Liage de données ouvertes et hétérogènes : application au domaine musical

Manel Achichi

THESE POUR OBTENIR LE GRADE DE DOCTEUR DE L'UNIVERSITE DE MONTPELLIER

En Informatique

École doctorale Information, Structure, Systèmes (I2S)

Unité de recherche Laboratoire d'Informatique, Robotique et Micro-électronique de Montpellier (LIRMM)

Linking Heterogeneous Open Data
- Application to the Musical Domain -

Présentée par **Manel ACHICHI**

Le 15/02/2018

Devant le jury composé de

Mathieu d'Aquin, Professeur, Université de Galway, Irlande	**Rapporteur**
Catherine Faron Zucker, Maître de Conférences (HDR), Université de Nice Sophia Antipolis	**Rapporteur**
Ollivier Haemmerlé, Professeur, Université Toulouse	**Président du jury**
Zohra Bellahsene, Professeur, Université Montpellier	**Directrice**
Konstantin Todorov, Maitre de conférences, Université Montpellier	**Co-Encadrant**

A mes très chers parents pour leur amour inconditionnel
A mon cher mari Ibrahim d'avoir toujours cru en moi
A mon frère et ma sœur à qui je dois cette réussite

Remerciements

En premier lieu, je remercie vivement Monsieur **Mathieu d'Aquin**, Professeur au National University d'Ireland (Galway) et à Madame **Catherine Faron Zucker**, Maître de Conférences à l'Université de de Nice Sophia Antipolis pour avoir accepté de rapporter sur ma thèse, ainsi qu'à Monsieur Ollivier Haemmerlé, Professeur à l'Université Toulouse pour avoir accepté de faire partie du jury.

Je tiens particulièrement à exprimer mes plus vifs remerciements à ma directrice de thèse **Zohra BELLAHSENE**, Professeur à l'Université de Montpellier, pour son soutien, sa gentillesse, ses nombreux conseils avisés tout au long de ce travail. Sa patience, sa disponibilité et surtout sa grande expérience n'ont jamais cessé de m'impressionner.

Je voudrai également remercier Monsieur **Konstantin Todorov**, Maître de Conférences à l'Université de Montpellier, pour la confiance qu'il m'a toujours accordée, pour son soutien, sa grande patience et ses conseils tout au long de ces années de travail. J'ai énormément appris à ses côtés.

Être membre du projet DOREMUS a été une expérience formidable pour moi. Je tiens à remercier tous les membres du projet pour l'excellent travail que nous avons fait ensemble. Ce projet m'a donné l'occasion de rencontrer des gens formidables, d'échanger des idées et de discuter; un processus qui m'a donné de l'enthousiasme et de la motivation pour ce travail.

Je ne serai pas arrivée là où je suis aujourd'hui si ce n'est grâce à mes parents, ma sœur et mon frère qui ont ancré en moi l'amour des études, qui ont trouvé leur place dans ce travail. Grâce à eux, je suis tributaire d'une éducation dont je suis fière. Pour ce, je ne cesserai jamais de les remercier.

Je ne peux oublier le reste de ma famille et en particulier la famille ZURLI qui

s'est beaucoup sacrifiée pour me voir toujours meilleure. J'espère qu'elle trouvera dans ce travail toute ma reconnaissance.

Last but not least, Ibrahim, mon mari, mon ami et mon épaule, merci d'avoir toujours cru en moi, d'avoir toujours été présent dans les bons comme dans les mauvais moments. On ne sait jamais ce que l'avenir nous réserve. Ce qui est sûr c'est qu'on mènera le chemin ensemble mais pour l'instant faisons ce que la vie nous a tant appris: *let's live the moment*.

Abstract

To a great extent, the growing requirement for interoperability between different information systems required the use of Semantic Web technologies to facilitate the data exchange. Indeed, the data in the "classical" Web are intended to be read and understood by humans. One of the most important aims of the Semantic Web project is to structure these data so that they can be understood by software agents. Many data models have been developed, particularly the RDF (Resource Description Framework) model. As suggests its name, RDF allows describing the resources on the web. A resource defines any real world entity that can be referred and described. It can refer to a photo, a video, a web page, a monument, an event or a person. Thus, different data providers can describe in the same way, differently or with complementary information the same entity. Given the massive growth of data published on the web, it becomes difficult to rapidly access the description of a given entity. From there, was developed the web of data, an initiative of W3C (World Wide Web Consortium), which consists of structuring, linking and sharing data described in RDF graphs. A semantic link can express any relationship between two resources. In this thesis, we are interested in the relations, called *identity links*, expressing an equivalence between two different resources describing the same entity of the real world. Formally and in the web of data, an *identity link* is expressed by an `owl:sameAs` statement.

This thesis is part of the ANR DOREMUS[1] -DOing REusable MUSical data- project. We are interested in the catalogs of three cultural institutions, including BNF (Bibliothèque Nationale de France), Philharmonie de Paris and Radio France, containing detailed descriptions about music works, such as their composers, their performances, their interpretations and the related events (concerts, recordings, etc). However, these data are stored in different formats and cannot be directly exchanged between the three institutions. Within the framework of the DOREMUS project, these institutions have adopted the Semantic Web technologies with the aim of making these data accessible to all and linked. Structuring metadata

[1] `http://www.doremus.org/`

on entities requires the use of a vocabulary that models a particular domain. The first task in this project was to model the semantics of knowledge. We were particularly interested in classical music. The selection of an existing vocabulary was a complicated process. Indeed, the project pays particular attention to the musical works which constitute the main entity held by the three partners. One of the main objectives of the DOREMUS project is to be able to express, through a model, all the physical manifestations (recordings, partitions) of a musical work, as well as all the events that define them (creation, publication and interpretations).

The creation of identity links is a non-trivial task and becomes particularly difficult considering the high heterogeneity between the descriptions of the same entity. In this thesis, our main objective is to propose a generic data linking approach, dealing with certain challenges, for a concrete application on DOREMUS data. Particularly, we focus on three major challenges: (1) reducing the tool configuration effort, (2) coping with different kinds of data heterogeneities across datasets and (3) dealing with datasets containing blocks of highly similar instances. Recently, many approaches for linking data from different domains have been proposed. They implemented various linking strategies. However, some of them often require the user intervention during the linking process to configure some parameters or to validate or to decline some generated links. This may be a costly task for the user who may not be an expert in the domain. Therefore, one of the research questions that arises is how to reduce human intervention as much as possible in the process of data linking. Moreover, the data can show various heterogeneities that a linking tool has to deal with. The descriptions can be expressed in different natural languages, with different vocabularies or with different values. Indeed, the lack of directives on how to structure and to describe resources can greatly impact the complexity of the matching decision. The comparison can be complicated due to the variations according to three dimensions: *value*-based, *ontological* and *logical*. In this thesis, we analyze the most recurrent aspects of heterogeneity by identifying a set of techniques that can be applied to them. Another challenge is the distinction between highly similar but not equivalent resource descriptions in the same dataset. In their presence, most of the existing tools are reduced in efficiency generating false positive matches. In this perspective, some approaches have been proposed to identify a set of discriminative properties called keys. Very often, such approaches discover a very large number of keys. The question that arises is whether all keys can discover the same pairs of equivalent instances, or if some are more meaningful than others. To our knowledge, no approach provides a strategy to classify the keys generated according to their effectiveness to discover the correct links. In order to ensure quality alignments, we have proposed in this work a new data linking approach addressing the challenges described above.

We developed *Legato* — a generic tool for automatic heterogeneous data linking. Our tool is based on instance profiling to represent each resource as a textual document of literals dealing with a variety of data heterogeneities. *Legato* implements a filtering step of so-called problematic properties allowing to clean the data of the noise likely to make the comparison task difficult. To address the problem of similar but distinct resources, *Legato* implements a key ranking algorithm called RANKey. The effectiveness of *Legato* has been validated on different datasets. Indeed, our system has participated in OAEI (Ontology Alignment Evaluation Initiative) 2017, where it has been successful on the data benchmark. It was the best system on datasets of highly similar instances with a high link accuracy of 100%. In addition, the results showed that *Legato* produces the best alignments in terms of accuracy thanks to its repairing strategy.

Titre en Français: Liage de Données Ouvertes et Hétérogènes -Application au Domaine Musical-.

Résumé

Dans une large mesure, l'exigence croissante d'interopérabilité entre les différents systèmes de stockage de données a nécessité l'adoption du Web Sémantique pour faciliter l'échange d'informations. En effet, les données qui sont stockées dans le web "classique" de documents ont été conçues pour être lues et comprises par les humains. Un des objectifs les plus importants du Web Sémantique est de structurer ces données de manière à ce qu'elles puissent être comprises par les agents logiciels. Pourtant, XML (eXtensible Markup Language), un langage communément utilisé, fournit une syntaxe pour créer, structurer et manipuler des données. Cependant, XML ne permet pas de modéliser la sémantique des données produites indépendamment les unes des autres. Pour pallier à ce problème, plusieurs modèles de données ont été développés dont particulièrement le modèle RDF (Resource Description Framework). Comme son nom l'indique, RDF permet de décrire des ressources sur le web. Celles-ci définissent toute entité, du monde réel, concrète ou aussi abstraite soit-elle pouvant être référée et décrite. Ces ressources peuvent référer des photos, des vidéos, des pages web, des monuments, des événements ou encore des personnes. Ainsi, différents fournisseurs de données peuvent décrire pareillement, différemment ou avec des informations complémentaires une même entité. Au vu de la croissance fulgurante de données publiées sur le web, se pose la difficulté d'accéder rapidement et en une seule requête à la description d'une entité donnée. De là, a été conçu le web de données, une initiative du W3C (World Wide Web Consortium), qui consiste à structurer, lier et partager des données décrites en graphe RDF. Un lien sémantique peut exprimer toute relation entre deux ressources. Dans cette thèse, nous nous intéressons aux relations, appelées *liens d'identité*, exprimant une équivalence entre deux ressources différentes mais qui décrivent la même entité du monde réel. Formellement et en web de données, un *lien d'identité* est exprimé par un lien `owl:sameAs`.

Des milliers d'œuvres musicales sont décrites dans des catalogues des institutions culturelles, dont le rôle est de stocker toutes les créations musicales à travers le catalogage et de les diffuser auprès du grand public. Cette thèse s'inscrit dans le cadre du projet ANR DOREMUS[2] -DOnnées en REutilisation pour la Musique en fonction des USages- qui vise à explorer les métadonnées des catalogues de trois grandes institutions culturelles : Bibliothèque Nationale de France (BNF), Philharmonie de Paris et Radio France afin qu'elles puissent communiquer entre elles et être mieux utilisées par les différents publics. Elles ne représentent pas toutes les données musicales de la même manière. En effet, les données sont stockées dans différents formats et ne peuvent être directement échangées entre les trois institutions. Dans le cadre du projet DOREMUS, ces institutions ont alors adopté les techniques et principes du Web Sémantique dont l'objectif est de rendre ces données ouvertes à tous, accessibles et liées. La création des liens d'identité s'avère être une tâche non triviale et devient particulièrement difficile considérant la grande hétérogénéité entre les descriptions d'une même entité. Dans cette thèse, notre objectif principal est de proposer une approche de liage générique, traitant certains challenges, avec comme cas concret d'utilisation les données de DOREMUS.

Structurer les métadonnées sur des entités nécessite l'utilisation d'un vocabulaire qui modélise un domaine particulier. La première tâche dans le projet DOREMUS a été de modéliser la sémantique de l'ensemble de connaissances. Nous nous sommes particulièrement intéressés à la musique classique. La sélection d'un vocabulaire existant fut un processus compliqué. En effet, le projet accorde une attention particulière aux œuvres musicales qui constituent la principale entité détenue par les trois partenaires. L'un des objectifs de DOREMUS est de pouvoir exprimer, via un modèle, toutes les manifestations physiques (enregistrements, partitions), d'une œuvre musicale, ainsi que tous les événements qui la définissent (création, publication, interprétations). Plusieurs modèles existants peuvent être réutilisés pour décrire une œuvre musicale, i.e., MusicOntology, alors que d'autres sont conçus pour les documents bibliographiques, i.e., FRBR[3]. Il existe également des modèles qui visent à décrire des événements (Event[4], LODE[5]), mais sont rares ceux qui définissent les processus de création et de publication. Par conséquent, le groupe de modélisation a développé un nouveau modèle basé sur FRBRoo[6] qui est lui-même une extension du modèle CIDOC-CRM[7]. Ce modèle consiste en une

[2] http://www.doremus.org/
[3] https://www.ifla.org/publications/functional-requirements-for-bibliographic-records
[4] http://motools.sourceforge.net/event/event.html
[5] http://linkedevents.org/ontology/
[6] http://www.cidoc-crm.org/frbroo/
[7] http://www.cidoc-crm.org/

nouvelle ontologie musicale permettant de prendre en compte la description des œuvres et des événements qui les concernent.

Le processus de transformation de données en RDF présente donc un travail préliminaire réalisé dans le cadre du projet DOREMUS. Les principales sources d'informations consistent en des documents au format MARC (Bibliothèque Nationale de France et la Philharmonie de Paris) ainsi que des documents au format XML (Radio France). Le format MARC (MAchine Readable Cataloging) est un format d'échange de données bibliographiques permettant d'informatiser les catalogues de bibliothèques. La conversion des données en RDF se basait essentiellement sur des règles de mapping écrites manuellement par les experts du groupe de modélisation. En effet, il était devenu crucial pour le groupe de modélisation de fournir un tableau de mapping expliquant la localisation de chaque propriété par rapport au format utilisé par chacune des institutions. Ces règles varient considérablement en fonction de la nature des données en entrée. Dans ce projet, j'ai participé au développement d'un outil, appelé MARC2RDF[8], de transformation de données venant de la BNF et de la Philharmonie de Paris. Comme son nom l'indique, cet outil prend en entrée des notices au format MARC (UNIMARC/INTERMARC) pour produire des données en RDF. MARC2RDF est donc basé sur le modèle DOREMUS dans lequel les trois entités "œuvre", "expression" et "événement" sont mises en relation tel qu'un événement crée une œuvre qui est réalisée dans une expression.

Une fois que la structuration des données en RDF effectuée, l'étape suivante consiste à établir des liens d'identité entre les œuvres musicales. Dans cette thèse, nous nous focalisons sur trois principaux challenges : (1) réduire la configuration manuelle de l'outil de liage, (2) faire face à différents types d'hétérogénéité entre les descriptions, et (3) Supprimer l'ambiguïté entre les ressources très similaires dans leur descriptions mais qui ne sont pas équivalentes. Récemment, des approches ont été proposées en déployant différentes stratégies pour faire face au problème de liage de données. Cependant, certaines d'entre elles demandent souvent l'intervention de l'utilisateur pour configurer certains paramètres ou encore pour valider ou décliner certains liens générés par le système. Ceci peut s'avérer être une tâche coûteuse pour l'utilisateur qui peut ne pas être expert du domaine. Par conséquent, une des questions de recherche que nous nous posons est comment réduire autant que possible l'intervention humaine dans le processus de liage des données. De plus, en fonction des sources, les descriptions des ressources peuvent présenter diverses hétérogénéités qu'un outil doit savoir gérer. Par ailleurs, les descriptions peuvent être exprimées dans différentes langues naturelles, avec des vocabulaires différents ou encore avec des valeurs différentes. En effet, l'absence de

[8] `https://github.com/DOREMUS-ANR/marc2rdf`

directives sur la manière de structurer et de décrire les ressources peut grandement impacter la complexité de la décision d'appariement. La comparaison peut alors s'avérer très difficile en raison des variations selon trois dimensions : basées sur les valeurs, ontologiques et logiques. Dans cette thèse, nous analysons les aspects d'hétérogénéité les plus récurrents en identifiant un ensemble de techniques qui peuvent leur être appliquées. Un autre défi est la distinction entre des descriptions de ressources fortement similaires mais non équivalentes dans un même jeu de données. En leur présence, la plupart des outils existants se voient diminuer leur efficacité en terme de qualité, en générant beaucoup de faux positifs. Dans cette optique, certaines approches ont été proposées pour identifier un ensemble de propriétés discriminatives appelées des clefs. Très souvent, de telles approches découvrent un très grand nombre de clés. La question qui se pose est de savoir si toutes les clés permettent de découvrir les mêmes paires d'instances équivalentes, ou si certaines sont plus significatives que d'autres. A notre connaissance, aucune approche ne fournit de stratégie pour classer les clefs générées en fonction de leur efficacité à découvrir les bons liens. Afin d'assurer des alignements de qualité, nous avons proposé dans ce travail une nouvelle approche de liage de données visant à relever les défis décrits ci-dessus.

Nous avons développé *Legato* — un outil générique de liage automatique de données hétérogènes qui répond aux challenges évoqués précédemment. Notre outil est basé sur la notion de profile d'instance permettant de représenter chaque ressource comme un document textuel de littéraux gérant une variété d'hétérogénéités de données sans l'intervention de l'utilisateur. *Legato* implémente également une étape de filtrage de propriétés dites problématiques permettant de nettoyer les données du bruit susceptible de rendre la tâche de comparaison difficile. Pour pallier au problème de distinction entre les ressources similaires dans leur description, *Legato* implémente un algorithme basé sur la sélection et le ranking des clefs. Une clef représente un ensemble de propriétés qui identifie les ressources de manière unique. Pour cela, la stratégie que nous avons adoptée est de regrouper les ressources similaires dans chaque jeu de données. Puis, pour chaque paire de clusters similaires de deux jeux de données, les ressources sont comparées en utilisant la meilleure clef identifiée par notre algorithme de sélection et de ranking des clefs ce qui devrait améliorer considérablement la précision au niveau des liens générés.

Les différentes expérimentations menées ont clairement montré l'apport de certaines étapes tout au long du processus de notre approche de liage. A l'étape de nettoyage des données, il a été démontré que la suppression des propriétés problématiques améliorait l'exactitude des liens générés. En effet, nous avons pu constater que ce type de propriétés rendait difficile la tâche de comparaison. Les

expérimentations effectuées à l'étape de profiling d'instances ont mis en avant la nécessité de représenter les ressources par leur CBD (Concise Bounded Description) étendu. Autrement dit, chaque ressource est représentée par un sous-graphe contenant tous les triplets de son CBD, du CBD de ses prédécesseurs directs et du CBD de ses successeurs direct. L'idée derrière cette stratégie de représentation est qu'il peut y avoir des informations pertinentes lors de la comparaison dans les prédécesseurs et/ou les successeurs de chaque ressource. Nous pouvons prendre l'exemple des données DOREMUS, où l'information sur le compositeur de chaque œuvre se trouve dans son prédécesseur. Au meilleur de nos connaissances, aucune approche dans la littérature n'exploite cette orientation dans le graphe. De plus, les expérimentations soulignent l'intérêt de l'étape de post-traitement que nous avons proposée en termes de réparation des liens erronés. En effet, cette étape fut particulièrement intéressante en présence d'instances très similaires dans un seul jeu de données. Toutefois, nous avons constaté que certains nouveaux liens peuvent être générés à l'étape de post-traitement mais qui n'ont pas pu être identifié lors de l'étape de pré-matching.

L'efficacité de l'approche de liage implémentée dans *Legato* a été validée sur différents jeux de données synthétiques et aussi réelles soient-elles. En effet, *Legato* a participé à la compagne d'évaluation OAEI[9] (Ontology Alignment Evaluation Initiative) 2017 où il s'est montré très performant sur le benchmark des données. Même en présence d'instances très similaires, *Legato* peut générer des liens de qualité. Il s'est d'ailleurs montré le meilleur outil sur un jeu de données de la compagne contenant uniquement ce type d'instances avec une exactitude de liens élevée à 100%. De plus, les différentes expérimentations menées ont clairement montré que notre système produit de très bons résultats en termes de F-Mesure et les meilleurs alignements en termes de précision grâce à sa stratégie de réparation de liens.

[9] `http://oaei.ontologymatching.org/2017/`

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Contents

An increasing amount of data, publishers, and users has been noticed over the last years on the worldwide web. Data providers can easily share their contents in the form of documents called web pages. The simplicity of their structure makes it easy to read new content through web browsers. However, access to relevant information becomes particularly complicated considering the explosion in the amount of stored data. Seeking the desired information could be a difficult or even impossible task in all this mass of data. In fact, the World Wide Web (WWW) is estimated currently at 4.58 billion pages[1]. The problem addressed above is also due to the heterogeneity of data: the same information can be encoded in different formats and in different languages. Note that the information content is expressed in natural language and consequently it is only in a human-readable form.

To face this problem, many efforts have been spent during the past years to make information automatically processed by the machines. In 2000, Tim Berners-Lee, the WWW inventor, proposed a new vision of the web of documents to the Semantic Web of annotated resources. The vision of Semantic Web, the principle of which is to semantically annotate the documents, appeared very promising. More particularly, thanks to this new architecture of the web, the software agents are able to understand the meaning of the published data.

The principles of the Web in general and the Semantic Web in particular can be summarized as follows [2]:

- Anyone can say Anything about Anything (AAA).
- Anything can be linked to anything.

Considering the AAA slogan, any user can describe any entity with any piece of data. Assume that no one knows everything about anything. Hereby, there always can be something else that somebody can say. As for the web of documents, the information content is provided under the Open World Assumption (OWA) in Semantic Web. Therefore, the first principle aims at enriching the published data. On the other hand, distributed information about the same real-world entity is produced. As an example, people can describe an artist with any piece of information they have. Given the large amount of published data, access to relevant information becomes difficult, hence the need to connect these data. This need has led to the emergence of the web of data.

In the following, we will motivate the problem of data linking on structured knowledge across heterogeneous data sources. Particularly, we will describe in Section 2.1 and 1.2 the popular standards, allowing to structure the information content on the web, and data linking problem, respectively. In Section 1.3, we start by

[1] `http://www.worldwidewebsize.com/`

motivating the general context of this thesis. Next in Section 1.4, we shift to a narrower context where we discuss a variety of challenges including data heterogeneities, similarities and the configuration cost. Section 1.5 presents the research questions alongside to the main contributions of this thesis on this regard. Finally, we conclude the chapter by highlighting the structure of the thesis.

1.1 Open Data

Data that are accessible and publicly available for reuse, redistribution, reproduction and without any restriction, are considered as *open*. The concept of *open data* appeared in the 2000s and has spread in scientific, governmental and civil societies fields. The idea aims at providing to the citizens a greater transparency for data processing allowing them to discover, to explore and to exploit the data across different areas. Indeed, it allows to develop applications or to conduct research work using this data. Wikipedia[2] is the best-known example of a data source whose information is made available under free license. Also, the United States offered a prominent example by providing citizens with a large amount of open data. This initiative allowed the launch of the portal *data.gov*[3] making available data generated by the federal government. Currently, this platform has almost $198k$ datasets about different areas (agriculture, education, finance, health, etc.).

To be considered as open, Tim Berners-Lee[4] proposed five criteria for evaluating the data:

- The data are available on the web and accessible via an URL.
- The data are provided in any non-proprietary format (e.g. `.csv` instead of `.xlsx`).
- The data are available in a structured and machine-readable form (e.g. CSV, XML and RDF).
- The data are provided with an open license so that they are freely reusable.
- The data are linked to other data from other sources.

However, due to the growing number of open data published on the web, they cannot be processed manually. From there appears the *open web of data* allowing access to the data through unique identifiers to be automatically queried regardless

[2] `https://fr.wikipedia.org/`
[3] `https://www.data.gov/`
[4] `https://www.w3.org/DesignIssues/LinkedData.html`

of where they are stored. In the open web of data, the information is made available in standard machine-readable formats. It denotes not only the supply of open data but also their linking to constitute a network of open linked data so that from a given information we can more easily and quickly access the other information. The process of linking entities to one another is called *data linking*.

1.2 Data Linking

Data are being published continuously on the web in a decentralized manner leading to a web of heterogeneous data, containing *duplicates*, *incomplete information* and often even *errors*. Data linking promises to address this issue and thus considerably improve the quality of the open web of data, facilitating the access to distributed and decentralized information.

In the Semantic Web field, the data linking task is defined as the process of establishing a relation between two resources coming from two distinct datasets, or in other words, declaring a triple that has its subject in one dataset and its object in another. We are interested in a specific kind of data linking that aims to connect *identical* resources through an equivalence relation called *identity link*. We will refer to data linking as the process of comparing two resources of two corresponding classes, across two datasets. The outcome of this process is the establishment of an identity link between the resources together with a degree of confidence of this assertion. Two resources that are found to refer to the same real world object are declared as being equivalent by linking their URIs r_1 and r_2 in an `owl:sameAs` statement of the kind $<r_1$,`owl:sameAs`,$r_2>$. The use of `owl:sameAs` further enriches the Linked Data space by declaratively supporting distributed semantic data integration at the instance level [3]. In fact, it allows to state individuals equality indicating that they have the same identity.

