrics in ontology matching, in ‘OTM Confederated International Conferences On the Move to Meaningful Internet Systems’, Springer, pp. 800–807.
- Ngo, D. H. (2012), Enhancing ontology matching by using machine learning, graph matching and information retrieval techniques, PhD thesis, Université Montpellier II-Sciences et Techniques du Languedoc.

- Ngo, D. H. & Bellahsene, Z. (2012), Yam++:(not) yet another matcher for ontology matching task, in ‘BDA: Bases de Données Avancées’.
- Nguyen, T. T. A. & Conrad, S. (2013), A semantic similarity measure between nouns based on the structure of wordnet, in ‘Proceedings of International Conference on Information Integration and Web-based Applications & Services’, pp. 605–609.
- Nguyen, T. T. A. & Conrad, S. (2015), Ontology matching using multiple similarity measures, in ‘The 7th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management (IC3K 2015)’, Vol. 1, IEEE, pp. 603–611.
- Nkisi-Orji, I., Wiratunga, N., Massie, S., Hui, K.-Y. & Heaven, R. (2018), Ontology alignment based on word embedding and random forest classification, in ‘Joint European Conference on Machine Learning and Knowledge Discovery in Databases’, Springer, pp. 557–572.
- Obraczka, D., Schuchart, J. & Rahm, E. (2021), ‘Eager: embedding-assisted entity resolution for knowledge graphs’, arXiv preprint arXiv:2101.06126 .
- Ochieng, P. & Kyanda, S. (2018), ‘Large-scale ontology matching: State-of-the-art analysis’, ACM Computing Surveys (CSUR) **51**(4), 1–35.
- Ossewaarde, R. (2007), Simple library thesaurus alignment with silas, in ‘Proceedings of the 2nd International Workshop on Ontology Matching (OM-2007) Collocated with the 6th International Semantic Web Conference (ISWC-2007)’, Vol. 304, CEUR-WS, pp. 255–260.
- Otero-Cerdeira, L., Rodríguez-Martínez, F. J. & Gómez-Rodríguez, A. (2015), ‘Ontology matching: A literature review’, Expert Systems with Applications **42**(2), 949–971.
- Padurariu, C. & Breaban, M. E. (2019), ‘Dealing with data imbalance in text classification’, Procedia Computer Science **159**, 736–745.
- Papadakis, G., Skoutas, D., Thanos, E. & Palpanas, T. (2020), ‘Blocking and filtering techniques for entity resolution: A survey’, ACM Computing Surveys (CSUR) **53**(2), 1–42.

- Paulheim, H. (2017), ‘Knowledge graph refinement: A survey of approaches and evaluation methods’, *Semantic web* **8**(3), 489–508.
- Peeters, R., Bizer, C. & Glavaš, G. (2020), ‘Intermediate training of bert for product matching’, *small* **745**(722), 2–112.
- Pellissier Tanon, T., Weikum, G. & Suchanek, F. (2020), Yago 4: A reason-able knowledge base, in ‘The Semantic Web - the 17th International Conference, ESWC 2020’, Vol. 12123, Springer, pp. 583–596.
- Pennington, J., Socher, R. & Manning, C. D. (2014), Glove: Global vectors for word representation, in ‘Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)’, pp. 1532–1543.
- Portisch, J., Hladik, M. & Paulheim, H. (2019), Wiktionary matcher, in ‘Proceedings of the 13th ISWC workshop on Ontology Matching (OM), co-located with the 18th International Semantic Web Conference (ISWC 2019)’, Vol. 2536, pp. 181–188.
- Portisch, J., Hladik, M. & Paulheim, H. (2021), ‘Background knowledge in ontology matching: A survey’, *Semantic web* .
- Portisch, J. & Paulheim, H. (2018), ‘Alod2vec matcher’, *Proceedings of the 12th ISWC workshop on Ontology Matching (OM) co-located with the 18th International Semantic Web Conference (ISWC 2018)* **2288**, 132–137.
- Portisch, J. & Paulheim, H. (2022a), Alod2vec matcher results for oaei 2021, in ‘Proceedings of the 16th International Workshop on Ontology Matching co-located with the 20th International Semantic Web Conference (ISWC2021)’, Vol. 3063, RWTH, pp. 117–123.
- Portisch, J. & Paulheim, H. (2022b), Wiktionary matcher results for oaei 2021, in ‘Proceedings of the 16th International Workshop on Ontology Matching co-located with the 20th International Semantic Web Conference (ISWC2021)’, Vol. 3063, CEUR-WS, pp. 199–206.
- Pour, M. A. N., Algergawy, A., Amardeilh, F., Amini, R., Fallatah, O., Faria, D., Fundulaki, I., Harrow, I., Hertling, S., Hitzler, P., Huschka, M., Ibanescu, L., Jiménez-Ruiz, E., Karam, N., Laadhar, A., Lambrix, P., Li, H., Li, Y., Michel, F., Nasr, E., Paulheim, H., Pesquita, C., Portisch, J., Roussey, C., Tzania, S., Splendiani, A.,