The term Linked Open Data (LOD) appears as of 2006 following the four principles recommended by Tim Berners-Lee[5]:

1. Use URIs to identify the resources;
2. Use the HTTP protocol to dereference URIs;
3. Provide useful information for URIs using Semantic Web technologies;
4. Include links to other URIs to discover more information.

[5] `https://www.w3.org/DesignIssues/LinkedData.html`

A URI (Uniform Resource Identifier) [4] is a protocol allowing to identify, in a unique and uniform way, any resource used in a web application. We mean by resource any concrete or abstract entity, such as a file, image, video, concept or a person.

The second principle indicates that it is possible to perform an HTTP request to a URI in order to receive the information about the resource identified by this unique identifier.

After referencing the resources, the next step is to annotate them, in order to give them meaning that can be interpreted by the machines. This is precisely the role of the RDF (Resource Description Framework) protocol. The RDF has been standardized and defined by the W3C as "a standard model for web-based data exchange" allowing to describe the resources. Please notice that a more detailed definition of this language is provided in Sub-section 2.1.1 of Chapter 2.

According to the fourth principle, it is necessary to provide references (links) to other URIs allowing the users to retrieve effectively and efficiently the relevant information.

To better illustrate the benefit of data linking consider the following example, where a journalist would like to have all possible information about a particular event. Anyone who was present at this event can describe what happened with their own information and their own attributes in different web sites. The main problem is that the journalist would have to inspect whether the different testimonies concern the same event. Given the size of the WWW and the number of descriptions to compare, this task cannot be handled manually. What we need is to explicitly show that all these facts describe the same event by establishing an identity link between them. This explicit correspondence helps to complete and to make more precise the available information about this event. In fact, if the equivalent descriptions are linked together, the journalist can obtain them only from one query on the web.

1.3 Context and Motivation

This thesis is part of the DOREMUS –DOing REusable MUSical data– project[6] which is conducted in partnership with: BNF[7] (Bibliothèque Nationale de France),

[6] http://www.doremus.org/
[7] http://www.bnf.fr

la Philharmonie de Paris[8] (PP), Radio France[9], Meaning Engines[10], EURECOM[11], LIRMM[12], GERiiCO[13] and OUROUK[14].

Thousands of musical works are described in the catalogs of cultural institutions whose role is: (i) Storing - store all the musical creation through the cataloging as for BNF for example; (ii) Broadcasting - stream music to the general public as it is the case of Radio France; (iii) Storing and dissemination - this role consists in cataloging, disseminating and popularizing (Philharmonie de Paris). The three cultural institutions mentioned above do not all manipulate musical data in the same manner or pursue the same objective. The role of the DOREMUS project is to explore their data in such a way that they can communicate and that they can be more valued for the users. The idea is to store, order, classify and, finally, to make the data about music accessible, as much as possible, to different audiences. More particularly, DOREMUS aims to provide common knowledge models and shared multilingual vocabularies to cultural institutions, publishers, distributors and users in the musical domain. The project pays particular attention to the musical works which constitute the main entity held by the three partners. Indeed, one of DOREMUS objectives is to express, through a model, all the physical manifestations (recordings, partitions) of a musical work, as well as all the events that define it (creation, publication, interpretations).

Considering the large amount of data, the project focuses only on classical and traditional musical works as well as their interpretations (events). DOREMUS also relies on the DISCOTHEKA project carried out by Meaning Engines, in particular, for the ability to reuse open data on musical works. Among the applications of DOREMUS, the recommendation is a typical example. Its richness is its peculiarity, particularly in comparison with other similar applications such as Deezer[15], in terms of fine description of music. In contrast to Deezer, DOREMUS allows describing, for example, that a musical work is derived from another one. Indeed, this will be achieved by means of a process of aggregation of musical content, languages, tools and practices of Semantic Web. The thesis goes towards this direction. More particularly, we consider the whole workflow in Semantic Web dealing with the semantics, the publication and the linking of these works on the web of data.

[8] https://philharmoniedeparis.fr
[9] http://www.radiofrance.fr/
[10] http://www.meaningengines.com/
[11] http://www.eurecom.fr
[12] https://www.lirmm.fr/
[13] http://geriico.recherche.univ-lille3.fr/
[14] http://www.ourouk.fr/
[15] https://www.deezer.com/fr/

In this context, we ought to deal with the complexity associated with the size of the databases and the data formats used by each institution. The main sources of information included in this project are MARC -Machine Readable Cataloging- (BNF and Philharmonie de Paris) documents as well as documents in XML format (Radio France). In the literature, we identified a number of methods for automatically linking identical resources to each other. Despite the good performances described by the authors, we analyzed the task of linking in itself. Indeed, few of them were interested in the MARC format and even less those who manage multilingualism. This aspect seems relevant given the existence of multilingual labels in the data of our partners.

Figure 1.1 illustrates the data workflow in DOREMUS, ranging from raw musical data (from the 3 partners) to the RDF data. Once the content is published on the web of data, it can be exploited by the users (from music lovers to professionals) in many ways. In this thesis, we focus on the following three steps: conversion, publishing and linking.

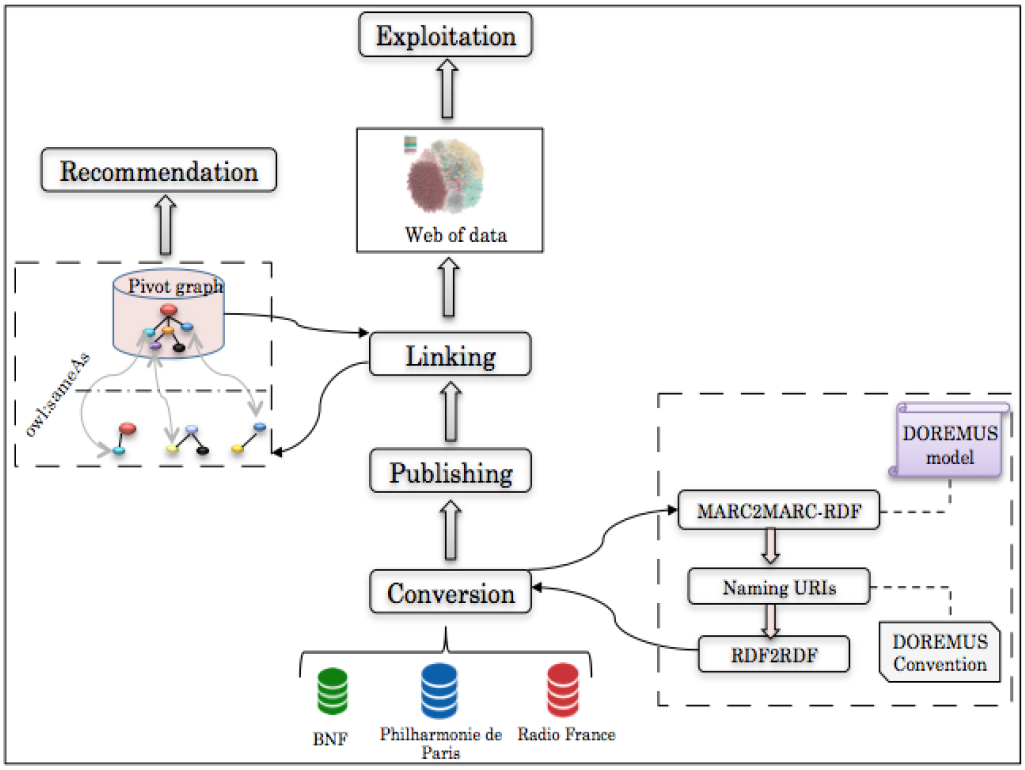


Figure 1.1: The DOREMUS workflow: from raw musical data to RDF on the web of data.

For the data transformation step where data is converted from its original format (MARC, XML) into RDF, an ontology was developed and provided by the modeling working group within the project. The modeling process is based on a narrower collaboration with the three cultural institutions. The DOREMUS project pays a particular attention to the musical works which constitute the main entity held by the three partners. Several existing models can be used to represent a musical work: MusicOntology[16], FRBR[17], Event[18], etc. However, a specificity of musical works is their complexity. In the context of DOREMUS project, the chosen model should be able to describe both the physical manifestations (recordings and partitions), the events that define them (creation, publication and interpretation) and possibly all the relationships between theses works. This is not the case for existing models. Therefore, the modeling group worked on a new model, based on FRBRoo[19] which itself is an extension of the CIDOC-CRM model[20], constitutes a new musical ontology allowing to take into account the description of the works and the events that concern them.

1.4 Challenges

One of the promising visions of the Semantic Web is the ability to share a very large amount of semantic data in an open context. As a result, the web of data, and particularly the Linked Open Data (LOD) project[21], has been receiving growing popularity over the past years, with hundreds of datasets published on the web. Given that data are published in a decentralized manner, this sets new challenges for discovering identity links in the open context of the Web.

Considering the first principle of Semantic Web mentioned above, any data provider can describe any real-world entity by assigning it a URI. Note that there is no URI naming convention, i.e., every user has the freedom to name the entity to be described in any manner. Without imposing constraints on the URI naming forces the discovering of resources referring to the same real-world entity and connecting them with identity links. Given the large amount of semantically structured information, it seems difficult if not impossible to manually proceed for aligning different datasets especially if there is no semantic and structural homogeneity between them. In fact, the lack of guidelines on the use of schema for each in-

[16] musicontology.com
[17] https://www.ifla.org/publications/functional-requirements-for-bibliographic-records
[18] http://motools.sourceforge.net/event/event.html
[19] http://www.cidoc-crm.org/frbroo/
[20] http://www.cidoc-crm.org/
[21] http://linkeddata.org

formation type may have a negative impact on the matching decision. A linking tool has to be able to deal with a large variety of data heterogeneities, by taking into account differences in descriptions on value, ontological or logical level. While heterogeneities on literals are rather well-handled by similarity measures and data unification techniques, ontological discrepancies (regarding structure and properties) appear to be way more challenging.

Another challenge for data linking problem in the open context of the web is to handle correctly datasets containing blocks of highly similar in their descriptions, but yet distinct resources (for instance, two datasets composed by piano sonatas by Beethoven, Brahms and Schubert), likely to generate false positives. We assume that descriptions about similar instances usually have very small distance, and that no tool is able to efficiently separate highly similar though not identical ones. In this particular case, no similarity threshold will be sufficient to catch all the equivalences and distinguish between these highly similar entities. In fact, even if the threshold is set at a lower value, a data linking tool may produce a higher number of false positive match.

Example. Three high similar musical works of Pierre Boulez retrieved from PP and BNF are shown in Figure 1.2 and 1.3, respectively. The identity mappings between them are declared by the experts of DOREMUS as follows:

<mw1, `owl:sameAs`, mw1'>
<mw2, `owl:sameAs`, mw2'>
<mw3, `owl:sameAs`, mw3'>

In order to make more explicit the complexity of the problem, we consider in this example only a sample of their descriptions.

Looking at the descriptions in Figure 1.2 and 1.3 separately, it is possible for a human person to conclude that they are different even if they share most of their attributes. In fact, only the underlined and in bold words differ from one work to another. However, it is not obvious for a tool to automatically distinguish between them. Consider the case where a tool has to compare the works coming from BNF with those coming from PP, the task of finding the right correspondences seems to be particularly complicated. Therefore, if we compare *mw1'* with *mw1*, *mw2* and *mw3*, the similarity value will be affected by a high weight for each of them which increases the problem of false positive matching.

We argue that setting the linking tool parameters is another challenging problem that often requires in-depth knowledge of the data. State-of-the-art tools like LIMES [5] or SILK [6] require link specification files where one has to indicate

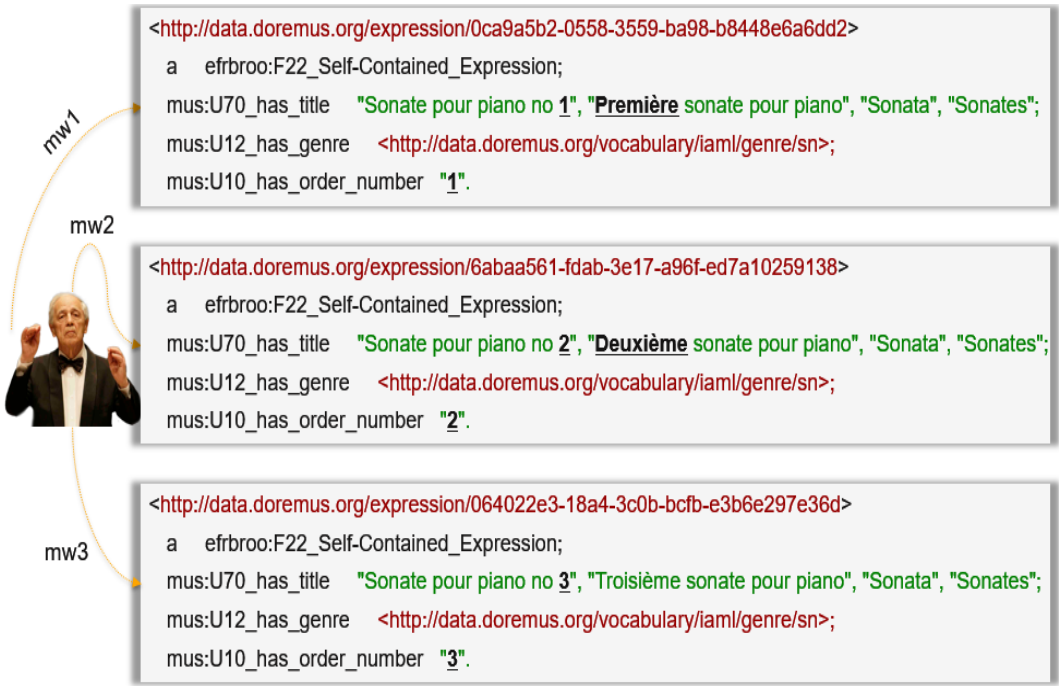


Figure 1.2: An excerpt of the musical descriptions of Pierre Boulez retrieved from PP.

property names, select and tune similarity measures. This process is handled either manually, or by specialized tools. Making a linking tool self dependent in that respect is among the challenges that we set in this work. Where the user is expected to provide the names of the classes, in which to look for instances to match and the properties whose values to compare. While the choice of pairs of corresponding classes is rather well-assisted by ontology matching techniques [7], property selection appears to be a more challenging issue. Often equivalent properties do not help in the matching process, or even lead to errors, as we show in the sequel. Automatic key discovery techniques [8, 9] attempt to alleviate that problem by providing discriminative properties for each dataset, but their direct application to data linking remains limited by the lack of measure of usefulness for these keys on both datasets. Providing a better user assistance in the linking configuration process appears as an important challenge. Since this process remains in its essence dissociated from the matching task and, therefore, of what the two datasets actually have in common, the usefulness of the automatically generated keys remains variable. This is particularly so because these algorithms generate keys in large numbers, lacking a link-likelihood score. To close the gap between automatic key discovery and data linking approaches, we propose an approach (as explained in Chapter 4) which allows reducing the user effort significantly by

Figure 1.3: An excerpt of the musical descriptions of Pierre Boulez retrieved from BNF.

selecting automatically the best keys relevant to the datasets to be linked.

1.5 Research Problem and Contributions

The Semantic Web community has addressed the problem of matching identical instances over RDF graphs, an area that keeps evolving. The configuration involving human user is not the only problem that faces a data linking system. As briefly explained in Section 1.4, another important issue is to discover the identity links under the heterogeneities that characterize the data and in the presence of highly similar resources. Our work is motivated by the fact that identifying the correspondences considering these issues is a non-trivial task which should clearly require filtering and repair processings. Indeed, the specific research problem that we address in this thesis can be stated as follows:

Research problem: *what requirements must a data linking system fulfill to reduce as much as possible the user's intervention and how can it overcome the differences of identical resources and discover the similarity of distinct ones?*

Of this problem arise four main research questions:

Research Question 1. *What are the different heterogeneity types and how are they organized?*

As we have stated in Section 1.4, data linking can be particularly complicated considering the heterogeneous descriptions of the resources. To gain an overview of different heterogeneities is itself a challenging task; that it is partially (not exhaustive) addressed in Chapter 2.

Research Question 2. *How to deal with the different heterogeneity aspects between the resources?*

There are several challenges to overcome varying from the schema to the instance layers. It is important to note that this work does not address the heterogeneities at the ontological level, as many state-of-the-art ontology matching approaches already exist [10–14]. Thus, we assume in our proposed data linking approach the presence of the ontological mappings.

Research Question 3. *How to make the linking process as automatic as possible?*

Most of the data linking approaches require user intervention throughout the process, especially in the preprocessing and post-processing steps. Usually, the preprocessing allows to train a classifier with links manually annotated by the user [15, 16] or to tune parameters such as thresholds [5, 6, 17–20], similarity measures or selecting in an appropriate manner the attributes of interest to compare. This requires a high degree of knowledge from the user about the data as well as about the functioning of the linking tool he uses. Such knowledge is usually time expensive for users to accomplish the task the first time he encounters the data. In fact, it is not possible to manually quantify the heterogeneity of data in order to fix a threshold about their similarities.

To address the question, we choose pragmatically to set a unique threshold regardless of the data (whether heterogeneous or not). In chapter 5, we describe how can identity links be effectively generated without setting the threshold according to the data. Furthermore, selecting the appropriate properties to compare is a non-trivial task. The majority of the linking approaches adopt a property-based philosophy. The precision in selecting the right properties to compare is of high importance for the quality of the produced linkset. Consider for example the property that provides information about the identifier of a music work belonging to

some institution. In the context of the DOREMUS project, for the same musical work, the identifier is undoubtedly different across the three institutions since they independently assign different identifiers for all the works. Thus, we argue that this type of property could have a significant impact on the matching decision. To overcome this issue, we propose a filtering strategy of so-called *problematic* properties (see Sub-section 5.1.1).

Research Question 4. *How to distinguish between highly similar resources?*

The existing data linking approaches are designed on the assumption that resources from the same data source usually have a high distance. Only a small number of approaches fits within the opposite hypothesis, i.e., those based on the *clustering* [17, 21] or *blocking* [22] techniques. When each of the two datasets to be matched contains resources with similar descriptions, these approaches regroup such resources together according to one of these three cases: (i) **Case 1**. Similar resources share a similar label (based on the `rdfs:label` property) according to a given threshold [17]; (ii) **Case 2**. Similar resources share similar values of the key predicates [22]; or (iii) **Case 3**. Similar resources share the same terms in their documents where a document contains the literal information retrieved at a given distance from a resource [21].

However, these solutions do not perform well since sharing similar labels is not necessarily a crucial matching criterion. If we consider the example given in Table 1.1, the musical works $mw1$, $mw2$, $mw3$ and $mw4$ retrieved from PP share the same value '*sonata*' of the `rdfs:label` property whereas they are completely different in their description.

Musical Work	Composer	Title	Opus	Key
$mw1$[22]	Ludwig van Beethoven	Sonate pour piano no 3	Op. 2 no 3	C Major
$mw2$[23]	Scriabine Alexandre	Sonate no 2 en sol dièse mineur	Op. 19	G sharp Minor
$mw3$[24]	Corelli Arcangelo	Sonate pour violon et basse continue no 4	Op. 5	F Major
$mw4$[25]	Grieg Edvard	Sonate pour violon et piano no 1	Op. 8	F Major

Table 1.1: Musical works from PP sharing the same value 'sonata' of the property `rdfs:label`.

As stated in case 2, some approaches gather the resources sharing similar values of the properties identified as keys. In practical terms, this means that given two

[22] `http://data.doremus.org/expression/1a62abda-26a1-36d1-b4aa-c930f68b31ed`
[23] `http://data.doremus.org/expression/08b19fb9-4ee1-3169-a04e-2d0d626d6c28`
[24] `http://data.doremus.org/expression/822e5a32-e3c4-3aac-9dca-f6da9948e616`
[25] `http://data.doremus.org/expression/4fd3dff5-1af3-3862-bb04-41f91b6e091e`

datasets A and B, and the sets of discovered keys K_A and K_B respectively, it consists at grouping the resources sharing the same values of $K_A(K_B)$ in A (in B). However, discovering keys does not seem relevant when it is performed in an upstream of the process of identifying similar resources in the same dataset. Indeed, we are quite confident that the reverse direction of this process allows us to reach our objective more efficiently and produces better results. In fact, once similar resources are clustered, the strategy consists in identifying the key predicates that distinguish them. Note that several keys can be generated for the same dataset. The final aim being to compare the most similar clusters, a tool can also identify different keys between them. Therefore, this entails the following fifth research question:

Research Question 5. *Are all the keys generated for one dataset equally important?*

In other words, we need to answer the question that whether some of the generated keys produce more accurate links than others. In Chapter 4, we formulate this problem as a ranking problem, where the goal is to rank the keys that are valid for *both* datasets and to select the best one. We also discuss the problems that can occur during the ranking.

Other approaches represent each resource using its literals retrieved at a given distance ($d >= 1$) and cluster those having similar representations (case 3). We argue that there is no better than comparing resources by their most representative description. In fact, here arises the problem of which information should be compared, i.e., set a high distance may introduce noise and set a low distance may not be sufficient to retrieve the important information. This issue entails the following sixth research question:

Research Question 6. *How can the resources be effectively represented?*

Considering the second research question, one of the main objectives of our work is to make our data linking tool as automatic as possible. Setting the distance parameter represents a real challenge. Therefore, it is important that the distance is not considered in the representation of resources. In this thesis, we refer to this problem as *instance profiling*, which is designed to represent the resources only by their most representative literal information. The resources that share similar profiles will be in the same cluster. Only the most similar clusters will be further compared in detail. In Chapter 5, we investigate how to profile the resources by their relevant description for effective data linking. We also show

how to distinguish between resources in the same cluster using the key ranking algorithm.

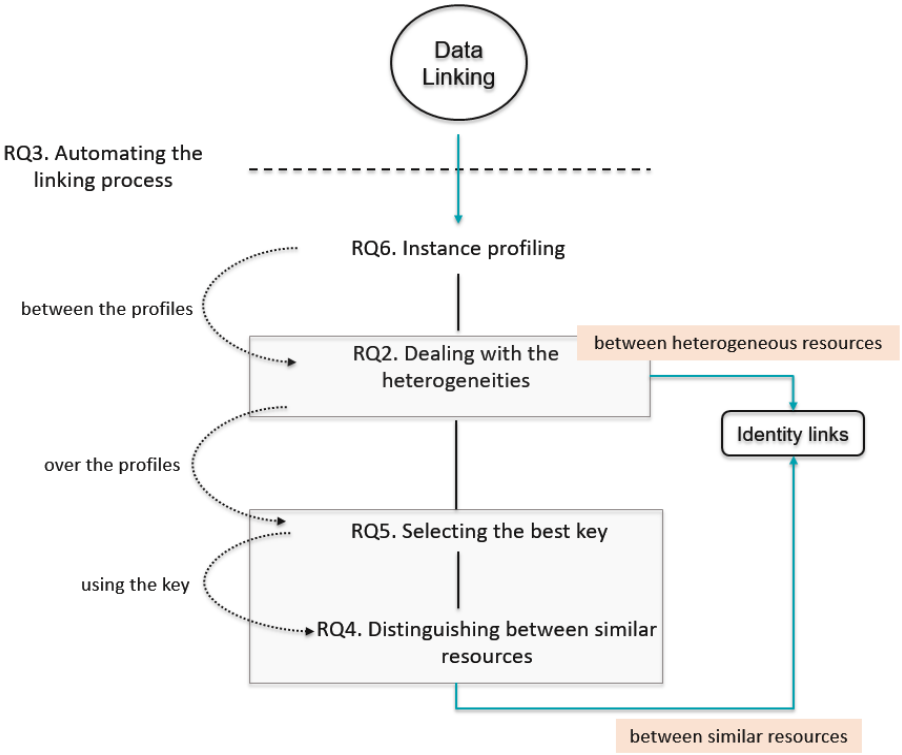


Figure 1.4: Positioning of the research issues in the overall linking process.

The relation between the proposed solutions, regarding the steps of our data linking approach (Chapter 5), of the questions raised above is depicted in Figure 1.4.

1.6 Thesis Outline

The remaining chapters discuss one of the main contributions cited before. This dissertation is organized as follows:

- *Chapter 2* provides the definitions of the main concepts used in this thesis. Particularly, it presents the necessary preliminaries on RDF data model, instance profiling and the notion of keys in a Semantic Web context. The second part of this chapter outlines in detail the data heterogeneity types that a linking tool might have to confront. We show that some of these issues are not tackled by most of the state-of-the-art alignment tools. Then,

we analyze these issues and identify a set of specific techniques that can be applied.

- *Chapter 3* provides an overview of the different techniques applied on each step in service of the global linking task. We consider the linking process as a pipeline composed of preprocessing, data linking and post-processing steps. Finally, we described and compared different state-of-the-art approaches and tools according to these steps and to the surveyed techniques. Exploring such techniques is important to discern which of them can be applied for which task and in what stage of the overall process of linking.

- *Chapter 4* describes our solution regarding the fifth research question where we propose an RDF key ranking approach that attempts to close the gap between automatic key discovery and data linking approaches and thus reduce the user effort in linking configuration. It presents the implementation of a prototype called RANKey that implements the approach proposed in this chapter. We prove the completeness of the top-ranked keys showing through the experiments that the combined use improves significantly the recall. We empirically evaluate the discriminability impact of a set of properties on ranking with respect to their coverage. Moreover, we investigate how some properties identified as problematic can have a negative effect on the keys ranking. This chapter builds upon our own work, published in [23].
- *Chapter 5* describes *Legato*, the approach and tool that automatically discovers identity links across RDF datasets. It is a novel way of thinking the instance matching task, attempting to reduce the difficulty of manual configuration, especially when it comes to data-related parameters, such as properties to compare, similarity measures to use or threshold setting. In this chapter, we describe how Legato addresses efficiently many of the data heterogeneities presented in sub-section 2.2. Also, our approach is able to discriminate between highly similar, but different resources, thanks to an efficient post-processing strategy. Using benchmark matching tasks, we show that *Legato* is in competition with state-of-the-art tools, outperforming them on datasets containing highly heterogeneous or difficult to disambiguate instances.
- *Chapter 6* summarizes the thesis with a discussion about the different results regarding the research problem and concludes with some perspectives for future research directions.

Chapter 2

Foundations and Challenges

Contents

Introduction

In the first part of this chapter, we introduce some preliminaries required to follow the rest of this thesis. After defining the basic concepts in Section 2.1, an overview of the most relevant issues considering the heterogeneities when comparing two descriptions, is presented. An important point is that the way that resources are structured and valued can greatly impact the complexity of matching decision. We show how the comparison can be complicated due to the variations according to three dimensions: *value*-based, *ontological* and *logical*. Following the observations made in Section 1.4, the second part of this chapter aims to analyze the more relevant data heterogeneities and identify a set of specific techniques that can be applied. Finally, we discuss the challenges and the focus of this work.

2.1 Preliminaries

This section introduces a standard of the open web of data. Particularly, we will see a fundamental recommendation of the semantic web: RDF for structuring linked data. We will then describe two main concepts: instance profiling and keys. Theses notions are used by most RDF data linking approaches including our approach described in Chapter 5.

2.1.1 RDF Data Model

A *data model* aims at structuring in an abstract way how real-world entities are organized and at describing the relationships between them. The data can be modeled in different ways[1]: (i) They can be stored in *relational* tables; (ii) For their exchange with information systems, they can be structured in an XML *tree*; or (iii) To make them in a machine-readable form, the data can be described in a *graph* structure. In the Semantic Web context, several graph-based models have been proposed. RDF (Resource Description Framework) is the first basic brick of Semantic Web, and was defined by W3C[2] as "the standard model for data interchange on the web".

As its name implies, RDF is a standard data model allowing to describe resources on the web using a set of statements such as:

[1] `https://www.w3.org/1999/04/WebData$#$models`
[2] `https://www.w3.org/RDF/`

- Each RDF statement is a triple of <*subject*, *predicate*, *object*>. This means that an RDF triple expresses a relation "*predicate*" between two given resources "*subject*" and "*object*".
- A resource denotes any element that can be referred, whether abstract or concrete such as a book, a picture, a web page, an event or a person.
- A set of RDF triples containing as subject the same resource r are considered as the *description* of r.

An RDF graph is defined by a set of triples of the kind $\boldsymbol{t = <s, p, o>}$; where $\boldsymbol{s}$ is a subject (an URI or a blank node), $\boldsymbol{o}$ is an object (an URI or a literal) and $\boldsymbol{p}$ is the relation (an URI) between $\boldsymbol{s}$ and $\boldsymbol{o}$.

For example, the declaration "The moonlight sonata is a musical work by Ludwig van Beethoven whose web page is: `https://fr.wikipedia.org/wiki/Ludwig$_$van$_$Beethoven`" can be decomposed into four triples as follows.

1	Resource	The moonlight sonata
2	Resource's type	musical work
3	Resource's composer	Ludwig van Beethoven
4	Composer's URL	`https://fr.wikipedia.org/wiki/Ludwig_van_Beethoven`

Notice that the two elements "the moonlight sonata" and "Ludwig van Beethoven" have the property values (2,3) and (4) respectively. However in Semantic Web, only a resource referred by an URI or a blank node can be described by a set of properties. In our example, "the moonlight sonata" and "Ludwig van Beethoven" are two strings (literals). Thereby, the statements (1) and (3) must be decomposed in two other statements each as follows.

1	Resource	`pref:mw`
5	Resource's title	The moonlight sonata
3	Resource's composer	`pref:composer`
6	Composer's full name	Ludwig van Beethoven

This means that we created two URIs `pref:mw` and `pref:composer` referring the resources about the moonlight sonata and Ludwig van Beethoven, respectively. Notice that to reduce the size of the URIs, it is possible to use prefixes. In our example, we used `pref` which may refer the prefix of any URI that we must set before declaring the RDF triples.

The graph given in Figure 2.1 presents the description of the resource `pref:mw`. The nodes represented by an ellipse designate an URI while an empty ellipse

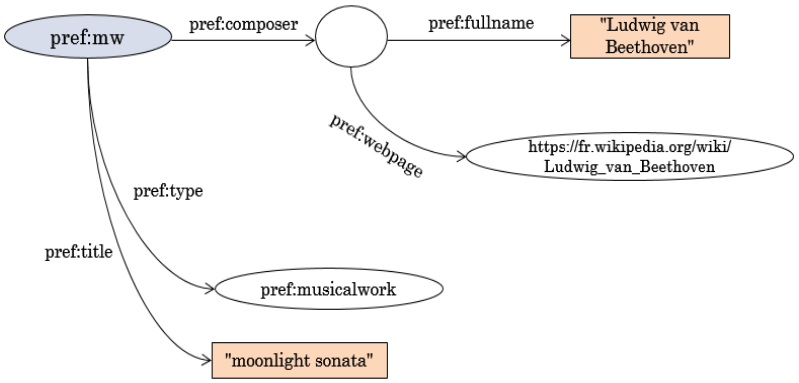


Figure 2.1: An RDF graph example.

denotes a blank node. Those represented by a rectangle designate a literal (string).

An RDF graph can be expressed with different syntaxes (serializations). The commonly used ones are: RDF/XML [24], turtle [25], n-triples [26] and N3 [27]. Figure 2.2 illustrates a serialization in RDF/XML of the graph in Figure 2.1. RDF/XML is the first syntax recommended by W3C for representing RDF graphs in XML structure. An RDF/XML document uses XML namespace notations and `.rdf` extension.

```
<?xml version="1.0"?>
<rdf:RDF
        xmlns:pref="https://bib-data.fr/" >

 <rdf:Description  rdf:about="https://bib-data.fr/musicalwork#mw">
 <pref:type  rdf:resource="https://bib-data.fr/musicalwork"/>
 <pref:title>  moonlight sonata  </pref:title>
 <pref:composer  rdf:nodeID = "idNode"/>
 </rdf:Description>

 <rdf:Description  rdf:nodeID = "idNode">
 <pref:fullname>  Ludwig van Beethoven  </pref:fullname>
 <pref:webpage  rdf:resource="https://fr.wikipedia.org/wiki/Ludwig_van_Beethoven"/>
 </rdf:Description>
```

Figure 2.2: Representation of the graph of Figure 2.1 in RDF/XML format.

2.1.2 Instance Profiling

An instance profile can be broadly defined as a set of features that define an instance. Typically, the profiles are the basis on which the instances are compared. In other words, we focus on the question of: for each source resource, what is the suitable information to use as a *profile* to be compared with the profile of a given

target resource? Following what is common practice in selecting the most relevant descriptions of resources in RDF data, an instance profile would be the set of triples where the subject is the described resource or any other resource with which it is connected. In this thesis, we adopted the notion of *instance profile* to represent a given resource with a set of features for which we intend to compare with other profiles.

The definition of profiling at instance level is analogous to the one defined at dataset level. A dataset profile as defined in [28] is *the formal representation of a set of features that describe a dataset and allow the comparison of different datasets with regard to their characteristics*. Remember that the last criterion of open RDF data is that they should be linked to other data from other sources (see Section 1.1). However, due to the huge amount of data on the web, manually finding potential datasets containing equivalent resources is a non-trivial task and may be expensive. Ideally, each dataset should provide information describing the resources it contains; so that we can quickly examine whether these resources can be linked or not. This allows a data provider to significantly reduce the search space and look directly at the candidate datasets for linking. Nevertheless, there were few researches that focused on this problem [29–32]. According to a more recent research [33], a dataset profile is formed from the information collected at the schema level; the process is referred as *intensional* profiling. In [33], two datasets are candidate for linking if they share at least one pair of semantically similar concepts. More particularly, the similarity between the concepts labels of two datasets is computed.

To remove any ambiguity over the interpretation of instance profiling considered in this dissertation, we are interested in profiling for linking and not to reduce the search space. At the step of matching of data linking process, the instance profiling operates at different levels:

Value-based profiling. The features at this level are identified according to some values only, i.e., the values of datatype properties as well as those of object properties. A typical example is profiles called *virtual documents* or *textual documents* containing only the literals based on the graph traversal according to a specified depth [34, 35].

<Predicate,Value>-based profiling. The features at this level are identified according to some valued properties. In other words, an instance profile is the set of <predicate,value> pairs such as two profiles are similar if their share the same values for most or all of their properties. A typical example is profiles based on

key dependencies such as for comparing two resources, the similarity between the values of a given set of discriminative properties is computed [31].

The main drawback of existing instance profiling techniques (except keys-based profiling) is that for each resource the information is retrieved at a given distance. Setting a small distance may not be enough to get relevant information. In contrast, setting a large distance may collect information with noise. It is not an easy task to decide the distance to which we want to represent a resource.

2.1.3 Keys for RDF Datasets

In the context of Linked Open Data (LOD) [36], the resources can be connected with different sorts of semantic links. As mentioned in Section 1.2, we focus in this dissertation on *identity* links, i.e., defining that two descriptions are about the same real-world entity. Many approaches were proposed to detect such links in RDF data (see Chapter 3 for a survey). Some of them rely on the use of *keys* to identify and establish these links between equivalent data items.

A set of properties are considered as a *key* for a given class if their combination uniquely identify each resource. This means that all the resources are distinct for this set of discriminative properties. Formally, a *key* in an RDF graph is defined as follows.

Definition 1 (Key) *Let G be an RDF graph, let $subj(G)$ be the set of resources in G and let $pred(G)$ be the set of properties in G. We define a key denoted by K as the set*

$K = \{P : P \subseteq pred(G)$ *and* $\nexists \mathtt{r}_1, \mathtt{r}_2 \in subj(G)$ *such as* $p(\mathtt{r}_1) = p(\mathtt{r}_2)$ *for all* $p \in P\}$,

where $p(\mathtt{r}_1)$ and $p(\mathtt{r}_2)$ are the values of the property p for the resources $\mathtt{r}_1$ and $\mathtt{r}_2$, respectively. A set of properties P is considered as a *minimal key* if it is a key and there is no subset of P which is also a key.

This definition of a key is similar to the one from the relational database field. However, the main difference is that in RDF data a property can be multivalued. Moreover, a new challenge emerging from Semantic Web is *incompleteness* of the data considering the OWA (Open World Assumption). In other words, another characteristic of RDF data is that a property does not necessarily cover all the data items. These features should be taken into account when identifying the keys in the RDF data.

Notice that for a given dataset several keys may be identified where several combinations of properties uniquely identify the resources. Let us consider the example

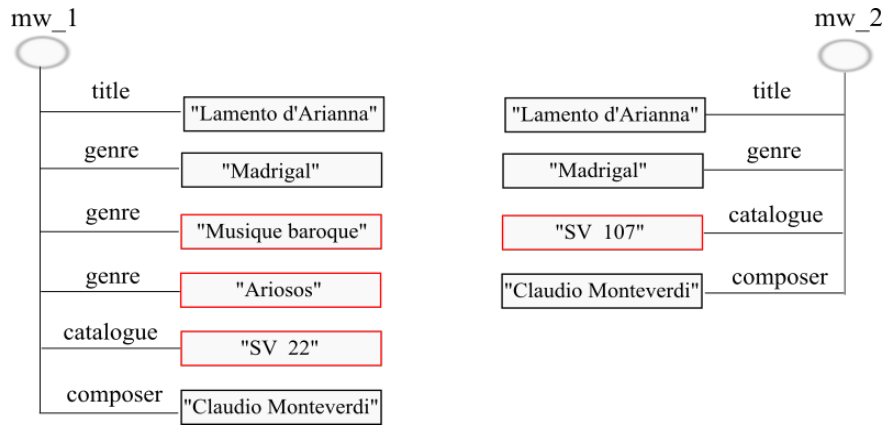


Figure 2.3: Property keys identification in an RDF dataset.

given in Figure 2.3 of two musical works *mw_1* and *mw_2* in an RDF dataset. Even if these works seem similar (the same title and the same composer), they refer to two different entities, hence the need to identify which properties, called keys, to compare in order to decide whether they are identical or not. Relying on key definition, we can deduce that the property *catalogue* is a key. However, a question that arises is: *How about the property* ***genre****?* Discovering keys under the *open world assumption* (OWA) considers that a property is a key if two resources share at least one common value for this property [8,37]. On the other hand, discovering keys under the *close world assumption* (CWA) considers that a property is a key if two resources share all the values for this property [9]. The former supposes that the description of a dataset is complete while the second does not. Thus, the property **genre** is considered as a key under the OWA but it is not the case under the CWA.

2.2 Data Heterogeneity Types

In this section, we focus on the most recurrent issues that may arise when comparing the resources illustrated by the means of a hypothetical example (see Figure 2.4).

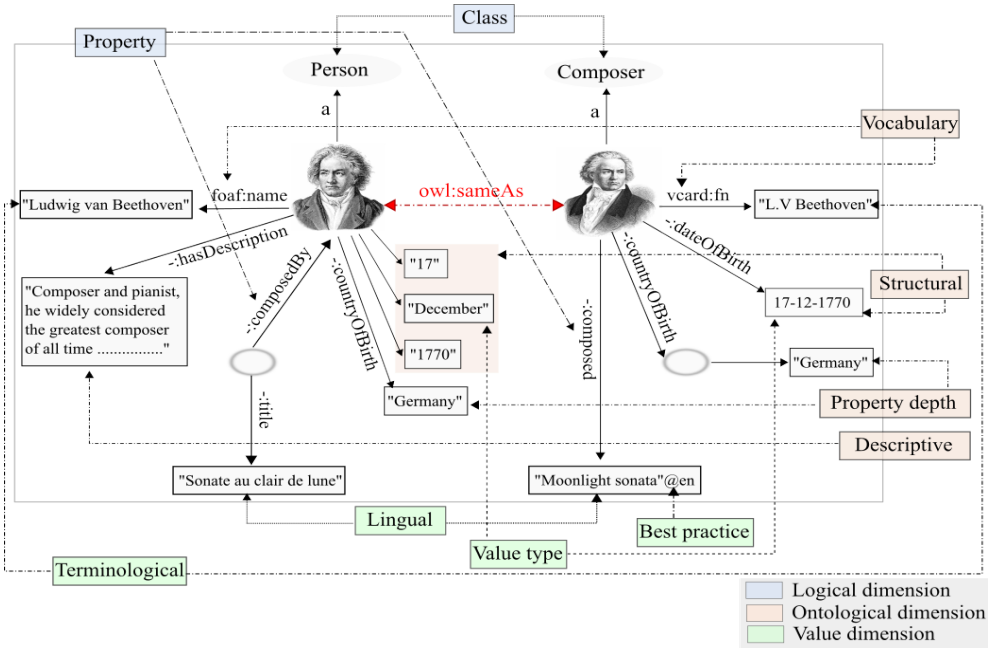


Figure 2.4: Issues occurring during the linking task.

2.2.1 Motivating Example

Let us imagine that the composer Ludwig van Beethoven is described by two different data sources without making any assumption about the way the data are structured in a general form nor about the used properties describing the composer. When the same entity is heterogeneously described (like the case of Ludwig van Beethoven in Figure 2.4), the comparison becomes much more complex. Furthermore, one of the main difficulties is the fact that data may be incomplete. This would raise the question about identity criteria for comparison – *between which attributes the comparison is done*? Variations in how such attributes are valued or structured may lead to missing true positive links or may produce false positive ones. Indeed, the matching quality improves when various aspects of heterogeneity between two resources are managed. In the following, we analyze the issues and identify a set of specific techniques that can be applied.

In the context of the instance matching tools benchmarking task, [38] introduce three major levels of data heterogeneity to be considered: *value*, *structural* and *logical* levels. We base ourselves on these levels and extend them in order to define a list of data heterogeneity problems relying on three core dimensions, namely: *value*, *ontological* and *logical* dimensions. The different heterogeneity types will be described in detail hereafter.

2.2.2 Value Dimension

In the open context of the web, the resources are described using natural languages in their string values. Any attribute valued with textual description may potentially raise matching issues. This characterizes the heterogeneities at data value level. In the following, we define a list of heterogeneity types of what we call *value dimension*.

1. *Terminological heterogeneity.* A description about an entity could present syntactical or semantic variations with respect to another description about the same entity. A term refers to a word or to a set of words. This heterogeneity can be expressed in: (i) Variation of terms to represent the same concept (synonymy); (ii) Variation of the meaning of a term representing different concepts (polysemy); or in (iii) Variation in spelling (acronymy or abbreviations). The problems of synonymy and polysemy can be solved using lexical databases such as WordNet [39] which is composed of sets of synonyms called *synsets* where each term belongs to one or more *synsets*. Each *synset* represents a particular concept and is accompanied by a description of the meaning it represents. Indeed, a word should be disambiguated to be compared. The disambiguation consists in assigning the most appropriate meaning for each word in a text according to the context in which it is found. Many works proposed solutions allowing to find the full form of an acronym or an abbreviation, among others, consider [40, 41]. This issue can be observed between "Ludwig van Beethoven" and "L.V. Beethoven" in our example (Figure 2.4). As we can see, the full name values of the two instances are distinct with the first source preferring to name a person by his/her first name and last name, while in the second source, a person is named using the initials of his/her first name and the full last name. Indeed, the comparison between such values becomes much more complex.

2. *Lingual heterogeneity.* Note that data providers often publish their data in their native language. To better clarify the impact of this fact on the matching decision, consider the example of BNF (French National Library) [42] and Freebase [43] that make their existing data available as RDF in their own language, i.e., French and English languages, respectively. Thereby, if we consider the same entity of Ludwig van Beethoven, we would end up not being able to compare correctly (if it is not impossible) its two representations particularly when applying string similarity measures. This problem is of crucial importance in the open web of data. Hence, it becomes necessary to discover links among RDF data with multilingual values. To overcome this problem, few studies proposed methods for automatic cross-lingual data

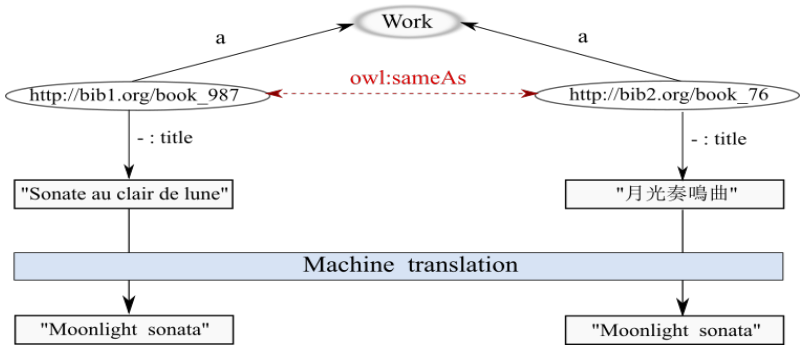


Figure 2.5: RDF resources described in different languages.

linking [20, 34, 35] reducing considerably the complexity of comparison task. Machine translation is performed by [20, 34] to make two descriptions in different languages comparable using Google Translator API[3] and Bing Translator API[4] respectively. A more recent study [35] proposed a data linking method based on the BabelNet multilingual lexicon to bridge the language gap. The authors define the resources as vectors of BabelNet identifiers where each of them represents a sense (concept identifier) of a term. The similarity between these vectors is then computed. TR-ESA [44] is a cross-lingual data linking tool which matches resources whose descriptions are written in different languages. Each resource description is translated into English using Bing Translator API[5]. Then, a wikipedia-based representation (a set of concepts) is generated for each resource. Then, these representations are indexed using Lucene. The main drawback of such approaches is that they do not deal with acronyms in the compared descriptions.

The difficulty of this issue is demonstrated in Figure 2.5 where two resources with multilingual object values are compared. In fact, the data property values in this example are expressed in English and in Chinese which makes it impossible to find a correspondence using string-based similarity metrics without using machine translation or external multilingual resources.

3. *Transgression to best practices.* Data representations on value level can differ depending on the degree to which the Semantic Web best practices are respected in the data publishing process. In Figure 2.4, this can be illustrated through the titles of Beethoven's work, where one of them ("*Moonlight sonata*") is annotated by the language in which it is given but not the other

[3] `http://code.google.com/apis/ajaxlanguage/`
[4] `http://datamarket.azure.com/dataset/bing/microsofttranslator`
[5] `http://datamarket.azure.com/dataset/bing/microsofttranslator`

(the presence/absence of a language tag). Another problem may occur on the values by introducing inappropriate symbols that convey no information. A good practice would be to ignore what we do not know due to the acceptance of the open world assumption (missing information is not necessarily wrong). This practice is not always respected, where symbols such as '♯' can be used to indicate an unknown value as for example when comparing the object values "29-07-1990" and "##-##-1990". Indeed, this may hinder the matching decision. Recently, best practices for the creation, linking, and use of multilingual linked data were elaborated by the multilingual linked open data community group.[6]

4. *Value type heterogeneity.* An efficient linking tool has to be able to deal with different value types expressing the same information. This heterogeneity type concerns differences in encoding of data. Often values are expressed in different formats or by using different value types, as for example, representing an '*age*'-value as a string (i.e., "*twenty six*") or as a number (26). This property describes the month of the year part of the birth date of a person (e.g. February). It is included mostly to ease problems related to structural heterogeneity in the representation of the birth date values. For example, some sources would not represent the date as a date format (consider the example of "17-12-1770"), but would rather represent the same information as a string ("*17 december 1770*"). The main challenge is to provide a means for semantic unification. Do not take into account the birth date would impact the quality of matching results. For instance, a possible solution consists in retrieving individual resource values in the RDF graph, transforming different object values in a given format, and standardizing the retrieved information to compare them in a uniform manner. Two data generators focused on this issue: the Semantic Publishing Instance Matching Benchmark (SPIMBENCH) [45] and LANCE [46] by transforming the source instances based on their values, structures and semantics. The aim of these generators is to produce benchmarks evaluating whether the data linking tools deal with certain problems. One of theses problems addressed by both SPIMBENCH and LANCE is the use of different date ("1948-12-21" vs "December 21, 1948"), gender ("Male" vs "M") and number formats ("1.3" vs "1.30").