- Tro-jahn, C., Vatašcinová, J., Yaman, B., Zamazal, O. & Zhou, L. (2021), Results of the ontology alignment evaluation initiative 2021, in ‘Proceedings of the 16th International Workshop on ontology matching, co-located with the 20th International Semantic Web Conference (ISWC 2021)’, Vol. 3063, CEUR, pp. 62–108.
- Pour, M., Algergawy, A., Buche, P., Castro, L. J., Chen, J., Dong, H., Fallatah, O., Faria, D., Fundulaki, I., Hertling, S., He, Y., Horrocks, I., Huschka, M., Ibanescu, L., Jiménez-Ruiz, E., Karam, N., Laadhar, A., Lambrix, P., Li, H., Li, Y., Michel, F., Nasr, E., Paulheim, H., Pesquita, C., Tzania, S., Shvaiko, P., Trojahn, C., Verhey, C., Wu, M., Yaman, B., Zamazal, O. & Zhou, L. (2022), Results of the ontology alignment evaluation initiative 2022, in ‘Proceedings of the 17th International Workshop on ontology matching, co-located with the 21st International Semantic Web Conference (ISWC 2022)’, CEUR.
- Primpeli, A. & Bizer, C. (2022), Impact of the characteristics of multi-source entity matching tasks on the performance of active learning methods, in ‘The 19th European Semantic Web Conference (ESWC 2022)’, Springer, pp. 113–129.
- Primpeli, A., Peeters, R. & Bizer, C. (2019), The wdc training dataset and gold standard for large-scale product matching, in ‘Companion Proceedings of The 2019 World Wide Web Conference’, pp. 381–386.
- Pujara, J., Miao, H., Getoor, L. & Cohen, W. (2013), Knowledge graph identification, in ‘Proceedings of the 12th International Semantic Web Conference (ISWC 2013)’, Vol. 1111, CEUR-WS, pp. 542–557.
- Raghavan, V. V. & Wong, S. M. (1986), ‘A critical analysis of vector space model for information retrieval’, Journal of the American Society for information Science **37**(5), 279–287.
- Rahm, E. (2011), Towards large-scale schema and ontology matching, in ‘Schema Matching and Mapping’, Springer, pp. 3–27.
- Rahm, E. & Peukert, E. (2019), ‘Large-scale schema matching’.
- Ramadhana, N. H., Darari, F., Putra, P. O. H., Nutt, W., Razniewski, S. & Akbar, R. I. (2020), User-centered design for knowledge imbalance analysis: A case study of prowd, in ‘Proceedings of the Visualization and Interaction for Ontologies and Linked