2.2.3 Ontological Dimension

This dimension refers to variations in the properties or classes of instances to compare. We identify four main heterogeneity problems:

[6] `https://www.w3.org/community/bpmlod/`

1. *Vocabulary heterogeneity.* Classes and properties are often described by using different vocabularies by different data publishers because the semantics of a given class or a property can be interpreted differently according to the application and the intension. This problem is even more complicated in the context of open web of data where all resources are not necessarily described in the same manner. Indeed, different data publishers may interpret differently the semantics of attributes when they decide how to model their data. Indeed, it is not uncommon that the open nature of the web is one of the main causes of different uses of properties. In fact, everyone can define their own ontology and can describe their own classes. Let us consider the example of the property providing the information about the full name of Beethoven.

$$< i_1, \text{foaf:name}, \text{“}\mathbf{Ludwig\ van\ Beethoven}\text{”}>$$
$$< i_2, \text{vcard:name}, \text{“}\mathbf{Ludwig\ van\ Beethoven}\text{”}>$$
$$< i_3, \text{foaf:name}, \text{“}\mathbf{Beethoven}\text{”}>$$

For a given data source, such information could be described by the property *foaf:name* (i_1 and i_3), while for another data source it could be described by the property *vcard:name* (i_2).

Description	Vocabulary	Property
Full Name	contact[7]	`http://www.w3.org/2000/10/swap/pim/contact#fullName`
	FOAF[8]	`http://xmlns.com/foaf/0.1/name`
	DBpedia[9]	`http://dbpedia.org/ontology/name`
	VCard[10]	`http://www.w3.org/2006/vcard/ns#Name`

Table 2.1: Vocabularies describing the full name of a person.

Beyond the fact that the same information can be described by different properties (see Table 2.1), the same property can be used to describe different information. In fact, the instances have been described with the same property to describe the full name and the last name, respectively. Yet in both cases, the use of the property *foaf:name* is correct regarding the FOAF ontology[11]. Given the large amount of existing vocabularies, it becomes necessary to propose a solution aiming at finding correspondences between properties conveying the same information. Notice that the Linked Open Vocabularies[12](LOV) is a catalog containing more than 600 vocabularies in

[11] `http://xmlns.com/foaf/spec/#term_name`
[12] `http://lov.okfn.org/dataset/lov/`

the linked open data cloud. A vocabulary in LOV provides a set of terms (classes and properties) describing a given type of entities. LOV facilitates the reuse of vocabularies through a SPARQL endpoint[13] for retrieving types and their properties. LOV is a good initiative but notice that to date it is not exhaustive and has as a characteristic of being incomplete (e.g. Yago, Freebase or MusicBrainz are not included in LOV).

2. *Structural heterogeneity.* The description of an entity can be done at different levels of granularity. To take a brief example, consider the birth date of Ludwig van Beethoven. It can be described in a single information field (as a value of the property *vcard:bday*[14]) "17 December 1770" or distributed over several information fields (multiple properties) "17" (day), "december" (month) and "1770" (year). The three last values are parts of the property *vcard:bday*. Comparing two instances presenting this heterogeneity may impact the matching decision. A possible solution consists in identifying the sub-properties (day, month and year) as belonging to a given type (date) and aggregating them or in decomposing one value in several values to make the properties comparable at the same level of granularity. To the best of our knowledge, the problem of structural heterogeneity is not treated in the literature. However, it is partially resolved using inverted indexes and NLP (Natural Language Processing) techniques in some data linking approaches [6, 18, 21, 34, 35]. The method consists in indexing each resource by its literals collected at a distance[15] $n >= 1$ in the RDF graph. Then, a vector space model is used to represent each resource description as word feature vector. The resources sharing similar vectors (similar words) are considered to be equivalent or to be linking candidates. By doing so, the resources are compared with respect to their literals without taking into account the properties describing them. The main drawback of this process is the loss of precision when comparing resources represented as bags of words instead of comparing their literals in pairs (those defined with equivalent properties).

3. *Property depth heterogeneity.* This type of heterogeneity regards the difference in property modeling. In a given dataset, a property can be specified directly (at a distance $n = 1$) through a literal value while in another dataset the same information is given by a longer property chain including several triples. In our example (see Figure 2.4), we can observe this problem between the two resources, where the name of the country of birth is a literal that is directly assigned to the resource describing Ludwig van Beethoven, while

[13] `http://lov.okfn.org/dataset/lov/sparql`

[14] `https://www.w3.org/2006/vcard/ns#bday`

[15] A distance in an RDF graph is defined as the minimal number of edges (properties) connecting two resources or a resource and a literal.

for the other resource the same value is assigned through a literal to another resource, which is the URI of the country itself. In fact, we can observe that for the first resource the country of birth is defined through a datatype property while for the second one it is defined through an object property which is described by the same value of place of birth "*Germany*". This problem can also be solved indexing the scope of literals describing each resource. Namely, for each entity, the distance to which the literals are collected can be set. However, this is a limitation given the fact that in this context the further we get from the resource, the more likely it is that we collect noisy information.

4. *Descriptive heterogeneity.* A resource can have several types (concepts) or it can be described with more information (a larger set of properties) in one dataset than in another, as we can see in our example (see Figure 2.4). We can observe that a resource contains more information denoted by a descriptive text (biography) about Ludwig van Beethoven through the property *-:hasDescription*, while the other resource, referring to the same real-world person is described by a much more austere set of properties. It is obvious that comparing these two resources by this property will not identify any equivalence between them namely for approaches considering the whole description (all the literals) of a resource or for approaches relying on key properties to compare the resources. In fact, for the former approaches [34,35], the property *-:hasDescription* decreases the similarity value between the two resources while it is identified as a key property by the most of the second approaches [8, 9, 37].

2.2.4 Logical Dimension

This heterogeneity problem refers to the fact that the equivalence between two pieces of information across two datasets is implicit but can be inferred by the help of reasoning methods. We outline two main heterogeneity problems.

1. *Class heterogeneity.* This type of heterogeneity regards the class hierarchy level. This is typically the case of two resources belonging to different classes for which an explicit or an implicit hierarchical relationship is defined (the concepts "*Person*" and "*Composer*", in Figure 2.4, illustrate this issue). Moreover, two instances referring to the same object can belong to two different subclasses of the same class.

2. *Property heterogeneity.* At this level, the equivalence between two values is deduced after performing a reasoning task on properties. Two resources

can have two properties that are semantically reversed. In this case, these two properties convey the same information, as illustrated in the example in Figure 2.4:

$$<< i2 >, - : composed, \text{“}Moonlight\ sonata''@en >$$

$$<< i4 >, - : composedBy, < i1 >>$$

$$<< i4 >, - : title, \text{“}Sonate\ au\ clair\ de\ lune'' > .$$

Here the instances comparison process has to go beyond the value and property levels by comparing an explicitly specified value and an implicitly specified one between the two entities. Another example of this problem is given by two instances having "*26*" and "*29 july 1990*" as values of *age* and*birthdate* properties, respectively.

2.3 Conclusion

In the beginning of this chapter, we introduced the main notions that are used in this thesis. It is important to notice that given the large number of heterogeneities between the data sources, the instance profiling should be able to capture the most relevant information of each resource and thus to avoid noise.

In the second part of this chapter, we gave a broad overview of what may make the matching decision complicated. As a conclusion of our study, we note that, during the past years, significant progress has been made in the field with numerous off-the-shelf tools now available to the data community at large. However, we also outline that more effort is needed in order to allow for the matching tools to cope with certain more difficult and less studied heterogeneity types. Particularly, the value-based, ontological, and logical heterogeneity dimensions have to be paid more attention to. The challenge of linking multilingual data also remains largely unexplored.

In the remainder of this dissertation, we will show how our linking approach profiles the resources facing certain heterogeneity aspects and especially those of the *ontological* dimension. Before that, we will investigate more deeply the existing matching solutions in the next chapter.

Chapter 3

Data Linking: State of the Art

Contents

Introduction

In this chapter, we describe the main steps of data linking on the basis of our analysis. This aims at providing a bird's-eye view on the main research problem of this thesis. We consider the linking process as a pipeline composed of preprocessing, (pre-)matching and post-processing steps. Then, we provide an overview of the different techniques applied on each step in service of the global linking task. Finally, we describe and compare different state-of-the-art approaches and tools according to these steps and to the surveyed techniques.

3.1 Data Linking Steps

The linking process is composed of three main steps: preprocessing, (pre-)matching, and post-processing (see Figure 3.1).

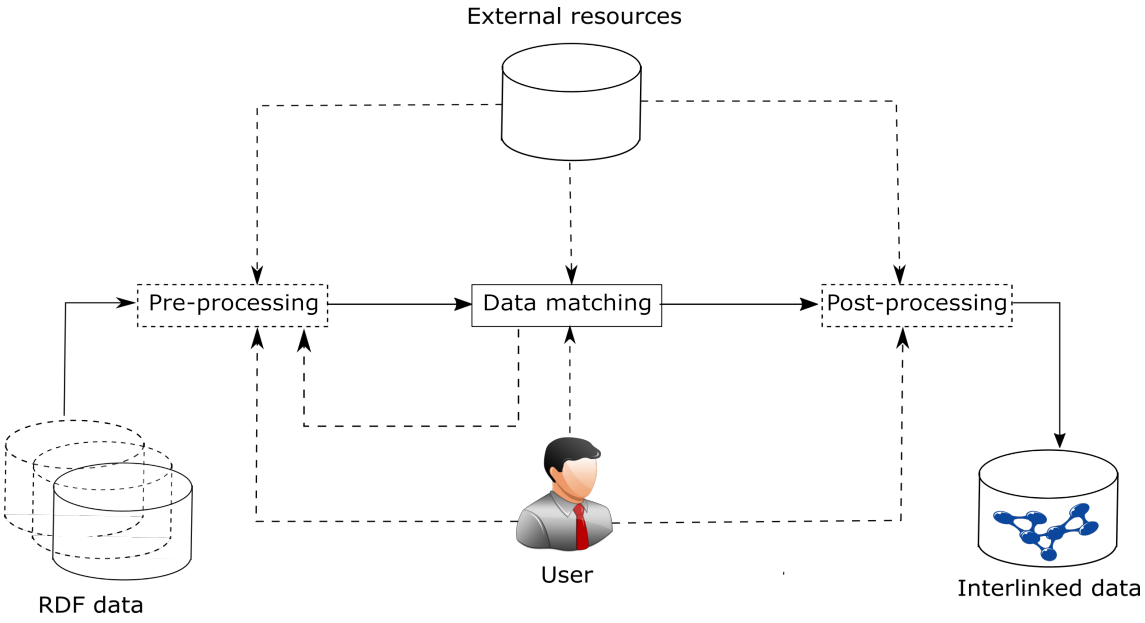


Figure 3.1: Data linking workflow, adopted from [1].

The preprocessing step aims at preparing the input RDF datasets for linking by representing them (see Sub-section 2.1.2) in a manner that allows for the comparison of data items. More than that, this step also aims at reducing the space where to look for potential linking candidates by identifying key properties and

equivalent classes of instances across datasets, assuring computational efficiency on large scale. The (pre-)matching task provides an assertion on the degree, to which data instances can be considered as referring to the same real-world object. Notice that in this thesis, we make a distinction between the notions of matching and pre-matching. The main difference is that for the former notion the generated links are final while for the second one the links are considered candidates and are the input of the post-processing step. Finally, the post-processing step further improves the linking results by filtering out erroneous matches or inferring new ones.

A plethora of approaches and systems exist, surveyed in [1, 47, 48]. The majority of the linking tools take as an input two RDF datasets and look for identity links between their resources (mostly of arity 1:1) declared as `owl:sameAs` statements.

Note that we make a distinction between linking *tasks*, linking *techniques* and linking *approaches/tools*. A *task* is defined as a subproblem of data linking, originating at a particular need and identified by a particular result that has to be achieved in service of the linking process. For example, "search space reduction" is a particular task aiming at reducing the number of instances to be compared. A *technique* is understood as the means to perform a task – for example, the "clustering" is used to perform search space reduction. Finally, an *approach*, or a *tool* refers to an engineering artefact that performs a given (set of) task(s) by applying and combining a number of techniques.

3.2 Techniques Applied to the Data Linking Process

Various techniques from different computer science fields are adapted and applied to the data linking problem. We outline several of the main groups of techniques, drawing the reader's attention to the fact that a single approach commonly combines several techniques from different groups in its workflow. The techniques presented in the current section are applied to solve different tasks on each of the three main levels of the linking process: preprocessing, (pre-)matching and post-processing. We have organized the techniques by task into four main categories, presented below.

1. ***Search space reduction***. Scalability and computational efficiency are major issues when dealing with data linking problems on the web scale, taking as input datasets with thousands of instances. Search-space reduction is, therefore, a central preprocessing task, which aims at minimizing the

number of comparisons to perform, regarding both resources and their properties. We focus on two main techniques used to achieve this aim. *Keys identification*: Keys identification techniques are well-known from relational databases, where they are used to identify and distinguish records (rows) within a database table. In data linking, for a given class, a key is identified by a property or a set of properties such that there do not exist two or more instances, referring to different real world objects and having the same values for these key-properties. Therefore, two instances, belonging to the same class, across two datasets, can be matched with an "owl:sameAs" link if they share the same values for the class key. *Clustering* [49] aims at forming groups of data items, based on their similarity with respect to their properties. These entities may be of different types: terms, documents or any given data entity represented as a set of features. Web resources sharing similar properties are likely to be identical, and are therefore potential linking candidates. Applying clustering methods on a set of instances from different datasets allows to reduce the search space of matching candidates.

2. ***Instance representation***. Instances need to be given a common representation so that comparison between them can be performed by the help of a similarity measure(s) of some kind. This representation can be done in terms of strings, numerical vectors or other and it often applies a transformation of the original data to enable comparison. The following techniques are of interest for achieving this task. *Linguistic techniques*: These techniques perform linguistic analysis of the textual information describing resources based on knowledge of the language and its structure. Most of the linguistic techniques exploit syntactic, lexical or morphological information.

   - *Low-level preprocessing* Trivially, tokenization, lemmatization and stopword filtering techniques are applied on string values prior to further analysis in order to prepare the data for additional processing or comparison.

   - *Word sense disambiguation* is a sub-problem of natural language processing, which consists in identifying the appropriate meaning of a word with respect to its context. A known example is the word "*apple*", which may refer to the *fruit* or to the *company*. In data linking, commonly, contextual information is explored to assign the appropriate meaning for a term which describes a resource.

   - *Lexical resources exploitation*: A lexicon is a linguistic resource, commonly used in information retrieval, defining terms in structured knowledge bases. For instance, WordNet [39] is a lexical database of English

words where semantic relations between synsets, such as hypernymy or meronymy, are explicitly defined. This allows to perform word sens disambiguation, but also to measure semantic distances between terms. The multilingual lexical databases can be applied to managing multilingualism by describing all RDF entities across two data sets via a single pivot language.

- *Machine translation* is a natural language processing method, which consists in automatically translating a word (or a text) from one language to another. In data linking, applying this technique is very important if we have instances described in different languages. Often textual information is compared across resources and the similarity measures that exploit this information take as an input terms in one single language. Through machine translation, instances, or their textual descriptions, are made comparable for the linking task.

Feature-based techniques [50]: This group of techniques of data representation, used in information retrieval, consists in representing a document (or any other data element for that purpose) in a model by using a set of features that describe this document. At the schema level, a dataset can be represented as a set of index elements. In this context, approaches aiming at identifying relevant data sources in the LOD cloud that contain resources of a given (set of) class(es) or of resources sharing certain properties [51] have been proposed. At the instance level, this technique is used to index each resource by a document of terms (called a *virtual document* or a *pseudo-document*), in which each term is a part of a string literal collected within a given distance to the resource in its RDF graph. In the example in Figure 3.2, the document built from the resource <`http://..../Ludwig_van_Beethoven`> (within a path length distance equal to 1) is composed of four terms (literals): "Ludwig van Beethoven", "composer", "17 december 1770" and "Germany". Each pair of documents is then represented in a feature model. Two documents $D_1 = \{t_1, t_2, .., t_n\}$ and $D_2 = \{t'_1, t'_2, .., t'_m\}$ are represented by vectors such as $D_1 = [v_1, v_2, .., v_o]$ ($o = n$ if $n > m$ or $o = m$ otherwise) and $D_2 = [v'_1, v'_2, .., v'_o]$ ($o = n$ if $n > m$ or $o = m$ otherwise), where v_i (v'_i) represents the weight of the term t_i (t'_i respectively) considering the document D_1 (D_2 respectively). Here, the objective is to provide a similarity comparison in a structure, called *similarity vector*, that will be understood and processed by learning algorithms. For the purposes of data linking, two main feature representations are applied:

- *Vector space model* [52]: For the similarity computation between documents, one of the best known and most commonly used weighting

schemes is TF-IDF (Term Frequency–Inverse Document Frequency). It is based on the frequency of occurrences of a term in a single document, penalized (or boosted) by the number of documents in which the term appears in the whole corpus. To compute the distance between pairs of documents (i.e., resources), several similarity measures can be used. The cosine similarity [53] is the most common one calculating the cosine of the angle between two vectors.

- *Boolean model*: In this representation type, each document D_i is mapped into a vector of binary values indicating the presence or absence of each term in a document. In data linking, this representation is used to built an *inverted index* for resources to be compared.

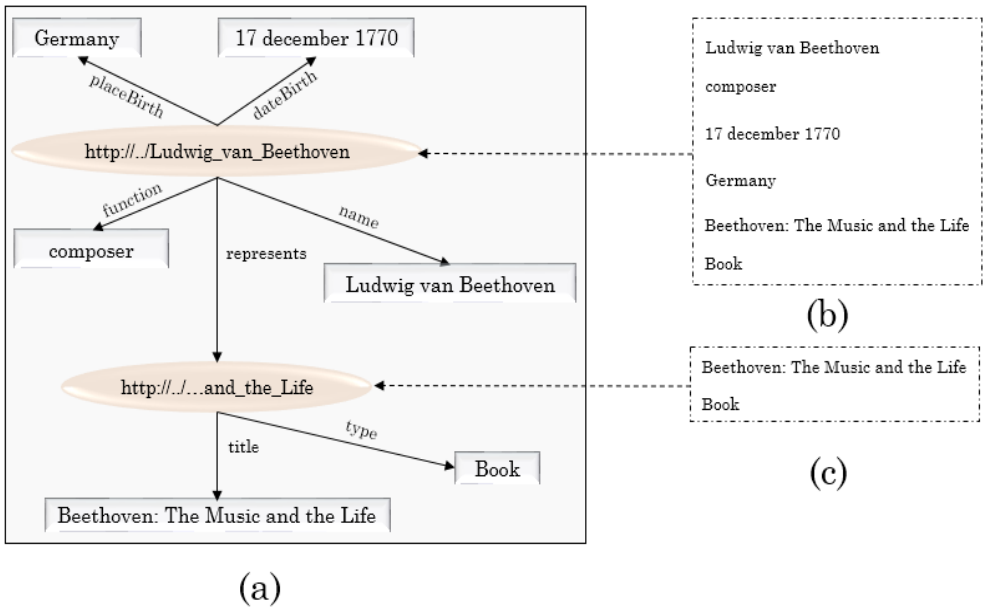


Figure 3.2: Exploring the context of a resource within an RDF graph.
(a) Given an RDF graph, (b) Literals (terms) of a corresponding resource within a distance equal to 2, and, (c) Literals of another resource within a distance equal to 1 are gathered and stored into a document.

3. ***Instance comparison***. Instance comparison is a task that stands in the core of the link discovery process. The goal of this task is to produce a score that quantifies the likelihood that there exist a link of some kind (most commonly "owl:sameAs") between two instances. We outline some of the main techniques applied to this task.
String matching: By using this technique, we compute the similarity D (often with $D \in [0, 1]$) between two strings. The result $D = 1$ means that we have an *exact match* between the two strings. If $D > \sigma$ ($\sigma \in (0, 1)$) then we have

an *approximate matching* between them with D being used a confidence value. In data linking, this technique is commonly used to measure the correspondence of different resources property values, in the case of string literals.
In the following example, we have two RDF triples that describe the same resource, the composer Ludwig van Beethoven:

- $<< http://..../Ludwig\_van\_Beethoven >, foaf:name, \text{“}L.V.\ Beethoven'' >$
- $<< http://..../Ludwig\_van\_Beethoven >, foaf:name, \text{“}L.\ van\ Beethoven'' >$

The string matching algorithm computes the similarity value between the two strings “L.V. Beethoven” and “L. van Beethoven”. For this purpose, several off-the-shelf similarity measures can be used. As an illustration, we present three of the most commonly used measures.

- The *Levenshtein distance* or *Edit distance*[53] is given as the cost (i.e., the minimum number of operations) to transform one string to another. Several edit operations are defined, such as: *insertion*, *deletion* or *substitution*. For instance, the Levenshtein distance between the words $A =$ “$L.V.\ Beethoven''$ and $B =$ “$L.\ van\ Beethoven''$ is 3 and it is computed as follows:
  - Adding the whitespace character between the first character “.” and the character “v” of the word A;
  - Substitution of the second character “.”, of the word A, by character “a”;
  - Adding the character “n” (after the added character “a”) to the word A.
  Levenshtein distance is commonly normalized and scaled into a [0,1] interval.

- The *Jaccard distance* [53] is the ratio between the number of characters in common between two strings on the total number of characters, defined by the formula:

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|},$$

  where A and B are the two strings of interest. For instance, the Jaccard similarity coefficient between the two strings $A =$ “$L.V.\ Beethoven''$ and $B =$ “$L.\ van\ Beethoven''$, seen as sets of characters, is computed as $J(\text{“}L.V.\ Beethoven'', \text{“}L.\ van\ Beethoven'') = 13 \div 14 = O.92$.

- The *Jaro distance* [53] is based on the common characters between two strings A and B. It is defined by the formula:

$$J(A,B) = \frac{1}{3}(\frac{m}{|A|} + \frac{m}{|B|} + \frac{m-t}{m}),$$

where $|A|$ and $|B|$ are the length of the strings A and B respectively. m denotes the number of corresponding characters and t is the ration of their transpositions. In other words, the number of all matching characters (in different sequence order) between the two words defines t. Given our running example, we have $m = 13$ (9 characters are matched) and $t = 0$. Thus, the Jaro distance between the two strings is computed as $(1 \div 3) \times ((13 \div 14) + (13 \div 16) + ((13 - 0) \div 13)) = 0.91$.

Binary classification [49] is a *supervised* learning technique, which consists in assigning elements to one out of two classes based on training data. In data linking, this technique can be used to assign pairs of resources to one of the two predefined categories: (1) pairs of resources to be matched, or linking candidates; (2) pairs of resources considered as different. Provided that sufficient amount of training data is available, this technique can be used to infer the category (match or non-match) of an unseen pair of resources from two datasets.

Graph-based techniques: An RDF dataset is seen as of a graph formed by a number of triples (see Subsection 2.1.1 for a definition and Figure 3.2(a) for an example). In data linking, graph traversal techniques from graph theory [54] are called upon in order to collect information that can be used to describe resources and further compute the similarity between them. For each resource, the information collected according to the graph traversal distance which is greater than 1 (coming from nodes that are not adjacent to the instance of interest) is called *contextual knowledge*. It may be of several types (see Figure 3.3 for more details): URIs, values (literals), <predicate, value> pairs, conceptual or descriptive knowledge. An example is given in Figure 3.2(b) and Figure 3.2(c), where literals at a distance equal to 2 (referring to the resource's *direct* neighbors) and literals at a distance equal to 1 (referring to the resource directly) are collected, respectively. Graph distances, as mentioned above, are also used in order to compute semantic closeness of terms in large semantic networks (such as WordNet).

4. ***Link Validation***. The final step of the linking process includes processing the output data. This consists in improving, in different ways, the quality of the link set produced by a data linking tool. One of the techniques used for this purpose is presented below.

   One-to-one filter. This technique aims to put a restriction on the cardinality of the discovered matchings. It creates a list for each source resource with the corresponding target resource. This list is sorted by the similarity value,

i.e., the resource pair with the best similarity score is selected and the algorithm stops. Where several resource pairs have the same similarity score, the algorithm is re-executed on those resources with other properties and the resource pair with the best similarity score is selected.

3.3 Preprocessing

As discussed in the previous section, the preprocessing step consists in preparing the input datasets for linking. It aims : (1) to reduce the search space, (2) to represent the instances in an appropriate manner for comparison, or (3) to automatically learn the link specification.

1. **Reducing the search space**. Comparing all pairs of instances through all their properties is costly and time consuming. An instance, being described by a set of properties, a number of comparisons of the different property values has to be performed for each pair of instances. It is, therefore, desirable to select the properties to compare and the candidates mappings prior to the data matching task.

   - **Candidates mappings selection.** It is usually based on *clustering* techniques, assuming that instances sharing some properties (keywords for example) may be potentially identical. Identity links occur in each cluster. In other words, the number of inter-cluster *owl:sameAs* links is fixed at *zero*. This is a way to avoid comparing all instances to decide whether they are identical or not.

   - **Properties selection.** It is based on *keys identification*, which consists in discovering sets of properties that uniquely identify the resources. This is a way to avoid comparing all property values to decide whether two resources are equivalent or not. The comparison task is applied at the data matching step where instances sharing the same values for a key, which may consist of one or more properties, are considered as equivalent. In what follows, we present approaches based on keys identification that act on the preprocessing step only.

     Atencia *et al* [31] proposed an approach which is based on two measures for keys detection. The approach consists in determining whether a set of properties is a key for the particular data set. However, because of erroneous and redundant data, the authors extended the key definition to that of a *pseudo-key*, defined as a set of properties that identifies *most* of instances in a RDF dataset. For this purpose, the measures

of discriminability and support are introduced. The *discriminability* is defined as the ratio between the number of instances sharing identical values (for this key) to the support of the key. The *support* is the proportion of individuals having all the predicates of the key instanciated. A set of properties is considered as a pseudo-key if its discriminability value is greater than a given threshold.

Similarly to [31], the notion of *almost-key* is proposed by Symeonidou *et al.* [8] to describe sets of properties that fail to be keys due to few exceptions. To filter data, the authors propose to discover first the maximal non keys and use them to derive the minimal keys. The approach is implemented in the tool SAKey (Scalable Almost Key discovery) and proceeds in three main steps. The first step allows to eliminate irrelevant sets of properties. Based on pruning strategies, the second step allows for the discovery of $(n + 1)$ maximal non-keys, i.e., $n + 1$-non keys that are not subsets of other $n + 1$-non keys for a fixed n. In the third step, the algorithm derives the almost keys from the set of $(n+1)$ non-keys, i.e., all the sets of properties that are not maximal $(n+1)$-non keys are n-almost keys.

In the same way, KD2R [37] discovers the maximal non-keys to infer keys. The authors introduce the notion of *undetermined keys* to designate a set of properties that are not a non-key. There are at least two instances that share same values for a subset of undetermined keys. The remaining properties are not instantiated for at least one of the two instances. In other words, an undetermined key defines a set of properties that cannot be considered neither as keys neither as non-keys due to the lack of information. Therefore, the authors introduce the optimist and the pessimist heuristics, implying the consideration of this set of properties as a key or not, respectively. To determine the set of keys in RDF datasets, the algorithm starts by presenting the instances of a given class in a structure called *prefix-tree*. This structure is used to discover the set of maximal undetermined keys and the set of maximal non-keys. The minimal keys are then derived from the previous sets. The algorithm is iterative and is applied for each class of a given ontology.

Unlike SAKey [8], ROCKER [9] considers two resources as distinguishable with respect to a set of properties P even if they share one (or more) object(s) for each $p \in P$. It relies on a scoring function to compare sets of properties and allows the discovery of keys or almost-keys within a given threshold (i.e., a set of properties is a non-key if its score

is less than 1). The score function expresses the number of subject resources that are distinguishable with respect to a set of properties. The authors combine the characteristics of the refinement operator (i.e., prune the refinement tree) with the key monotonicity property to obtain a time-efficient approach for detecting keys. In particular, using monotonicity of keys allows to check for the existence of keys as well as decide on nodes that need not be refined.

2. **Instance representation**. Various kinds of transformations on the original data can be applied to make the resources comparable. These transformations may be the result of linguistic analysis where the entities are described in different languages, or numerical where the entities are represented as vectors based on computed scores. The linguistic representation often uses external resources, such as BabelNet [55] or WordNet [39], to perform the data transformation.

2. **Learning of link specification.** Another way to prepare the data is to specify the conditions that all data items must fulfill in order to be compared. These conditions are defined in a so-called *link specification*. It is defined by [56] as: (i) the setting of the elements $\mathtt{S}$ and $\mathtt{T}$ to compare from two knowledge bases $\mathtt{K_S}$ and $\mathtt{K_T}$ respectively, (ii) the setting of a complex similarity metric via the combination of several atomic similarity measures, and (iii) the setting of thresholds for the similarity measures. Note that a *link specification* can either be set by the user or learned from labeled training data [57]. The challenges issues with regard to manually define link specification have witnessed recent interest and have been addressed employing machine learning [58] whether supervised [15, 16, 59] or not [60].

   The authors in [60] proposed an unsupervised learning approach of the linking specification implemented in KnoFuss [7]. The approach is based on genetic programming to learn iteratively the optimal similarity parameters. However, this approach still require setting the fitness function and the specification of fitness measures, thresholds and the maximum number of iterations which does not guarantee an optimal solution.

   Supervised machine learning to compute link specification requires a domain expert to label a set of candidate links. With respect to this, two main categories emerged: *active* [16, 56, 61] and *batch* [15, 59] learning approaches. The *batch* approaches require a large amount of candidate links as input to learn the classifiers while the *active* approaches proceed iteratively and for each iteration the user is required to label a set of generated links. For example, the data linking tool LIMES [5] includes EAGLE [16] as well as Silk

[6] includes ActiveGenLink [15], semi-supervised active learning approaches based on genetic programming. It proceeds iteratively until the maximal number of iterations is reached or the fitness value is greater than a given threshold. In each iteration it asks the user to label (valid or decline) the generated links.

3.4 (Pre-)Matching

The *(pre-)matching* step is at the very heart of the linking problem. It aims at finding instances referring to the same real world object by using appropriate instance similarity measures. In the semantic web field, the data linking process operates at different **levels** depending on which piece of information the comparison between instances is done. In fact, we have identified in the literature two levels of comparison -an *intensional* and an *extensional* level (see Figure 3.3). In this context, the *intensionality* means information that describes implicitly a resource. On the other hand, the *extensionality* covers information that describes explicitly a resource. For the linking task, a question is raised around this issue: ***between which pieces of information the comparison is done ?*** Whether at *extensional* or *intensional* levels, resources can be compared based on several types of information that define them.

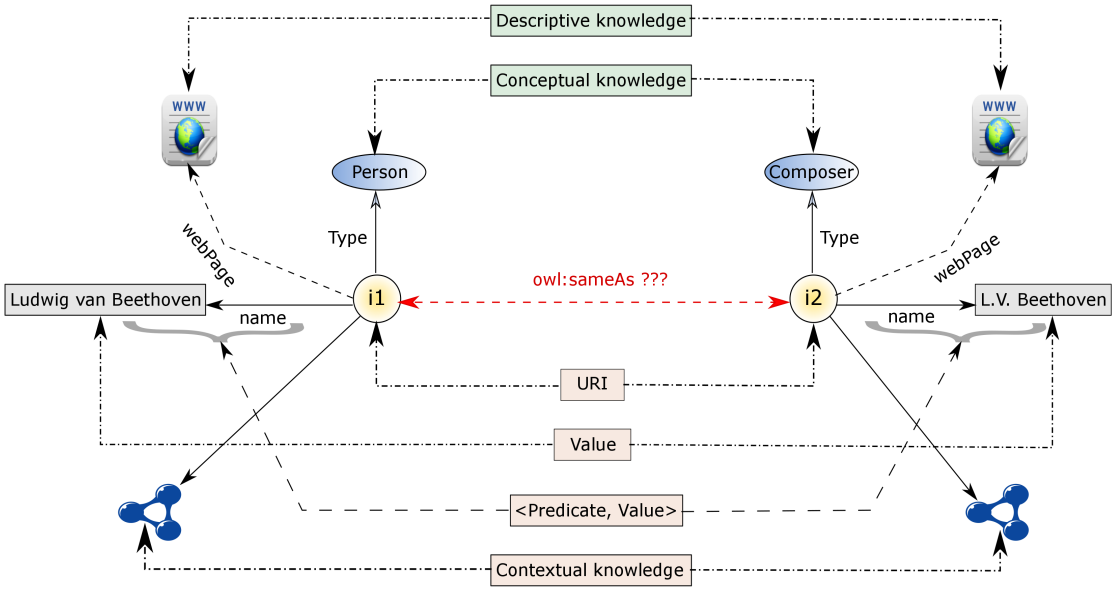


Figure 3.3: Data matching levels.

1. Linking at **extensional level** consists in generating identity links based on explicit information. In this case, links can be established, between resources, comparing their:

   - ***URIs***: The idea is to interlink entities comparing their URIs. Links can be established if the last fragments of their URIs are identical. For example, we find that the two resources I_1= `http://dbpedia.org/page/Ludwig_van_Beethoven` and I_2= `http://yago-knowledge.org/resource/Ludwig_van_Beethoven` in DBpedia [62] and Yago [63] represent the same entity.

   - <***Predicate, Value***> ***pairs***: To interlink two resources, some approaches compare their literals attached to given properties. In other words, two resources are linked together if they share the same values for all or a subset of their properties identified as a key (see Section 3.3).

   - ***Values***: Here, the idea is to compare the literals, which are at a distance[1] $n = 1$ to each of the two resources (the literals are directly attached to the resources to be compared) regardless to the properties.

   - ***Contextual knowledge***: In some cases, especially in the presence of property chains, information related to *neighboring* resources in the RDF graph is used, because literals directly related to them are not sufficient to decide whether these resources are identical. In other words, two entities are compared based on the literals which are at distances 1 to n to the resources with $n > 1$ in the RDF graph.

2. Linking at **intensional level** generates identity links based on information derived from instance's *descriptive knowledge* or *conceptual knowledge*.

   - ***Descriptive knowledge***: Generally, approaches relying on this type of information exploit the words contained in the descriptions of two resources to decide whether they are likely to be linked. This means that two entities sharing the same words in their description (for instance, the corresponding wikipedia pages) are considered to be linking candidates.

   - ***Conceptual knowledge***: The types of instances carry important knowledge of their similarity and are often used to filter out candidates both in the preprocessing and the matching steps. Indeed, the semantic equivalences between data items can be discovered only if these items

[1] A distance is defined over the nodes (resources) of a (RDF) graph as the minimal number of edges connecting two nodes (resources).

belong to the same class (i.e., their concepts match). Many approaches have been proposed where the problem is referred to *concept matching*. Note that this paper is devoted to survey only data linking tools and approaches. Hence, no approach will be assigned at this category. Another solution to linking issue is to group entities, belonging to the same class, together. Approaches relying on conceptual knowledge determine correspondences between ontological concepts before establishing links. This means that resources sharing the same label of their concepts are assigned to the same cluster (where these resources are considered to be candidate for the (pre-)matching task).

Note that some categories will not be illustrated in the remainder of this subsection as is the case of descriptive knowledge-based approaches that may perform other steps of the overall linking process (see Section 3.6). In the following, we briefly describe approaches proposed in [64], [65] and [66] acting only at this stage and they can be based on several pieces of information for the comparison task.

To illustrate the interlinking process based on ***contextual knowledge*** and on the ***similarity between values***, we present LD-Mapper [64], an interlinking system of musical datasets, which is based on the similarity between the resources in the RDF graph and the similarity of their *neighbors*. The system starts by computing the *similarity values between literals* of all pairs of resources. Then, it combines a measure (aggregation of all the similarity values between resources pairs) for each graph mapping. The graph mappings are all the possible combinations between the resources of the two graphs. Finally, the mapping of which the similarity value is the largest will be selected by the algorithm.

In the same category, the approach proposed in [65] is based on ***contextual knowledge*** and on the ***similarity between values***. It is a user profiles interlinking approach of different social networks such as Facebook, Twitter and MySpace. The idea consists in constructing RDF graphs from XML files of the social networks. These RDF graphs will then be interconnected based on the *user identifiers* of each social network. Data linking uses several similarity measures based on the graphs in order to create *owl:sameAs* links.

In the conference/university field, a coreference resolution system called RKB-CRS [66] has been introduced, based on the ***similarity between values***. It consists in establishing a list of equivalent URIs (resources). For each datasets pair, a new program has to be written. This program selects the resources to be aligned and it compares them using *similarity measures*. RKB-Explorer [2] uses this system to store, manipulate and reuse coreference data.

[2] http ://www.rkbexplorer.com/explorer/

3.5 Post-processing

In this sub-section, we present approaches operating only once data is interlinked. They aim to evaluate the existing links in a dataset. As stated in Sub-section 1.2, the *identity* links are crucial for integrating data, inferring new ones or navigating between the resources in different RDF graphs. Furthermore, they also can be used to detect erroneous literal values [67, 68]. For example, the approach proposed by [68] aims at identifying wrong numerical values in Linked Data using the outlier detection technique.

It is important to notice that any change in the data may: (i) invalidate certain links when modifying the data (*invalid links*); (ii) destroy them when removing the data (*dead links*); or (iii) require to generate new ones when adding the data (*new links*) [69]. The task of keeping existing links in a valid state whatever the changes in the data is called *mapping maintenance* [70]. The fast growth in the amount of Linked Open Data[3] raised new challenges for the solution of the *mapping maintenance* problem. Hence, in recent years, an increasing number of works have shown interest in the maintaining of good quality links [69, 71–77]. These approaches have been proposed in order to detect change events in the datasets supporting to maintain the links between them. For exemple, the data linking framework Silk [6] proposes WOD-LMP (Web of Data - Link Maintenance Protocol) for synchronizing and maintaining links on data change. As well as DSNotify [73], WOD-LMP periodically accesses a dataset via SPARQL queries which does not require a persistent maintenance of these links. SDValidate approach [75] proved to be effective improving the quality of RDF statements in DBpedia [62]. In fact, applying this approach, 13.000 erroneous links have been identified and removed from the knowledge base. However, the *mapping maintenance* approaches try to assess the correctness of any type of statements; while in this work we focus on `owl:sameAs` statements.

In [78], the authors proposed an unsupervised approach for identifying wrong *identity* links between datasets. The approach is based on outlier detection methods [79–84] representing the links as multidimensional feature vectors and assigning them outlier scores. The links that are far from the overall distribution in the feature space are considered as erroneous.

A **partitioning method**, proposed by [85], detects the existing erroneous links in a bibliographic knowledge base where each bibliographic record (i.e., a document describing a book for instance) is connected to one or several authority records (i.e., where each of them describes a person who edited or wrote the book). The

[3] `http://lod-cloud.net/#history`

main idea is to discover if these existing links are correct. The concept of partition, for a set of objects, is defined as a set of classes such that each object belongs to only one class. Their global method proceeds as follows:

1. Construct contextual authorities where each is composed of an authority record with one of the bibliographic records pointing to it. It corresponds to an author with one written book.
2. Partition the set of contextual authorities according to different criteria (title, date of publication, domain). Each partition makes sense from the point of view of the respective criteria.
3. Aggregate criteria to decide whether these contextual authorities represent or not a same person using two proposed semantics.

3.6 Multi-step methods

Multi-step methods, as mentioned above, combine two or three steps of the data linking workflow.

The SERIMI system [17] operates in two steps: selection and disambiguation. In the selection phase, SERIMI extracts, for each instance in dataset A its label (identifier) and looks for instances in the dataset B that have a similar label according to a given threshold (i.e., retrieve target candidate resources with a string similarity greater than 70%, as fixed by the authors). This step outputs a set of *pseudo-homonym* resources for each source resource. As distinct instances may share the same label, the disambiguation phase will allow to select the appropriate instances among the set of *pseudo-homonym* resources. Indeed, the disambiguation phase consists in filtering the instances found in the dataset B and keeping only those that identify the same real world object as the resources in the dataset A. As the class of the target resources is not known, SERIMI selects the resources that belong to the *class of interest* to filter instances in the *pseudo-homonym* sets. The authors define the concept of *class of interest* as a set of attributes that instances may share in common. The idea is to classify a target resource as belonging to a given *class of interest* by comparing it to all the other *pseudo-homonym* sets. Finally, the system selects all resources with a score greater than a given threshold.

To tackle the multilingualism problem, Lesnikova *et al.* [34] proposed a linking method using indexing and NLP techniques. The method consists in creating a document (accumulation of data collected in the graph traversal) for each URI. All documents are automatically translated into a pivot language before computing

similarity between them. The same authors proposed a similar approach, which relies on the use of a ***multilingual*** lexical resource (BabelNet [55]) instead of a machine translation technique [35]. Once the documents are constructed, the algorithm replaces each term by the corresponding identifier (ID) from BabelNet. There may exist many senses (IDs) per term. If it is the case, word sense disambiguation techniques are applied in order to select the most appropriate sense. After preprocessing, the authors index the documents by using the vector space model to represent the documents as word feature vectors. They compute a set of similarity values between pairs of documents by using the standard term weighting scheme TF-IDF and applying the cosine similarity. Resources are then linked based on the produced similarity scores between the documents.

Indexing techniques can also be used to reduce the number of instances comparison. Indeed, Rong *et al.* [21] developed a system, which extracts in the first step literal information from entities. The literal information $l = \{l_1, l_2, ..., l_n\}$ is similar to a document generated from an instance. Then, it uses a vector space model to represent this information. An inverted ***index*** is built for instances of some keywords in their descriptions. The instances sharing the same keywords in their index are considered to be linking candidates. Then, the algorithm computes the similarity vectors for the candidate instance pairs. The authors propose a ***feature vector*** of similarity metrics. To train a ***binary classifier*** based on the similarity vectors, they are labeled into two classes: *non-matching* and *matching*. The existing links in the LOD cloud can also be used to train the classifier. The authors evaluate the proposed approach on the datasets of OAEI 2010 showing that the approach performed better than the other paricipants.

To prepare the instances for the linking task, RiMOM-IM [18] unifies languages and/or formats in which data are expressed, removes stop words, or computes the TF-IDF of words composing the predicate values. After that, RiMOM-IM applies a blocking technique which consists in using inverted indexing to generate candidate sets and unique instance sets. It takes the predicate and the top five words of the object (ordered by TF-IDF values) as index keys of instances. For each pair in the candidate set, it computes similarities over all aligned predicates and then aggregates them to get the final matching score of two instances. It iteratively selects the pair with the highest score (above a given threshold) as the aligned pair. It will be used to infer new candidate pairs without computing the similarities until no new aligned instances are generated, by using two strategies: *unique subject matching* and *one-left object matching* (see [18] for detailed descriptions).

To avoid exhaustive pairwise comparisons of instances, a blocking step is also applied in [19] by grouping together properties. Unlike RiMOM-IM, two instances are in the same cluster if they share tokens of the labels of any two properties that

were grouped together. The authors propose to apply the block purging algorithm which eliminates clusters larger than a threshold value, with the premise that such clusters are the result of stop-word tokens. Once the candidate set is generated, the system tries to ensure that only compatible instance pairs are evaluated by generating restriction sets (matching classes and properties between two RDF files). Each compatible instance pair is represented by the help of the boolean model where the values are equal either to 1 if the instances share a common token or to 0 otherwise. A binary classifier is trained on these data and applied to discover new links.

To prune the search space, Silk [6] implements indexing and entity pre-selection methods. The pre-selection consists in finding a limited set of target entities that are likely to match a given source entity. All target resources are indexed by one or more specified property values (most commonly, their labels). The `rdfs:label` of a source resource is used as a search term into the generated indexes and only the first target resources found in each index are considered as link candidates for matching. This strategy does not ensure the identification of all equivalent resources in the target dataset. Silk is based on user link specification (Silk-LSL). In other words, it features a declarative language for specifying which types of RDF links should be discovered between data sources and which conditions entities must fulfill in order to be interlinked.

As Silk, LIMES [5] is configured by using a link specification language. It is based on the mathematical characteristics of metric spaces to compute the similarity between instances. In particular, it utilizes the *triangle inequality* to reduce the number of comparisons and therefore to reduce the time complexity of the mapping task, which is one of its major advantages. By these means, LIMES partitions the (instance) metric space by representing each of these portions by an *exemplar* that allows to compute an accurate approximation of the distance between instances based on already known distances. Due to the considerable gain in efficiency provided by the tool, LIMES is capable of linking very large datasets, where other tools usually fail. However, LIMES assigns for each source resource at most one target resource. A dataset may contain duplicates while it does not generate mappings `1:n`.

Nguyen *et al.* [22] propose an approach that also combines preprocessing with data matching. The authors introduce the notions of coverage and discriminability to define a property as a key. The *coverage* of a property is defined as the ratio of the number of instances having that property to the total number of instances. The *discriminability* is the ratio of the number of distinct values for the property to the total number of instances having that property. The instances that have similar literal values for the candidate selection key are linked together.

RDF-AI [20] is a semi-automatic tool which acts on the three main data linking steps. It takes as an input two RDF datasets and two XML configuration files and generates as an output either a new dataset resulting from the fusion of the two input datasets, or a list of correspondences between equivalent resources of the two datasets. The XML configuration file specifies the preprocessing operations to perform (consistency checking, properties translation, linking techniques, properties transformation) for each resource. Indeed, the preprocessing step generates two datasets processed by the user operations. The second configuration file describes the post-processing parameters such as the correlation threshold and the fusion parameters. The system includes two similarity computation algorithms: a string matching algorithm and a word relations algorithm. There are two implementations to the latter: a synonyms comparison algorithm based on WordNet and a taxonomical similarity algorithm based on the SKOS vocabulary. Finally, the system checks the inconsistencies (for example, breaking ontology axioms) that may appear as a result of the linkset or of the fused graph.

Like RDF-AI, KnoFuss [7] takes as an input two datasets with their respective ontologies by specifying the resources to be compared and the comparison techniques to use. It is also able to merge two datasets using existing ontology alignments in case when the two datasets are described by different ontologies. In this context, *ontology matching* allows to translate the SPARQL queries that are used to select the appropriate instances and the comparable properties from the terms of one scheme to another. In this way, the instances are compared in the same manner as if they were described by the same ontology. Just like [20], KnoFuss includes a post-processing step to check the inconsistencies of the merged datasets.

AML's matching pipeline [86] includes a matching step and a filtering step. In the former, AML combines four matching algorithms in order to generate the mapping candidates: (1) *HybridStringMatcher*: for a given pair of individuals, it computes the maximum similarity between their entries in the Lexicons, (2) *ValueStringMatcher*: for a given pair of individuals, computes the maximum string similarity between their entries in the ValueMaps, (3) *Value2LexiconMatcher*: it compares the lexicon entries of one individual with the ValueMap entries of the other individual, in order to account for cases where a property is not detected as a "name", and (4) *ValueMatcher*: it identifies the individuals that have the exact same value and scores their similarity as the inverse of the total number of individuals that have that value for that property. In the filtering step, the similarities of all of the matchers above are computed and combined for each mapping candidate, in order to select the final set of mappings.

3.7 Discussion and comparison of the tools and approaches

In this Section, for sake of clarity, we present a classification of somes approaches and tools depicted in Figure 3.4. We organize them in a (pseudo-) taxonomy with respect to the three major steps of the matching process (*preprocessing*, *data (pre-)matching* and *post-processing*), splitting them further into several categories according to the tasks that each approach adresses and finally according to the techniques that are applied. We additionally consider a fourth, *multi-step* category of methods – those that act on more than one step of the matching process (they can be found on multiple leaf-nodes of our taxonomy). In order to evaluate and compare the data linking tools and approaches, we define several criteria allowing to highlight the specificity of each of them. The criteria are described hereafter:

Domain. Certain tools are developed for datasets of a specific field (music, library or conference) while others are generic. The interest here is whether the domain has an impact on the interlinking results or not.

Input. This criterion specifies the type of input data. Indeed, as we have seen, we are interested in this paper to different data formats. The goal is to address similar issues as interlinking RDF data.

Output. The output data type differs depending on the input data type (RDF data or articles) or on the stage (preprocessing, data matching or post-processing) on which focuses the tool.

Preprocessing. This criterion specifies whether the tool applies a treatment before the interlinking phase.

Data matching. As shown in this paper, some tools have been developed just in order to identify keys among a given dataset. Indeed, even if our work focuses on data linking tools, those presented in this paper were not necessarily conceived to discover identical entities. This criterion specifies whether the tool performs the linking task or not.

Post-processing. This criterion specifies whether the tool applies a treatment after the interlinking phase.

Techniques. This criterion specifies the techniques used by the tool to be compared.