- Data workshop (VOILA2020) co-located with the 19th International Semantic Web Conference (ISWC 2020)', pp. 14–27.
- Ramos, J. et al. (2003), Using tf-idf to determine word relevance in document queries, in 'Proceedings of the first instructional conference on machine learning', Vol. 242, Citeseer, pp. 29–48.
- Ren, X., He, W., Qu, M., Huang, L., Ji, H. & Han, J. (2016), Afet: Automatic fine-grained entity typing by hierarchical partial-label embedding, in 'Proceedings of the 2016 conference on empirical methods in natural language processing', pp. 1369–1378.
- Reul, Q. & Pan, J. Z. (2010), Kosimap: Use of description logic reasoning to align heterogeneous ontologies, in '23rd International Workshop on Description Logics DL2010', Vol. 489, p. 16.
- Ringler, D. & Paulheim, H. (2017), One knowledge graph to rule them all? analyzing the differences between dbpedia, yago, wikidata & co., in 'Joint German/Austrian Conference on Artificial Intelligence (Künstliche Intelligenz)', Springer, pp. 366–372.
- Ristoski, P., Faralli, S., Ponzetto, S. P. & Paulheim, H. (2017), Large-scale taxonomy induction using entity and word embeddings, in 'Proceedings of the International Conference on Web Intelligence', pp. 81–87.
- Ritze, D., Lehmberg, O., Oulabi, Y. & Bizer, C. (2016), Profiling the potential of web tables for augmenting cross-domain knowledge bases, in 'Proceedings of the 25th international conference on world wide web', pp. 251–261.
- Sahare, M. & Gupta, H. (2012), 'A review of multi-class classification for imbalanced data', International Journal of Advanced Computer Research **2**(3), 160.
- Sánchez-Ruiz, A. A., Ontanón, S., González-Calero, P. A. & Plaza, E. (2011), Measuring similarity in description logics using refinement operators, in 'International Conference on Case-Based Reasoning', Springer, pp. 289–303.
- Seddiqui, M. H. & Aono, M. (2009), Anchor-flood: results for oaei 2009, in 'Proceedings of the ISWC 2009 Workshop on ontology matching', Vol. 551, pp. 127–134.

- Seo, S., Oh, B., Jo, E., Lee, S., Lee, D., Lee, K.-H., Shin, D. & Lee, Y. (2021), ‘Active learning for knowledge graph schema expansion’, IEEE Transactions on Knowledge and Data Engineering .
- Sharma, A., Patel, A. & Jain, S. (2021), Lsmatch results for oaei 2021, in ‘Proceedings of the 16th International Workshop on Ontology Matching co-located with the 20th International Semantic Web Conference (ISWC2021)’, Vol. 3063, pp. 178–184.
- Shvaiko, P. & Euzenat, J. (2005), A survey of schema-based matching approaches, in ‘Journal on data semantics’, Vol. 4, Springer, pp. 146–171.
- Shvaiko, P. & Euzenat, J. (2011), ‘Ontology matching: state of the art and future challenges’, IEEE Transactions on Knowledge and Data Engineering **25**(1), 158–176.
- Sikdar, U. K., Barik, B. & Gambäck, B. (2018), Named entity recognition on code-switched data using conditional random fields, in ‘Proceedings of the Third Workshop on Computational Approaches to Linguistic Code-Switching’, pp. 115–119.
- Suchanek, F. M., Abiteboul, S. & Senellart, P. (2012), Paris: Probabilistic alignment of relations, instances, and schema, in ‘Proceedings of the VLDB Endowment’, Vol. 5, p. 157–168.
- Suchanek, F. M., Kasneci, G. & Weikum, G. (2007), Yago: a core of semantic knowledge, in ‘Proceedings of the 16th international conference on World Wide Web’, pp. 697–706.
- Suhara, Y., Li, J., Li, Y., Zhang, D., Demiralp, Ç., Chen, C. & Tan, W.-C. (2022), Annotating columns with pre-trained language models, in ‘Proceedings of the 2022 International Conference on Management of Data’, pp. 1493–1503.
- Sunna, W. & Cruz, I. F. (2007), Using the agreementmaker to align ontologies for the oaei campaign 2007, in ‘Proceedings of the 2nd International Workshop on ontology matching, co-located with the 6th International Semantic Web Conference (ISWC 2007)’, Vol. 304.
- Tan, P.-N., Steinbach, M. & Kumar, V. (2006), Introduction to data mining, Pearson Education India.