Multilingualism. Any entity matching tool must necessarily dealing the multilingual RDF datasets. In fact, nowadays, RDF data are expressed in 474

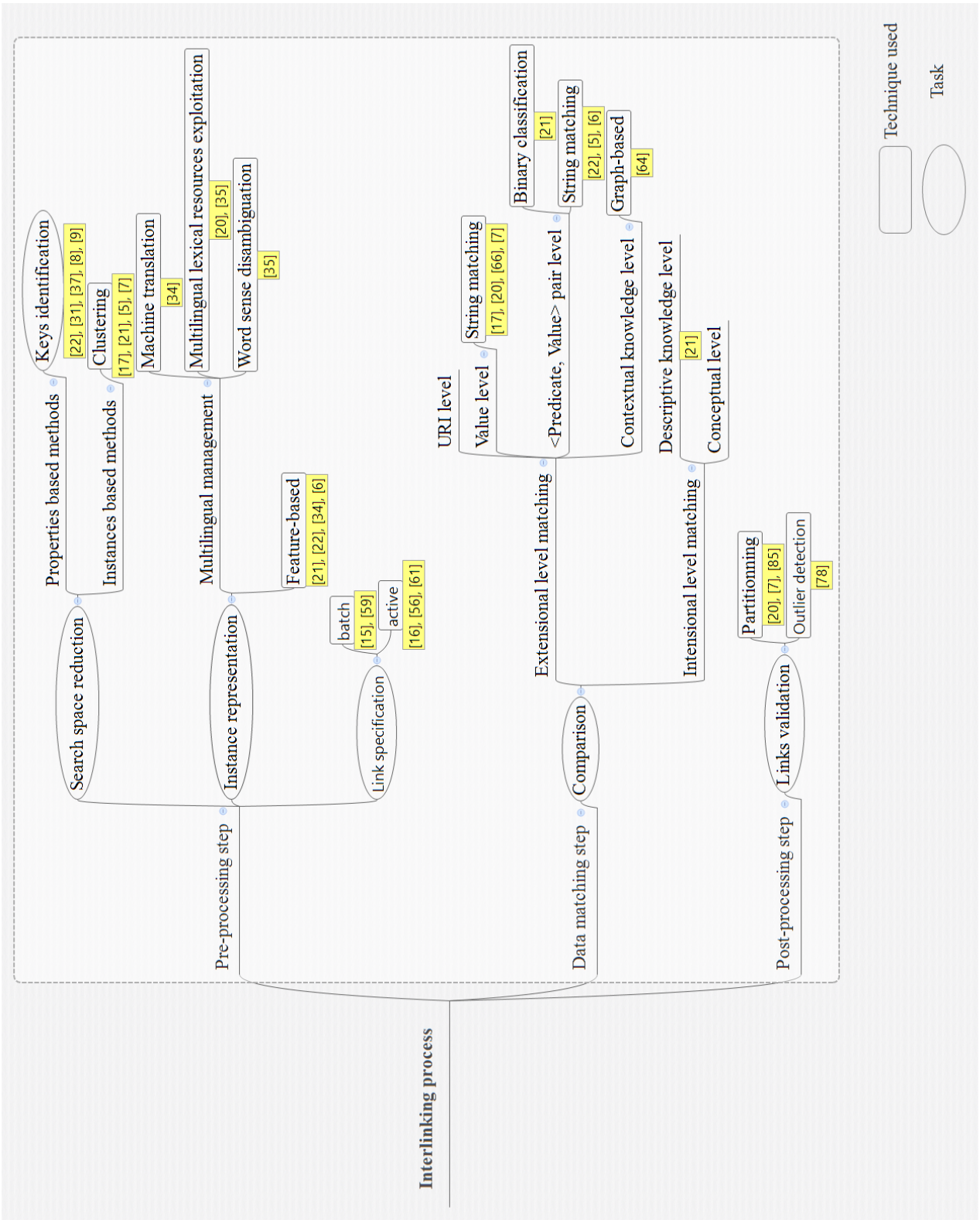


Figure 3.4: Classification of tools/approaches according to the three main data linking steps, to the tasks and the techniques

languages[4] on the LOD. Therefore, the lingual heterogeneity occurs mainly at the value level. This criterion specifies whether the tool distinguishes the multilingual aspect between the RDF data.

Automation. In addition to the results that the linking tools produce, they should also be compared according to their degree of automation. Indeed, the user intervention may be required before (configuration stage) or after (validation stage) the entities are linked. This allows to give increased costs with respect to the computing time. This criterion specifies the degree of automation of the compared tools.

Approaches	Data Linking Step	Technique	Input	Output	Domain	Multilingualism	Degree of Automation	Evaluation
Atencia et al. [31]	Preprocessing	Keys Identification	RDF datasets	Keys set	LOD	No	/	No
SAKey [8]							Automatic	
KD2R [37]								
ROCKER [9]								
EAGLE [16]		Link Specification Learning		Link specification			Semi-automatic	
ActiveGenLink [15]								
SLINT [22]	Preprocessing Matching	Keys Identification Blocking String Matching		Linkset			Automatic	
SERIMI [17]		Clustering String Matching						Yes (2011)
Lesnikova et al. [35]		WSD Multilingual Lexical Resources Feature-based				Yes	/	/
Rong et al. [21]		Feature-based Clustering Binary classification				No	Semi-automatic	No
Lesnikova et al. [34]		Machine translation Feature-based				Yes	/	/
Raimond et al. [64]	Matching	Graph-based	RDF datasets + link specification		Music	No	/	/
SILK [6]		Link Specification			LOD		Semi-automatic	No
LIMES [5]								
RDF-AI [20]	Preprocessing (Pre-)matching Post-processing	Machine translation String matching Inconsistency checking		Linkset/ Merged datasets				
KnoFuss [7]	(Pre-)matching Post-processing	String matching One-to-one filter		Linkset				
Paulheim et al. [78]	Post-processing	Outlier detection	RDF datasets	RDF datasets			/	/
Guizol et al. [85]	Post-processing	Partitioning	Bibliographic records	Link validation	Bibliographic knowledge		/	/

Table 3.1: Summary table of the main data linking tools/approaches.

Table 3.1 provides a comparison between the surveyed approaches classified according to the three steps of a data linking process, i.e., preprocessing, data linking and post-processing steps. As we can see, each approach can perform more than

[4]http://stats.lod2.eu/languages

one step. We refer to such approaches as hybrid. For each step of a matching process, an approach performs a set of techniques. Moreover, each approach is described by its principle (inputs and outputs), the specificity of its application to a particular field (music, LOD), its management of multilingualism, and in the case where it is implemented in a tool: its degree of automation and its participation in the Ontology Alignment Evaluation Initiative[5] (OAEI) campaign.

We can notice that string matching techniques are used by every approach for data linking step as they are the obvious techniques for comparing resources values. For preparing data in the preprocessing step, the used techniques differ according to the purpose of the approach that can be either *search space reduction* or *representing the resources* in an uniform manner. However, it seems clear that there is few approaches performing the final step of post-processing. This can surely be explained by the fact that the most of approaches trust their process of matching. The same remark applies, in fact, to multilingualism where only few approaches tackle this issue. However, it is obvious that this criterion presents a crucial problem which shall be resolved particularly to match equivalent resources described in multi-languages.

3.8 Conclusion

In this chapter, we have provided a broad overview of the existing data linking approaches. We classified these approaches according to the step on which they focus and the techniques used in each step.

We have seen in this study that there are as many *single-step* approaches as *multi-step* approaches (combination of several steps).

However, there are very few works that focus on the *post-processing* step. Furthermore, there is no generic approach to tackle the data linking issue from *preprocessing* to *post-processing* step. Also, the *multilingualism* proves to be an important issue to be taken into account in order to identify the same resources across multilingual RDF datasets and interlink them. However, this study has shown that only few approaches tackle this problem. We note that many of the linking tools require a laborious step of configuration, by specifying manually properties, types, similarity measures and other parameters before launching the tool. Automation of the configuration step appears to be a pressing problem in the field. Finally, while there exist multiple generic linking tools/approaches, there is just a few works dedicated to interlinking specific type of data. In fact, the authors of [64] seem

[5] `http://oaei.ontologymatching.org/`

to be the only ones to propose an approach for linking musical datasets. *Does the domain specificity of data affect the results produced by a generic linking approach*? We leave this as an open question which has to be answered by evaluating linking tools on different domain specific datasets.

Chapter 4

Automatic Key Selection using RANKey

Contents

Introduction

Due to the large amount of data already available on the web, defining manually `owl:sameAs` links would not be feasible. Therefore, many approaches try to answer to this challenge by providing different strategies to automate this process. Datasets conforming to different ontologies, data described using different vocabularies, datasets described in different languages are only several of the examples that make this problem hard to solve.

Many of the existing link discovery approaches are semi-automatic and require manual configuration. Some of these approaches use *keys*, declared by a domain expert, to link. A key represents a set of properties that uniquely identify the resources (see Sub-section2.1.3). Keys can be used as logical rules to link data ensuring high precision results in the linking process. Additionally, they can be exploited to construct more complex rules. Nevertheless, keys are rarely known and are very hard to declare even for experts. Indeed, experts may not know all the specificities of a dataset leading to overlook certain keys or even introduce erroneous ones. For this reason, several automatic key discovery approaches have been already proposed in the context of the Semantic Web [8, 9, 31, 37, 87].

In spite of that fact, applying the output of these approaches directly is, in most of the cases, impossible due to the characteristics of the data. Ontology and data heterogeneity are not the only issues that can arise while trying to apply keys directly for data linking. Even if the datasets conform to the same ontology and the vocabulary of the properties is uniform, this does not ensure the success of the linking process. Very often, key discovery approaches identify a very large number of keys. The question that arises is whether all the keys are equally important among them for the linking quality, or there are some that are more significant than others. In this chapter, we propose a strategy to rank the discovered keys, by taking in consideration their effectiveness for the matching task at hand.

Bridging the gap between key discovery and data linking approaches is critical in order to obtain successful data linking results. Therefore, in this chapter we propose a new approach that, given two datasets to be linked, provides a set of ranked keys, valid for both of them. We introduce the notion of "effectiveness" of a discovered key. Intuitively, a key is considered effective if it is able to provide many correct `owl:sameAs` links. In order to measure the effectiveness of keys, a support-based key quality criterion is provided. Unlike classic approaches using support for the discovered keys, in this work we introduce a new global support for keys valid for a set of (usually two) datasets.

4.1 Problem Statement

Given two RDF datasets, candidates to be linked, our approach aims at ranking the keys that are valid for *both* of them. These keys can be used successfully as link specifications by link discovery frameworks. Notice that the definition of a key given in Sub-section 2.1.3 is only based on distinguishing resources coming from the *same* dataset. In this section, we introduce the definition of a key for two datasets. Formally, it is defined as follows.

$$\forall \mathtt{r}, \forall \mathtt{r}', \wedge C(\mathtt{r}) \wedge C(\mathtt{r}') \wedge \mathtt{r} \in subj(G) \wedge \mathtt{r}' \in subj(G') \bigwedge_{i=1}^{n} (p_i(\mathtt{r}, o_i) \wedge p_i(\mathtt{r}', o_i)) \Rightarrow \mathtt{r} = \mathtt{r}', \tag{4.1}$$

Where `r` and `r'` are resources of the graphs G and G' respectively and they are of the class C. $p_i(\mathtt{r}, o_i) \wedge p_i(\mathtt{r}', o_i)$ expresses that both `r` and `r'` share the same value o_i for every property p_i in the key.

In the next section, we describe how do we select keys that are valid for two datasets. Afterwards, we describe our ranking approach on the set of these keys.

4.2 RANKey: Automatic Key Selection for Data Linking

The number of available vocabularies has been growing with the growth of the LOD cloud, resulting in datasets described by a mixture of reused vocabulary terms. It is therefore often the case that two different datasets to be linked are described by different vocabularies. This makes the comparison more complicated if not impossible. To answer to that, ontology alignment methods [88] are used in order to create mappings between vocabulary terms. In this work, we assume that equivalence mappings between classes and properties across two input datasets are declared (either manually, or by the help of an ontology matching tool). These mappings will be used to obtain keys that are valid for *both* datasets.

4.2.1 RANKey Overview

Algorithm 1 gives an overview of the main steps of our approach called RANKey, also depicted in Figure 4.1. Overall, given two datasets to be linked, this algorithm returns a set of ranked keys valid for both datasets. In addition to that, every

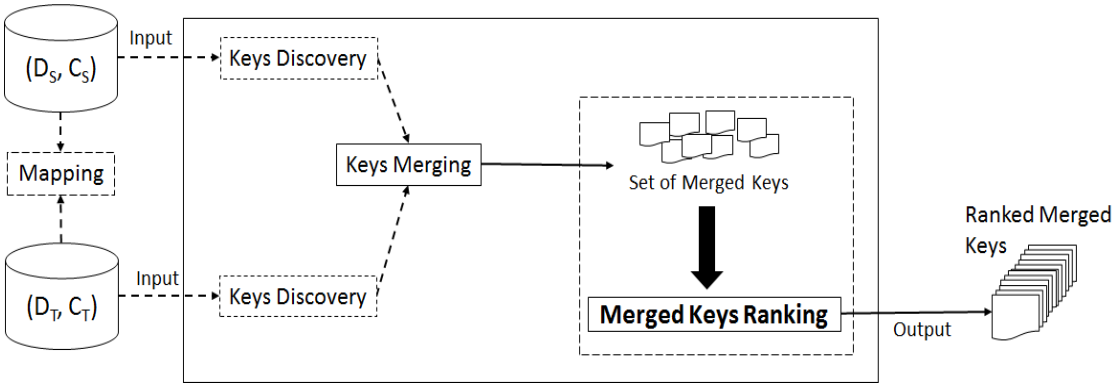


Figure 4.1: The processing pipeline of Algorithm 1.

proposed key is given a score, allowing to rank keys according to their effectiveness for the data linking process. This process is described step by step below.

First, given the datasets D_S and D_T, a set of property mappings M between the two datasets is computed. As described in [37], property mappings allow the identification of properties that belong to both datasets simultaneously. We assume that the instances are of the same class C. If this is not the case, we apply ontology alignment methods.

A key discovery step is applied to both datasets independently allowing the discovery of valid keys in each dataset. Only mapped properties, appearing in M, will be contained in the discovered keys. For this step, existing key discovery tools such as SAKey [8] or ROCKER [9] can be used to obtain keys for a given class C.

However, even if keys consist of properties that belong to both datasets, nothing ensures that the discovered keys found in each dataset independently will be the same. Indeed, there can be cases where something found as a key in one dataset is not in the other. Since key discovery approaches learn keys from the data, the generality of each dataset affects the generality of the discovered keys. For example, if a dataset contains people working in a specific university, it is possible to discover that the last name is a key. Thus, to deal with this challenge a merging step is performed. Indeed, merging keys coming from different datasets allows to verify the validity of discovered keys and to obtain more meaningful keys since they are applicable to more than one datasets. Different strategies for key merging could be applied. In this work, we apply a merging strategy proposed in [37] providing minimal keys valid in both datasets.

The result is a set of merged keys considered as valid for *both* datasets. However, the number of merged keys produced by the algorithm can be significantly high, which makes manual selection difficult, particularly in the lack information of the

keys suitability for the data linking task. Therefore, we introduce a novel ranking method for merged keys to identify the most suitable keys to be used in the link specification, introduced in next section.

Algorithm 1 The merged keys ranking algorithm

Input: D_S and D_T, a pair of datasets candidates to be linked
Output: A set of merged and ranked keys: *rankedMergedKeys*
1: $M \leftarrow$ Mapping(D_S, D_T)
2: $KeysD_S \leftarrow$ keysDiscovery(D_S, M)
3: $KeysD_T \leftarrow$ keysDiscovery(D_T, M)
4: *MergedKeys* $\leftarrow$ keysMerging($KeysD_S$, $KeysD_T$)
5: *rankedMergedKeys* $\leftarrow$ mergedKeysRanking(D_S, D_T, *MergedKeys*)
6: **return** *rankedMergedKeys*;

4.2.2 Merged Keys Ranking

As described before, the merged keys are valid for both datasets. However, these keys may vary in terms of "effectiveness" in the linking process. Therefore, we propose to first to assign a score reflecting the "effectiveness"[1] of a discovered key and second use this score to rank the discovered keys among them.

In general, it is very common that not all the properties are used to describe every instance of a given class. This happens often due to the nature of the property or the incompleteness of the data and may have significant impact on the quality of the discovered keys with respect to the linking task. While many properties apply to every instance of a class, there exist cases of properties that have values only for certain instances (the property "spouse" for a person applies only to people that are married). In addition, in the case when data are incomplete, an instance may not have a value for a specific property even if a value exists in reality. This can lead to the discovery of wrong keys since not all the possible scenarios are visible in the data. Since it is very hard to differentiate these two cases automatically and a manual identification would not be feasible due to the size of the existing datasets, we use the notion of support to measure the completeness of a key. The support measures the presence of a set of properties in a dataset. Intuitively, we tend to trust more keys that are valid for many instances in the data, i.e., keys with high support.

Basing ourselves on the support definition initially given by Atencia *et al.* in [31], we redefine this measure in order to provide a ranking score for properties with respect to a given dataset.

[1]It denotes if the key allows to discover identical resources with accuracy.

Let D be an RDF dataset described by an ontology O. For a given class $C \in O$, let I_C be the set of instances of type C and P the set of properties having an element of I_C as a subject and let G_C be the subgraph defined by the set of triples of I_C and P, $G_C = \{< i, p, . >: i \in I_C, p \in P\}$.

Definition 2 (Property Support Score) *The support of a property $p \in P$ with respect to the pair (D, C) is defined by:*

$$supportProp(p, D, C) = \left| \bigcup_{i \in I_C} < i, p, . > \right| \frac{1}{|I_C|}.$$

In other words, $supportProp(p, D, C) = N\frac{1}{|I_C|}$ means that N instances of type C in the dataset D have a value for the property p ($supportProp(p, D, C) \in [0, 1]$).

As keys for a given class can be composed of one or several properties, we introduce a ranking score for keys based on the supports of their properties, again with respect to their dataset.

Definition 3 (Key Support Score) *Let $K = \{p_1, ..., p_n\}$ be a key corresponding to the pair (D, C), where $p_j \in P, j \in [1, n]$. We define the support of K with respect to (D, C) as*

$$supportKey(K, D, C) = \left| \bigcup_{i \in I_C} < i, K, . > \right| \frac{1}{|I_C|},$$

where $< i, K, . >$ means that $\forall p_j \in K, \exists < i, p_j, . > \in G_C$.

In other words, $supportKey(K, D, C)$ can be seen as a measure of the **co-occurrence** of $\{p_1, ..., p_n\}$ in G_C.

To illustrate, let us consider a source dataset D_S having 300 instances of type C_S. Respectively, let D_T be a target dataset having 100 instances of type C_T, where C_S and C_T are two mapped (equivalent) classes, potentially sharing instances. Let K_i and K_j be two merged keys, obtained as described in Algorithm 1, with the following supports for (D_S, C_S) and (D_T, C_T), respectively:

$$supportKey(K_i, D_S, C_S) = \frac{160}{300}; \quad supportKey(K_i, D_T, C_T) = \frac{40}{100};$$

$$supportKey(K_j, D_S, C_S) = \frac{110}{300}; \quad supportKey(K_j, D_T, C_T) = \frac{90}{100}.$$

Obviously, the challenge that arises here is how to rank the merged keys in order to ensure a maximum instance representativeness.

We note that key support score expresses the importance of a merged key with respect to each dataset, however, it is still necessary to provide a ranking function allowing to measure the importance of the merged keys for *both* datasets *simultaneously*.

An intuitive strategy to compute the final support of a merged key, given the supports computed locally in each dataset, would be to compute the average score of these supports. Nevertheless, this strategy would fail to capture all the different scenarios that could lead to a support value. For example, a key having supports 1 and 0.4 in datasets 1 and 2, would have the same merged support than a key having supports of 0.7 and 0.7 in datasets 1 and 2 respectively. Thus, we propose a multiplication function between already computed key supports which ensures better results in the context of data linking evaluation. Consequently, we adopt this ranking function as defined below.

Definition 4 (Merged Keys Rank Function) *We define the rank of a merged key K with respect to two datasets D_S and D_T and two classes C_S and C_T as:*

$$mergedKeysRank(K) = supportKey(K, D_S, C_S) \times supportKey(K, D_T, C_T).$$

Applying the ranking to our example, we obtain the following scores:

$$globalRank(K_j) = 0.33; \quad globalRank(K_k) = 0.22; \quad globalRank(K_i) = 0.21;$$

Therefore, in this example, the key K_j is more important than K_i which means that intuitively should lead to better data linking results.

4.3 Experiments

In order to confirm the effectiveness of the proposed approach, we have conducted an experimental evaluation applying two state-of-the-art key discovery tools: SAKey and ROCKER. We have used two different datasets, a real-world dataset coming from the DOREMUS project[2] and a synthetic benchmark provided by the Instance Matching Track of the Ontology Alignment Evaluation Initiative

[2] `http://www.doremus.org`

(OAEI) 2010[3]. The current experiments were applied on links generated semi-automatically using the linking tool SILK. In this evaluation, we highlight a set of issues raised during these experiments. But first, let us define the criteria and the measures used for this evaluation.
Two aspects are taken into account through the keys ranking performed using our approach, first the *correctness* that determines whether the discovered links are correct and second, the *completeness* that determines whether all the correct links are discovered. These criteria are evaluated by the help of three commonly used evaluation metrics:

- ***Precision***: expresses the ratio between the cardinalities of the set of valid matchings and all matching pairs identified by the system.
- ***Recall***: expresses the ratio between the cardinalities of the set of valid matchings and the all matching pairs that belong in the reference alignment.
- ***F-Measure***: is computed by the following formula :

$$\text{F-Measure} = 2 * \frac{Precision * Recall}{Precision + Recall}$$

We note that all considered pairs of datasets are using the same ontology model, hence, the ontology mapping process is not considered in our experiments. We first execute SAKey or ROCKER on each dataset in order to identify the set of keys. However, we emphasize the fact that advanced key exceptions like pseudo-keys or almost keys are not the focus of this work, therefore, only traditional keys are discovered. These keys are then merged and ranked according to their support score. We launch SILK iteratively as many times as the number of the retrieved keys and produce an F-measure at each run by the help of the reference alignment of our benchmark data. We expect to find a monotonic relation between the ranks of keys and the F-measure values produced by SILK by using these keys. Note that the purpose of these experiments is not to evaluate the performance of the linking tools, but to evaluate the quality of the automatically computed ranks of keys. In other words, we assess whether the generated links are increasingly correct in an ascending order of the ranked keys.

4.3.1 Experiments on the DOREMUS Benchmark

The data in our first experiment come from the DOREMUS project and consist of bibliographical records found in the music catalogs of two major French institutions – *La Bibliothèque Nationale de France (BnF)* and *La Philharmonie de Paris*

[3] `http://oaei.ontologymatching.org/2010/`

(PP). These data describe music works and contain properties such as work titles ("Moonlight Sonata"), composer (Beethoven), genre (sonata), opus number, *etc.*. The benchmark datasets were built based on these data with the help of music librarian experts of both institutions, providing at each time sets of works that exist in both of their catalogs, together with a reference alignment. The data were converted from their original MARC format to RDF using the marc2rdf prototype[4] [89]. We consider two benchmark datasets[5], each manifesting a number data heterogeneities:

1) DS1 is a small benchmark dataset, consisting of a source and a target dataset form the BnF and the PP, respectively, each containing 17 music works. These data show recurrent heterogeneity problems such as letters and numbers in the property values, orthographic differences, missing catalog numbers and/or opus numbers, multilingualism in titles, presence of diacritical characters, different value distances, different properties describing the same information, missing properties (lack of description) and missing titles. SAKey produced eight keys in this scenario. The three top-ranked merged keys using our approach are:

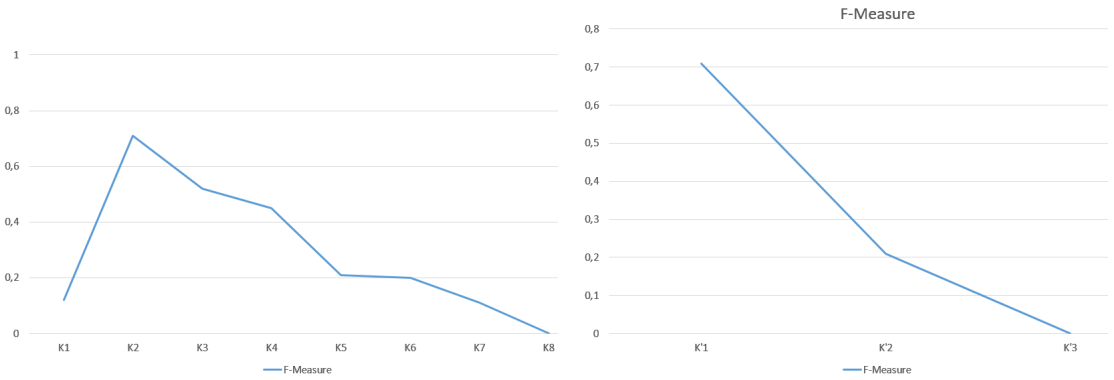
1. K1: $\{P3\_has\_note\}$
2. K2: $\{P102\_has\_title\}$
3. K3: $\{P131\_is\_identified\_by, P3\_has\_note\}$,

where $P3\_has\_note$, $P102\_has\_title$, $P131\_is\_identified\_by$ and $P3\_has\_note$ correspond to a *comment*, *title*, *composer* and *creation date* of a musical work, respectively.

As we can see in Figure 4.2(a), our ranking function ensures a decrease of the F-measure with the decrease of the key-rank, in the prominent exception of the top-ranked key, which obtains a very low value of F-Measure. This is explained by the nature of the property $P3\_has\_note$. This property describes a comment in a free format text written by a cataloguer providing information on the works, creations or authors of such works. The values for this property for the same work are highly heterogeneous (most commonly they are completely different) across the two institutions, which introduces noise and considerably increases the alignment complexity between these resources. Thus, we decided to conduct a second experiment on the same data by removing the property *has_note* in order to confirm our observation. Figure 4.2 (b) reports the results of this experiment and shows a net decrease of the curve. Overall, the experiment showed that our

[4] `https://github.com/DOREMUS-ANR/marc2rdf`

[5] Doremus datasets, together with their reference alignments, are available at `http://lirmm.fr/benellefi/doremus-bench`



(a) (b)

Figure 4.2: Results by using SAKey on DS1: (a) by considering all properties, (b) without the property *has_note*.

ranking approach is efficient and the misplaced key is due to the heterogeneous nature of data.

The same experiment has been conducted using this time the key discovery approach ROCKER. The results are reported in Figure 4.3 showing that the keys were well ranked. Note that, due to the different keys identification definition used by ROCKER, the problematic property *has_note* did not appear in the keys produced by the system.

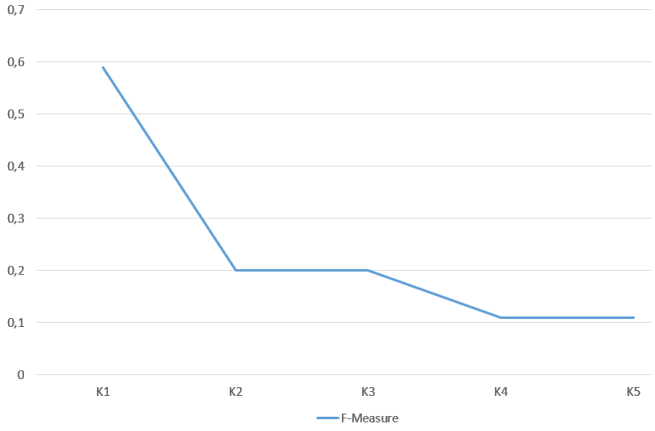
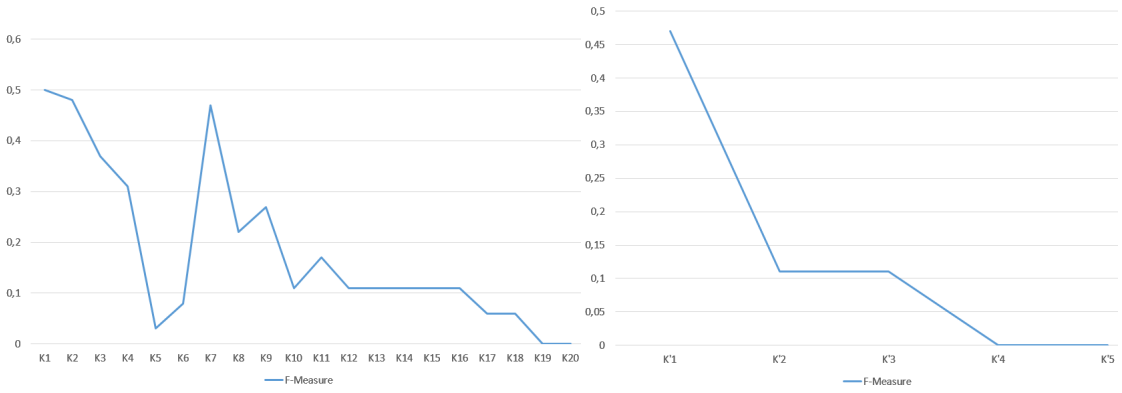


Figure 4.3: Results on DS1 by using ROCKER.

2) DS2 is a benchmark dataset consisting of a source and a target dataset from the BnF and the PP, respectively, each composed of 32 music works. Contrarily to DS2, these datasets consist of blocks that are highly similar in their description



(a) (b)

Figure 4.4: Results by using SAKey on DS2: (a) by considering all properties, (b) without the property *has_note*.

works (i.e., works of the same composer and with same titles).

The results on this dataset by using SAKey are reported in Figure 4.4(a). The three top-ranked merged keys are:

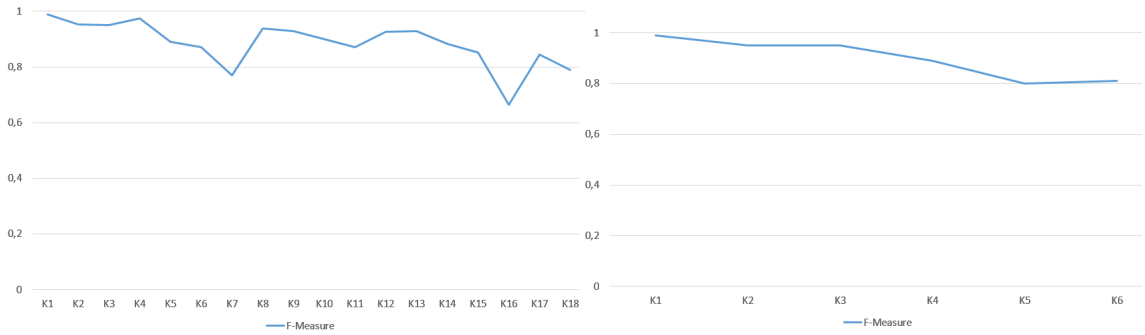
1. K1: $\{P3\_has\_note, P102\_has\_title, P131\_is\_identified\_by\}$
2. K2: $\{P3\_has\_note, P102\_has\_title, U35\_had\_function\_of\_type\}$
3. K3: $\{P3\_has\_note, P131\_is\_identified\_by, P3\_has\_note\}$

As their names suggest the properties $P3\_has\_note$ (in *K1* and the first property in *K2*), $P102\_has\_title$, $P131\_is\_identified\_by$, $U35\_had\_function\_of\_type$ and $P3\_has\_note$ (the third property in *K3*) correspond to a *creation date*, *title*, *composer*, *function of the composer* and *comment* on a musical work, respectively. The results of this experiment are similar to the first one. Not considering the property $P3\_has\_note$ improves considerably (see Figure 4.4 (a) and (b)) the keys ranking. Indeed, as shown in Figure 4.4 (a), the key $K5$ which is composed by the properties $P102\_has\_title$, $U35\_had\_function\_of\_type$ and $P3\_has\_note$ has significantly lowered the *f-measure* value; which is not the case of the keys in Figure 4.4 (b).

4.3.2 Experiments on the OAEI Benchmark Data

In the second series of experiments, we apply our ranking approach on keys identified in datasets proposed in the instance matching track of OAEI 2010. In this

work, we report the obtained results on the dataset *Person*1. The results by using SAKey and ROCKER are shown in Figure 4.5(a) and (b), respectivey, where one can notice that there is an overall decrease in the F-Measure values in the two cases. Note that in Figure 4.5(a), there are some problematic key-ranks, showing increase in F-measure while the ranks descend. We observed that SILK achieves better results comparing string characters than numeric characters. Indeed, this explains why we have had an increasing curve between the keys $K7$ and $K8$, knowing that they are composed of *street* and *house_number* properties (*street* and *surname* properties), respectively. The three top ranked merged keys (in Figure 4.5(a)) on



(a) (b)

Figure 4.5: Results on the dataset Person1: (a) by using SAKey, (b) by using ROCKER.

the dataset Person1 using SAKey are:

1. K1: $\{soc\_sec\_id\}$,
2. K2: $\{given\_name, postcode\}$
3. K3: $\{surname, postcode\}$,

where the properties *soc_sec_id*, *given_name*, *surname* and *postcode* correspond to the *social security number*, *given name*, *surname* and *postal code address* of a person, respectively. In the same manner, we reiterated the experiment using ROCKER which gives better results as shown in Figure 4.5(b).

4.3.3 Top Ranked Keys Complementarity

In this evaluation, we want to examine whether using the k (we have taken $k = 3$) top-ranked keys in combination can improve the linking scores as compared to using only one of the top-ranked keys (e.g., the first one) for linking. As discussed

above, even if a key is discovered as a first-rank key, nothing ensures that the vocabulary used in both datasets to describe that key is homogeneous. To answer to that, combining a set of top ranked keys would lead to better linking results.

	SAKey						ROCKER			
	Dataset 1			Dataset 2			Dataset 1			Dataset 2
	F	P	R	F	P	R	F	P	R	
K1	*0.12*	*0.12*	*0.11*	*0.5*	**0.75**	*0.37*	*0.59*	**0.8**	*0.47*	No merged key has been identified.
K2	0.71	0.9	0.58	0.48	0.7	0.37	0.2	0.66	0.11	
K3	0.52	1	0.35	0.37	0.56	0.28	0.2	0.66	0.11	
K1+K2+K3	**0.54**	**0.44**	**0.7**	**0.51**	*0.63*	**0.43**	**0.62**	*0.75*	**0.52**	

Table 4.1: Results of the combination of the three top-ranked keys on the DOREMUS datasets.

Notice that by doing so, the *recall* value remains the same or increases as compared to the single key approach, while the *precision* may increase (if the proportion of the positive matching pairs becomes larger than the negative matching pairs) as it may as well decrease.
As shown in Table 4.1, the experiments on DOREMUS datasets using the three top ranked keys increased relatively (in bold in the table) the *F-Measure* with respect to the first-rank key (where the improved values are in italics) and significantly the *recall* scores (more positive matching pairs were recovered). Thus, it seems reasonable to conclude that merging the matching results retrieved from the top ranked keys allows to improve significantly the results in terms of *recall*, while this cannot guarantee an improvement in *precision*.

4.4 Conclusion

In this chapter, we have presented a new approach called RANKey for automatically ranking and selecting the most relevant key. As we have shown, the current approaches discover keys only per dataset. To address the problem of finding keys that are valid for a given pair of input datasets, RANKey adopts a strategy of merging the keys generated separately for each of them. Then, it ranks them with respect to their "effectiveness" for the task of discovering `owl:sameAs` links between them. The effectiveness of a merged key is defined as a function of the combination of its respective supports on each of the two input datasets. We also look into the complementarity properties of a small set of top-ranked keys and show that their combined use improves significantly the recall. To demonstrate our concepts, we have conducted a series of experiments on data coming from the OAEI campaign, as well as on real-world data from the field of classical music cataloging.

The proposed approach can be summarized in the following main steps. *(1) Preprocessing:* in this step, given two datasets to be linked, only properties that are shared by both datasets are kept. This ensures that a key can be applied on both the source and the target datasets, and not only on each of them independently. At this point, it is important to state that we consider that the datasets use either common vocabularies or that the explicit mapping between the respective vocabularies is known. *(2) Merge:* the key candidates discovered in each dataset are then merged by computing their cartesian product (recall that a key is a set of properties). *(3) Ranking:* we introduce a ranking criterion on the set of merged keys that is a function of the respective supports of each merged key in each dataset, normalized by the dataset sizes. *(4) Keys combination:* finally, the combined use of several top-ranked merged keys is evaluated, showing an improvement of the recall of a given link discovery tool.

On the one hand, our approach attempts to bridge the gap between configuration-oriented approaches, such as automatic key discovery and automatic link specification. Therefore, it allows to reduce significantly the user effort in the selection of keys used as a parameter of a data linking tool. However, as we will see in the next chapter, RANKey will be used in our proposed approach to disambiguate between highly similar but different resources from the same dataset (see research question 3 in Section 1.5).

Chapter 5

Data Linking Approach across Highly Heterogeneous and Similar Data

Contents

Introduction

The web of data, and particularly the Linked Open Data (LOD) project[1], has been receiving growing popularity over the past years, with hundreds of datasets published on the web following the semantic web principles. However, the vision of the web of data will only become reality when related resources across datasets are linked together—a process that cannot be handled manually, given the amount of data published as RDF knowledge graphs. Data linking is the semantic web research field that has taken the challenge of proposing methods and providing tools for the automatic detection of relations between cross-dataset resources.

As mentioned in Section 1.4, setting the linking tool parameters is a challenging problem that often requires in-depth knowledge of the data. State-of-the-art tools like LIMES [5] or SILK [6] require link specification files where one has to indicate property names, select and tune similarity measures. This process is handled either manually, or by specialized tools [15, 16]. Making a linking tool self dependent in that respect is among the challenges that we set in this work.

Also, a data matcher has to be able to deal with a large variety of data heterogeneities, by taking into account differences in descriptions on value, ontological or logical level (see Section 2.2). While heterogeneities on literals are rather well-handled by similarity measures and data unification techniques, ontological discrepancies (regarding structure and properties) appear to be way more challenging.

Finally, as we will show in our experiments, if the evaluated approaches handle correctly datasets containing blocks of highly similar in their descriptions, but yet distinct resources (for instance, two datasets composed by piano sonatas by Beethoven, Brahms and Schubert), likely to generate false positives.

In this chapter, we aim at reducing the difficulty of manual configuration, especially when it comes to data-related parameters, such as properties to compare, similarity measures to use or threshold setting. We propose a system, called *Legato*, based on indexing techniques that allows to represent each instance as a textual document consisting of its literals, addressing in its mechanism a large variety of data heterogeneities without requiring user input. A data cleaning module allows to decrease noise prior to data linking. Our approach is able to discriminate between highly similar, but different resources, thanks to an efficient post-processinig strategy.

[1] `http://linkeddata.org`

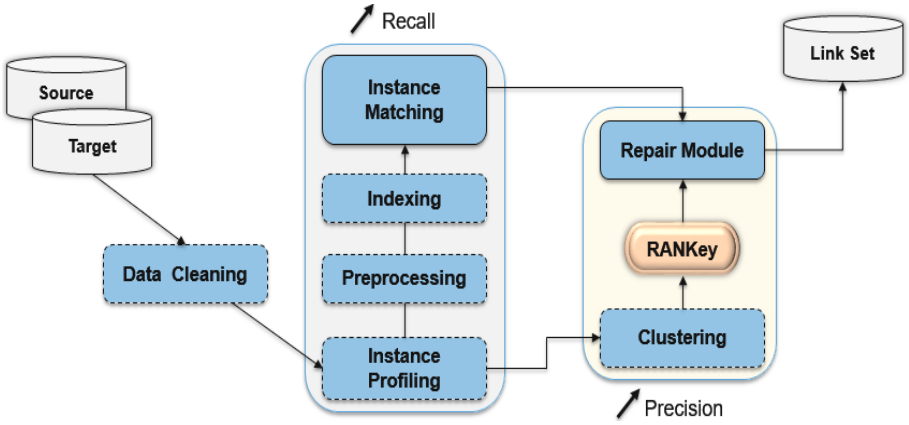


Figure 5.1: Processing pipeline of *Legato* for capturing and repairing identity links

5.1 *Legato* Approach

In this section, we present the complete *Legato* framework, which consists in automatically processing, comparing, repairing and providing a set of identity links (also called a *link set*), as shown in Figure 5.1. The pseudo-code of *Legato* is given in Algorithm 2. It takes one single parameter as input: a pair of datasets to be compared and linked.

5.1.1 Data Cleaning

Certain property values make it difficult, if not impossible, to compare the resources, even if identified as parts of a key. *Legato* considers a property as *problematic* if it makes the comparison difficult for all resources. In this case, we are convinced that it is useless to use this property for matching decision. In fact, there may be a possibility of missing some matches considering this kind of properties. To better clarify this issue, consider the example presented in Table 5.1.

Descriptions `mw1` and `mw1'` are about two equivalent musical works retrieved from *Philharmonie de Paris* and *Bibliothèque Nationale de France* (BNF), respectively (in DOREMUS data). These works are high similar in their description, except the `ecrm:P3_has_note` property values. Indeed, the descriptions of `mw1` and `mw1'` do not match, by considering this property, we would have a very low value of the similarity score.

The problematic properties may concern the properties that have values in a free text format, i.e., comments (as in the above example), as well as known values. By

`mw1`[2]	a efrbroo:F22_SelfContained_Expression mus:U70_has_title “Sonates” mus:U12_has_genre sonate[3] ecrm:P3_has_note “Cette sonate est constituée de cinq formants: Antiphonie, Trope, Constellation, Strophe et Séquence. Seuls les 2e et 3e formants sont publiés. Le Formant 2 (Trope) est composé de quatre sections : Commentaire, Glose , Texte, Parenthèse, qui peuvent être jouées dans différents ordres. Cette oeuvre nécessite un piano à 3 pédales. - Durée d’exécution : 20 minutes environ"
`mw1'`[4]	a efrbroo:F22_SelfContained_Expression mus:U70_has_title “Sonates” mus:U12_has_genre sonate[5] ecrm:P3_has_note “Date de révision : 1963, comprend : Antiphonie; Trope; Constellation (ou Constellation-Miroir); Strophe; Séquence"

Table 5.1: `ecrm:P3_has_note` — An example of a problematic property in DOREMUS data

known values, we mean resources-specific values, that the publisher can not freely describe. Imagine the likely case of different data providers assigning different identifiers to equivalent resources across datasets (e.g., the same record of a musical work is identified differently in the catalogs of two different libraries, as this is the case of DOREMUS data at OAEI 2016[6]). Nonetheless, the property `http://erlangen-crm.org/efrbroo/R46_assigned` will be considered as a key by any key discovery algorithm. If a given linking tool uses this key to compare the instances, it will fail to find a correspondence. One way of going around this problem is to remove the properties of such values, that we call *problematic properties* (line 3 in Algorithm 2).

The way we propose to identify automatically these properties, is to discover mono-property keys that are valid over **both** datasets, i.e., each object for this property has at most one subject in both datasets. Avoiding to compare such properties allows to alleviate *descriptive heterogeneity* (*cf.* Section 2.2) and therefore to improve the linking results.

[6] `http://islab.di.unimi.it/content/im_oaei/2016`

Algorithm 2 *Legato* Data Linking Algorithm

Input: *Datasets*=$\{\mathcal{D}_\mathcal{S}, \mathcal{D}_\mathcal{T}\}$ A pair of datasets to be compared

1: *LinkSet* = $\emptyset$ {*the set of generated links*}
2: **for** ($\mathcal{D} \in$ *Datasets*) **do**
3: $\mathcal{D}$ = clean($\mathcal{D}$);
4: $\mathcal{R}$=$\{r$: $\exists p, o$ with $(r,p,o) \in \mathcal{D}\}$;
5: Set $\mathcal{F}$ = new Set();
6: **for** ($r \in \mathcal{R}$) **do**
7: $\mathcal{F}$.add($\mathcal{CBD}^*(r)$);
8: **end for**
9: **end for**
10: **for** ($f_s \in \mathcal{F}(\mathcal{D}_\mathcal{S})$) **do**
11: **for** ($f_t \in \mathcal{F}(\mathcal{D}_\mathcal{T})$) **do**
12: **if** sim(f_s,f_t)> σ **then**
13: *LinkSet*.add(map(f_s,f_t));
14: **end if**
15: **end for**
16: **end for**
17: $\mathcal{C}_\mathcal{S}$ = Hierarchical-Clustering($\mathcal{D}_\mathcal{S}$);
18: $\mathcal{C}_\mathcal{T}$ = Hierarchical-Clustering($\mathcal{D}_\mathcal{T}$);
19: *SureLinks* = $\emptyset$
20: **for** ($c_s \in \mathcal{C}_\mathcal{S}$) **do**
21: **for** ($c_t \in \mathcal{C}_\mathcal{T}$) **do**
22: **if** sim(c_s, c_t)> ρ **then**
23: *bestKey*=RANKey(c_s, c_t);
24: links = linking(< c_s,c_t >,*bestKey*)
25: *SureLinks*.add(links);
26: **end if**
27: **end for**
28: **end for**
29: *LinkSet*= repair(*LinkSet*, *SureLinks*);
30: **return** *LinkSet*;

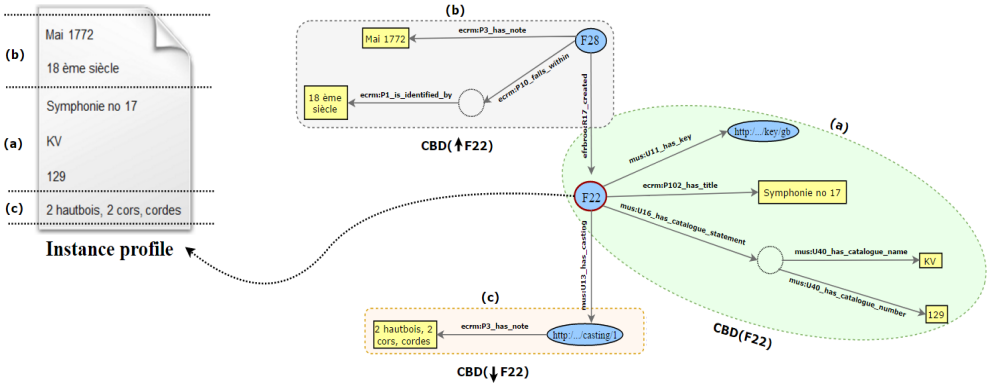


Figure 5.2: Constructing CBD-based instance profile

5.1.2 CBD-based Instance profiling

A core feature of our approach is the representation of resources as text documents (line 7 in Algorithm 2). More particularly, each resource is represented by a set of literals considered as relevant to its description, based on the $\mathcal{CBD}$ subgraphs (Figure 5.2). $\mathcal{CBD}$, for Concise Bounded Description[7], allows to represent a given resource r by a subgraph such that all triples of this subgraph have as a subject r or a blank node connected to r. The $\mathcal{CBD}$ of r is denoted $\mathcal{CBD}(r)$.

Formally, the $\mathcal{CBD}$ of r in an RDF graph $\mathcal{G}$ is defined as follows.

Definition 5 (CBD) $\mathcal{CBD}(r) = \{< r, p, o >: \exists r, p, o \in \mathcal{G}\} \cup \{< r', p', o' >: \exists r', p', o' \in \mathcal{G} \wedge \exists r'' | < r'', p'', r' > \in \mathcal{CBD}(r)\}$,

where r' is a blank node.

We extend the $\mathcal{CBD}$ definition by also considering the description of neighboring nodes of r in its graph defined as follows:

- $\uparrow \mathcal{CBD}(r)$ defines the scope of resource description including $\mathcal{CBD}(r)$ and the concise bounded description of its direct *predecessors* denoted $\mathcal{CBD}(\uparrow r)$.
- $\downarrow \mathcal{CBD}(r)$ defines the scope of resource description including $\mathcal{CBD}(r)$ and the concise bounded description of its direct *successors* denoted $\mathcal{CBD}(\downarrow r)$.
- $\updownarrow \mathcal{CBD}(r)$ defines the scope of resource description including $\mathcal{CBD}(r)$, $\mathcal{CBD}(\uparrow r)$ and $\mathcal{CBD}(\downarrow r)$.
- $\mathcal{CBD}^*(r)$ defines the scope of resource description including one of the $\mathcal{CBD}s$ cited above, i.e., $\mathcal{CBD}(r)$, $\uparrow \mathcal{CBD}(r)$, $\downarrow \mathcal{CBD}(r)$ or $\updownarrow \mathcal{CBD}(r)$.

[7] `https://www.w3.org/Submission/CBD/`

We refer to an instance representation obtained on that basis as an *instance profile* (see Section 2.1.2), defined as follows.