- Thada, V. & Jaglan, V. (2013), ‘Comparison of jaccard, dice, cosine similarity coefficient to find best fitness value for web retrieved documents using genetic algorithm’, International Journal of Innovations in Engineering and Technology **2**(4), 202–205.
- Thor, A., Kirsten, T. & Rahm, E. (2007), ‘Instance-based matching of hierarchical ontologies’, Proceedings of the 12th German Database Conference (BTW 2007) .
- Tomek, I. (1976), ‘Two modifications of cnn.’, IEEE Trans. Systems, Man and Cybernetics .
- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. & Polosukhin, I. (2017), ‘Attention is all you need’, Advances in neural information processing systems **30**.
- Wang, S., Englebienne, G. & Schlobach, S. (2008), Learning concept mappings from instance similarity, in ‘Proceedings of the 7th International Semantic Web Conference (ISWC 2008)’, Springer, pp. 339–355.
- Xiao, C., Wang, W. & Lin, X. (2008), ‘Ed-join: an efficient algorithm for similarity joins with edit distance constraints’, Proceedings of the VLDB Endowment **1**(1), 933–944.
- Xue, X. & Pan, J.-S. (2018), ‘Evolutionary algorithm based ontology matching technique’, Journal of Information Hiding and Multimedia Signal Processing **9**, 75–88.
- Yogatama, D., Gillick, D. & Lazic, N. (2015), Embedding methods for fine grained entity type classification, in ‘Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing’, pp. 291–296.
- Yosef, M. A., Bauer, S., Hoffart, J., Spaniol, M. & Weikum, G. (2012), Hyena: Hierarchical type classification for entity names, in ‘Proceedings of COLING 2012: Posters’, pp. 1361–1370.
- Zafar, B., Cochez, M. & Qamar, U. (2016), Using distributional semantics for automatic taxonomy induction, in ‘2016 International Conference on Frontiers of Information Technology (FIT)’, IEEE, pp. 348–353.
- Zhang, Y., Jin, H., Pan, L. & Li, J.-Z. (2016), Rimom results for oaei 2016, in ‘Proceedings of the 11th International Workshop on Ontology Matching co-located with the 15th International Semantic Web Conference (ISWC 2016)’, Vol. 1766, pp. 210–216.

- Zhang, Z., Gentile, A. L., Blomqvist, E., Augenstein, I. & Ciravegna, F. (2017), ‘An unsupervised data-driven method to discover equivalent relations in large linked datasets’, The Semantic Web Journal **8**(2), 197–223.
- Zhao, M. & Zhang, S. (2016), Fca-map results for oaei 2016, in ‘Proceedings of the 11th International Workshop on Ontology Matching co-located with the 15th International Semantic Web Conference (ISWC 2016)’, Vol. 1766, pp. 172–177.
- Zhu, G. & Iglesias, C. A. (2016), ‘Computing semantic similarity of concepts in knowledge graphs’, IEEE Transactions on Knowledge and Data Engineering **29**(1), 72–85.

Appendices

Appendix A

Annotation Task Instructions

Thank you for participating in our survey. This task is about finding corresponding (i, e., similar) classes in two different yet complementary knowledge graphs. A class in this context can be considered as a category of real-world entities. For instance, **Hotel**, **Airport**, **Magazine**, **Movie**, and **Artist** are all examples of a knowledge graph class. Classes can have different names in different knowledge graph, for example: class **Movie** could be referred to as **Film** in another knowledge graph, and **ReligiousBuilding** could be named as **PlaceforWorship**.

The goal of here is to annotate pairs of classes in the dataset. The Excel sheet sent to you has seven columns where the first three columns (**Class_Name.1**, **URI.1**, **Instances_Names.1**) describing a class in the first knowledge graph. While the following three columns describe a class in the second knowledge graph. Please see table A.1 below for an example. The final column is the **Relation** column where you will be adding one of the following values for annotation:

- Write (1) if you believe that Class 1 and Class 2 are equivalent (referring to the same thing)
- Write (0) if you believe that Class 1 and Class 2 are not equivalent (referring to completely different category)
- Write (*) if you believe that Class 1 is more general than Class 2: e.g., **Building** is more general than **Hotel**, **Person** is more general than **Artist**, and **Place** is more general than **Beach**
- Write (#) if you believe that Class 2 is more general than Class 1.

Class_Name.1	URI.1	Instances_Names.1	Class_Name.2	URI.2	Instances_Names.2	Relation
academicfield	http://rtw.ml.cmu.edu/rtw/kbbrowser/preacademicfield	science management neurotrauma bioclimatology biometeorology	AcademicConference	http://dbpedia.org/ontology/AcademicConference	European Conference on OOP Vision (festival) European Symposium on Programming	

Table A.1: Annotation Example

In order to decide what relation to add, you can: (1) use the link in the URI columns to read more the description of each class, and (2) use the information in the Instances.Name columns. These are examples of real-world entities that belong to each class. Note: The data in this column can be random/incorrect or missing in some cases, because they are extracted automatically from different resources without human involvement. In this case, you can decide the relation based on the class name, URIs and your background knowledge.