Definition 6 (CBD-based Instance Profile) *Let $\mathcal{G}$ be an RDF graph, let r be a resource of $\mathcal{G}$ and let $\mathcal{F}(\mathcal{G})$ be the set of literals found in $\mathcal{G}$. We define the instance profile of r as the set*

$$f_r = \{l_r : l_r \in \mathcal{F}(\mathcal{G}) \wedge l_r \in \mathcal{CBD}^*(r)\}.$$

Figure 5.2 provides an illustration. We integrate in our framework a parameter that allows the user to choose the profile based on the depth at which the literals are collected – a choice that has to be motivated by the specificities of the data. Selecting the most relevant profile depends on the way the resources are modeled in the graph. Without any user intervention, the default setting represents the instances only by literals found in their $\mathcal{CBD}s$. Thus, avoiding property-based comparison addresses the *vocabulary*, *structural* and *property depth* heterogeneities.

5.1.3 (Pre-)Matching

Once all resources in both datasets are profiled, it is necessary to process the resulting documents in order to prepare data for the matching task. Data preprocessing includes tokenization and stop-words removal by applying NLP (Natural Language Processing) filters. The set of *instance profiles* in both datasets are indexed in a standard manner by using all remaining terms. We apply a vector space model where the terms are weighted by their TF-IDF (Term Frequency-Inverse Document Frequency) weights.

To compute the similarity between the resources (line 13 in Algorithm 2), we compute the correlation between their vectors. To this end, several similarity measures can be used. In our system, we applied the well-known cosine similarity based on the cosine of the angle between two vectors. According to the arity of the matching to perform, we propose two strategies:

— **1:1 expected mappings**. If the matches are expected to be of type **1:1**, we aim at detecting, for each instance from the source dataset, the one from the target dataset that has the *highest* similarity score greater than the threshold σ.

— **1:n expected mappings**. If the matches are expected to be of type **1:n**, we aim at detecting, for each instance, the set of instances that are most similar to it. We apply a blocking method by selecting a subset of vector pairs with high similarity scores by ignoring the remaining pairs, by using σ as threshold to filter

out the matching candidates.
Note that in the (pre-)matching step we have decided to keep the threshold σ low in order to capture as many links as possible, achieving high recall. More precisely, for any input datasets, σ is set to 0.2 regardless to the nature of data to capture as many links as possible. Thereby, it is necessary to improve the quality of the generated links by applying a post-processing task to repair erroneous links which may have been generated at this step, aiming to boost precision.

5.1.4 Link Repairing

For each dataset, the algorithm proceeds to cluster highly similar instances (lines 17 and 18 in Algorithm 2), i.e., sharing the same values for most properties, relying on the generic *Hierarchical Clustering Algorithm* [90]. Then, a cluster matching procedure across the two datasets allows to isolate pairs of similar clusters, where the first belongs to the source dataset and the second – to the target dataset. For that purpose, we use a distance metric on the cluster centroids. Each pair of similar clusters is then analyzed separately and their respective instances are compared on a property basis (lines 20, 21 and 22 in Algorithm 2). The effectiveness of this process depends on the quality of the compared properties. Therefore, the RANKey algorithm [23] is then applied to select the best key (i.e., set of discriminative properties) allowing to disambiguate highly similar instances (line 23 in Algorithm 2). This ensures a maximization of the rate of correct alignments (line 25 in Algorithm 2), or what we call *sure links*. Finally, for each link $\ell=(r_s, r_t)$ produced in the earlier step, the repair module begins by searching over the set of *sure links* for a link with a source resource r_s and with a target resource $r'_t \neq r_t$. The target resource r_t in ℓ is then replaced with r'_t.

5.2 Algorithm Analysis

In this section, we will analyze the efficiency of our data linking algorithm based on three criteria [91]: (i) *Termination*: if it terminates its execution at all; (ii) *Correctness*: if it produces the expected results [92]; and (iii) *Computational complexity*: the number of operations (or time) needed to terminate its execution [93].

Termination. Our algorithm takes in input two RDF datasets and provides a set of *identity* links. The termination of the algorithm depends on the number of comparisons performed between the resources. The number of comparisons

depends on the size of the input datasets. We will study in particular the cases leading to compare all the resources. Since the set of resources in each dataset is finite, the *LinkSet* returns necessarily a finite number of `owl:sameAs` links. Therefore, the variables `i` of *for* loops used in the algorithm are between 1 and the cardinal of $\mathcal{D}_\mathcal{S}$ and $\mathcal{D}_\mathcal{T}$. Thus, these *for* loops terminate in the sense of a finite number of iterations.

In the case where the input datasets are not finite, the algorithm certainly loops to the infinite, but it would be just enough to limit the reached resources in the graphs, i.e., if the numbers n and m ($n, m \in \mathbb{N}$) of source and target resources (respectively) are compared, then the algorithm stops its execution. Therefore, we consider that the input datasets are finite.

Correctness. Regarding Legato, we have to show that it produces the expected results in terms of generated links. Indeed, we must ensure that our algorithm generates an identity link for each pair of resources whose similarity value is higher than a given threshold. Previously, we have demonstrated that the algorithm treats all possible combinations of resources in both datasets in a finite time. Then, we have to check its correctness at two steps of the linking process:

1. (Pre-)Matching: The result is correct for $n * m$ (n and m are the number of source and target resources, respectively) iterations of the algorithm. For each pair of data items, if the similarity value is higher than a given threshold σ, then the algorithm generates a new link between them (see line 12 in Algorithm 2).
2. Link repairing: The result is correct for all source and target cluster pairs. For each pair of clusters, if the similarity value is higher than a given threshold ρ, then the algorithm compares all the resources which are contained therein (see line 22 in Algorithm 2).

This criterion ensures that our algorithm always gives the expected link for any input datasets.

Computational Complexity. The classic definition of computational complexity is the number of operations performed by the algorithm to return the expected result. The number of operations is expressed according to the size of the input data. Regarding Legato, the operations are the conditions, the *for* loops and the generation of the identity links. The computational complexity of our algorithm depends on: (pre-)matching, clustering and the similarity computation between the formed clusters.

For (pre-)matching, it is important to notice that at most only one link is generated by iteration. Consequently, the total number of produced links can vary between 1 and $n*m$ (n and m are the number of source and target resources, respectively). Therefore, the complexity of this step is of order $O(n*m)$. In order to simplify, if we consider the worst case, we select the highest cardinal denoted n. In this case, we suppose that n is the cardinal of the source dataset and also the one of the target dataset. Therefore, the complexity would be of order $O(n^2)$.

Following the work of [94], the hierarchical algorithm (see the lines 17-18 in Algorithm 2) has a time complexity of $O(n^2 log(n))$, where n is the number of data points. It is clear that the computational complexity of this algorithm severely slows the execution of our data matcher for large datasets. If we want to deal with this theoretical complexity, our algorithm could be improved by using other more efficient clustering algorithms [95, 96] of complexity $O(n^2)$.

Moreover, the complexity of comparing the pairs of the formed clusters (see the lines 20-22 in Algorithm 2) depends on the number of the formed clusters in both datasets. Consequently, the total number of comparisons can vary between 1 and $k*l$ (k and l are the number of source and target clusters, respectively). Thereby, the complexity is of order $O(k^2)$. For the sake of simplicity, we suppose that k is the number of the clusters in each dataset.

Finally, considering these three steps, the global computational complexity of our algorithm is of order $O(n^2 + k^2 + n^2 log(n))$.

5.3 Experiments

Legato was implemented in Java 8 and the experiments were conducted on a machine running under Windows 10 over an Intel Core i5-5300U, with 2.30 GHz CPU and 16 GBytes RAM. The system is available an open source (see the Introduction). We provide an open source implementation of our prototype *Legato* in a GitHub project[8].

5.3.1 Experimental Setting

We evaluate *Legato* on datasets from the Ontology Alignment Evaluation Initiative (OAEI) instance matching tracks, as well as on real-world music-related data from

[8] `https://github.com/DOREMUS-ANR/legato`

the DOREMUS[9] project.

Datasets. Our experiments were performed on real-world datasets in the field of classical music coming from the DOREMUS project and on several well-known benchmark datasets from Instance Matching tracks of the Ontology Alignment Evaluation Initiative (IM@OAEI2015[10] and IM@OAEI2016[11]).

• **DOREMUS datasets.** The DOREMUS project brings together a number of major French cultural institutions, among which *Radio France*, *Bibliotheque Nationale de France* (BNF) and *Philharmonie de Paris* (PP). Among the aims of the project is to represent and publish the catalogs of these institutions, containing rich data about music works and events, as RDF graphs, following a specifically designed for this purpose model [89]. We have asked the librarian experts of DOREMUS to construct reference alignment datasets containing matching music works of their respective institutions with different degrees of heterogeneities. This resulted in two dataset pairs **PP-BNF**, denoted by D_{HT} (for heterogeneities) and D_{FP} (for false positives). We make these data available under the following link: `https://github.com/DOREMUS-ANR/doremus-playground`. Note that an earlier version of this benchmark was part of the IM@OAEI 2016 campaign, namely 9-HT, 4-HT and FP-trap datasets. D_{HT} contains 476 musical works, which are highly heterogeneous (see Section 2.2 for *data heterogeneity* types), while D_{FP} contains 66 musical works, which have very similar descriptions, challenging the ability of linking systems to correctly disambiguate them. All data follow the same model and therefore share significant number of vocabulary terms. Because of their very high heterogeneity, richness of description and because of our unsuccessful earlier attempts to link them by the help of off-the-shelf tools, we have used these data as a main motivating example for developing and evaluating *Legato*.

• **OAEI datasets.** We additionally evaluated LEGATO on five artificial benchmark dataset pairs. The first three datasets come from the OAEI 2015 campaign. They have been generated through the Semantic Publishing Instance Matching Benchmark (SPIMBENCH) [97] by transforming the source instances based on their values and semantics (val-sem task), on their values and structures (val-struct task) and on their values, structures and semantics (val-struct-sem task). Note that each of the three datasets is about 10000 instances (referred as SANDBOX datasets). In the same manner, the two other datasets (SPIMBENCH small and SPIMBENCH large) coming from the OAEI 2016 campaign have been produced by employing value-based, structure-based and semantics-aware transformations. We then compared the performance of LEGATO with the participating systems.

[9] `http://www.doremus.org`
[10] `http://oaei.ontologymatching.org/2015/`
[11] `http://oaei.ontologymatching.org/2016/`

In 2015, the two systems STRIM [98] and LogMap [99] participated in the three tasks. In 2016, the systems LogMapIm, AML [86] and RiMOM [18] participated in both datasets.

To evaluate data linking systems, three traditional metrics are used: Precision ($\mathcal{P}$), Recall ($\mathcal{R}$) and F-Measure ($\mathcal{F}$). $\mathcal{P}$ and $\mathcal{R}$ evaluate the *correctness* and the *completeness* of the generated links, respectively, while $\mathcal{F}$ is their harmonic mean.

Scenarii. Before comparing to other linking systems, we evaluated *Legato* with respect to three of its core features: (1) the impact of problematic properties on the quality of the generated links, (2) the choice of instance representation and (3) the use of keys to efficiently assess and repair such links. In the first scenario, instances are first compared considering all the properties and then after removing problematic ones. An improved performance is expected after property filtering. As a second scenario, the identity links are identified by comparing the different representations of the instances, i.e., their $\mathcal{CBD}$, $\uparrow \mathcal{CBD}$, $\downarrow \mathcal{CBD}$ or $\updownarrow \mathcal{CBD}$. The aim is to analyze the behavior of *Legato* according to these different representations. Finally, to evaluate the repairing capacity of the post-processing step, we applied *Legato* with and without performing this step. The repair module should at least validate the generated links and at best repair the erroneous ones.

We have conducted a series of experiments by varying the threshold value σ observing its impact on the $\mathcal{F}$, $\mathcal{P}$ and $\mathcal{R}$ measures. We observed that the best results of *Legato* on all data were achieved with $\sigma = 0.2$ – a low threshold, which guarantees a fair trade-off between the matching module (ensuring high recall) and the repairing module (improving precision).

5.3.2 Effectiveness of Data Cleaning

To assess property filtering effectiveness, we perform experiments on DOREMUS datasets articulated in two phases: without (histograms) and with (curves) removing problematic properties.

The experiments (Figure 5.3) show that applying property filtering allows to improve the linking quality for all datasets except for D_{HT} and 9-HT. Examining the results on these two datasets reveals that removing all the other properties separately does not improve the results either, which indicates that all the properties are important for the comparison.

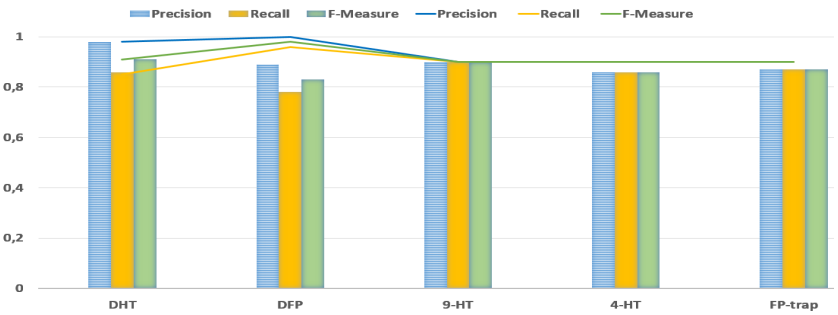


Figure 5.3: Property filtering evaluation on DOREMUS datasets.

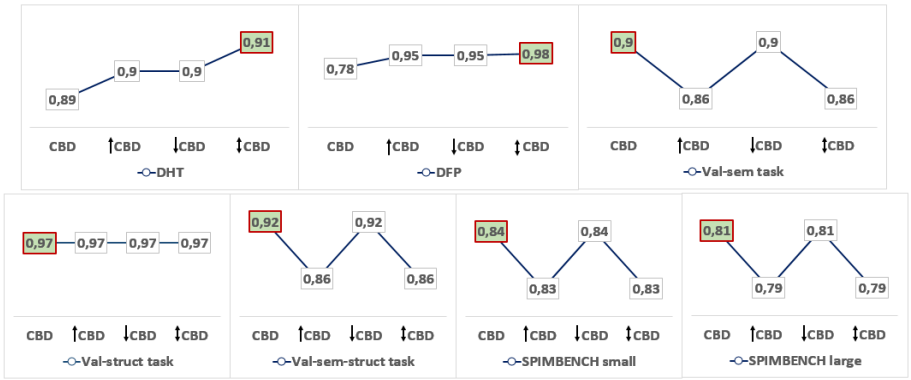


Figure 5.4: CBD-based instance profiling evaluation on reference datasets.

5.3.3 Effectiveness of Instance Profiling

The choice of instance profiles to pick up only the relevant data is shown in this second series of experiments. As shown in Figure 5.4, the effectiveness of instance profiling depends on how the data are modeled. Observing the F-Measure values, it is easy to deduce that: (1) For instances of both D_{HT} and D_{FP} data, the relevant information is located in the $\updownarrow \mathcal{CBD}$ of the resources, with 91% and 98% of F-Measure respectively; (2) For all the other datasets, the relevant information is located only in their $\mathcal{CBD}$. For those datasets, we can also deduce that taking into account the description of predecessors does not impact the matching decision which assumes that creative works have no description in their predecessors. Thereby, we set $\updownarrow \mathcal{CBD}$ and $\mathcal{CBD}$ as profile parameters for these two datasets, respectively.

		#safe_links	#repaired_links	#added_links
D_{HT}	$\mathcal{CBD}$	$\simeq 10\%$	$\simeq 2\%$	0%
	$\downarrow \mathcal{CBD}$	$\simeq 14\%$	$\simeq 2\%$	0%
	$\uparrow \mathcal{CBD}$	10.5%	$\simeq 2\%$	0%
	$\updownarrow \mathcal{CBD}$	$\simeq 5\%$	$\simeq 1\%$	0%
D_{FP}	$\mathcal{CBD}$	$\simeq \mathbf{55}\%$	$\simeq \mathbf{18}\%$	$\simeq 3\%$
	$\downarrow \mathcal{CBD}$	$\simeq 21\%$	$\simeq 3\%$	0%
	$\uparrow \mathcal{CBD}$	$\simeq 15\%$	0%	0%
	$\updownarrow \mathcal{CBD}$	$\simeq 15\%$	0%	0%

Table 5.2: Link repairing evaluation on experimental datasets

5.3.4 Effectiveness of Link Repairing

Table 5.2 presents the effectiveness of post-processing on DOREMUS data, where **#safe_links** represents the proportion of `owl:sameAs` links generated in both matching and repairing steps. The *same* links produced in those steps are considered as valid, **#repaired_links** represents the proportion of erroneous links repaired by the *repair* module, and **#added_links** represents the proportion of links generated by the *repair* module and added in the final link set. We can observe that performing the link repairing step is significant in the presence of highly similar yet distinct instances.

5.3.5 General Results and Discussion

In this experiment, we test the performance of *Legato* in its complete version, i.e., performing both *property filtering* and *link repairing* steps. We compare *Legato* to the participant tools to the instance matching track for OAEI and with AML [86] and SILK [6] for DOREMUS data. Note that the comparison to SILK was made by using the best keys in its link specification identified by the RANKey algorithm [23]. The results of the evaluation are presented in 5.3.

We can observe that *Legato* significantly outperforms the state-of-the-art systems in 4 out of the 10 datasets. These results are of primary significance as they show that *Legato* is well suited for dealing with real-world data containing difficult to disambiguate instances (D_{FP} and FP-trap). *Legato* succeeds where other tools fail thanks to its clustering and key-based *post-processing* phase (Subsection 5.1.4). Also, our system outperforms other tools on highly heterogeneous data (D_{HT} and 4-HT). In fact, *Legato* performs well when data heterogeneity is related to *descriptive* differences and all remaining heterogeneity types of the *ontological*

	System	$\mathcal{P}$	$\mathcal{R}$	$\mathcal{F}$	size
D_{HT}	LEGATO	0,98	0,86	**0,91**	$\simeq$ 476
	AML	0,63	0,77	0,69	
	SILK	0,98	0,29	0,45	
D_{FP}	LEGATO	1	0,96	**0,98**	$\simeq$ 66
	AML	0,68	0,72	0,7	
	SILK	0,7	0,21	0,32	
$9-HT$	LEGATO	0,9	0,9	0,9	$\simeq$ 60
	AML	0,96	0,87	**0,91**	
	RIMOM	0,81	0,81	0,81	
$4-HT$	LEGATO	0,9	0,9	**0,9**	$\simeq$ 400
	AML	0,93	0,77	0,84	
	RIMOM	0,74	0,74	0,74	
$FP-trap$	LEGATO	0,9	0,9	**0,9**	$\simeq$ 80
	AML	0,92	0,85	0,88	
	RIMOM	0,7	0,7	0,7	
Val-Sem Task	LEGATO	0,91	0,89	0,9	$\simeq$ 10000
	STRIM	0.91	0.99	**0.95**	
	LogMap	0.99	0.86	0.92	
Val-Struct Task	LEGATO	0,98	0,96	0,97	$\simeq$ 10000
	STRIM	0.99	0.99	**0.99**	
	LogMap	0.99	0.82	0.9	
Val-Sem-Struct Task	LEGATO	0,94	0,89	0,92	$\simeq$ 10000
	STRIM	0.92	0.99	**0.96**	
	LogMap	0.99	0.79	0.88	
SPIMBENCH small	LEGATO	0,98	0,74	0,84	$\simeq$ 380
	LogMapIm	0.95	0.76	0.85	
	AML	0.9	0.74	0.82	
	RiMOM	0.98	1.0	**0.99**	
SPIMBENCH large	LEGATO	0,96	0,71	0,81	$\simeq$ 1800
	LogMapIm	0.98	0.69	0.81	
	AML	0.9	0.74	0.81	
	RiMOM	0.99	1.0	**0.99**	

Table 5.3: Results on the benchmark datasets for Legato, compared to other linking tools.

dimension (Section 2.2) thanks to its *property filtering* and *indexing* techniques, respectively. As expected, our system is less efficient on the synthetic data transformed with SPIMBENCH, where it ranks second, for most of the experiments. This can be explained by the fact that the heterogeneities of *value dimension* are not considered by *Legato*. Nevertheless, we consider the results of *Legato* satisfactory on these data, given that it provides comparable results as the systems, to which it was confronted, but requires significantly less user-tuning effort.

To sum up, *Legato* ranks among the best linking tools, outperforming them or achieving comparable results on the used datasets, by demanding less user intervention in the linking process. On the one hand, *Legato* is particularly suited to deal with highly heterogeneous real-world data, and on the other hand it is able to cope with difficult particular cases where blocks of highly similar but yet distinct instances are described in both datasets.

System	Precision	Recall	F-measure
AML	0.849	**1.000**	0.918
I-Match	0.854	0.997	**0.920**
Legato	**0.980**	0.730	0.840
LogMap	0.938	0.763	0.841

Table 5.4: The results for SPIMBENCH sandbox of synthetic task

System	Precision	Recall	F-measure
AML	0.855	**1.000**	**0.922**
I-Match	0.856	0.997	0.921
Legato	**0.970**	0.700	0.810
LogMap	0.893	0.709	0.790

Table 5.5: The results for SPIMBENCH mainbox of synthetic task

System	Precision	Recall	F-measure
AML	0.851	0.479	0.613
I-Match	0.680	0.071	0.129
Legato	0.930	0.920	0.930
LogMap	0.406	0.882	0.556
NjuLink	**0.966**	**0.945**	**0.955**

Table 5.6: The results for HT of DOREMUS task

System	Precision	Recall	F-measure
AML	0.914	0.427	0.582
I-Match	1.000	0.053	0.101
Legato	**1.000**	**0.980**	**0.990**
LogMap	0.119	0.880	0.210
NjuLink	0.959	0.933	0.946

Table 5.7: The results for FPT of DOREMUS task

5.4 Related Work Positioning

Numerous research efforts have been made to find identical instances across RDF graphs. This research has been described and surveyed in Chapter 3. In this section, in a more concise manner, we describe the main commonalities and differences between several of the most well-known linking tools, by putting an emphasis on

the configuration and the post-processing phases, in an attempt to better situate our contribution.

The core idea behind the link specification learning methods [15, 16, 56, 61] is to provide no configuration to data linking. The problem of identifying accurate link specifications has been receiving increased attention due to the laborious human effort required to know which setting of instance relatedness parameters allows to take accurate matching decisions. As stated in Sub-section 3.3, only few approaches address this problem and the majority of them applies the active learning technique [100]. For learning link specification, these approaches employ the algorithm of genetic programming [101]. The main drawback of such algorithm is the large number of iterations that does not guarantee an optimal solution. Furthermore, it is true that it reduces the manual effort of specifying the linkage rules but the user is still required to label a set of candidate links to learn the algorithm. In contrast, *Legato* aims to reduce as much as possible the user intervention. The only required user-configuration in *Legato* is the type of the resources to link since choosing CBD-types remains optional. In fact, as shown through experiments in Sub-section 5.3.3, the default setting using only the $\mathcal{CBD}$ of the resources produces good F-Measure values on all benchmarks. Moreover, regardless the nature of the data, the default setting for the similarity threshold is always set at a low value in order to capture as many links as possible in the (pre-)matching step and thus to increase the recall. In contrast, in the link repairing step, the similarity threshold is always set at a high value in order to increase the accuracy of the generated links. These are design decisions that do not involve the end-user.

5.5 Conclusion

In this work, we proposed, implemented and evaluated *Legato*, an open source framework for capturing and repairing identity links in the context of web data. We have identified and addressed the data heterogeneities that make the matching decision difficult. *Legato* addresses effectively many of these heterogeneities by the help of its data cleaning from the problematic properties and link repairing modules. In addition, our system has the advantage that no user configuration is required in terms of selection of properties to compare, similarity measures to select and threshold values to set. In fact, a unique threshold is set for all data regardless of their degree of heterogeneity.

A core feature of *Legato* is its capacity to avoid the generation of false positive links by disambiguate effectively highly similar instances across datasets. We evaluate *Legato* on real-world music-related data and on synthetic datasets coming from

OAEI 2015, 2016 and 2017. The results showed clearly that our system is in competition with state-of-the-art tools, outperforming them on datasets containing highly heterogeneous or difficult to disambiguate instances.

Regarding our algorithm, there is a need for more challenging problem of data linking. Actually, *Legato* implements an approach handling structurally heterogeneous descriptions. However, the limit of the current version of our system is that it is not dealing with value-based heterogeneity since it searches for exact matches. Also, we have shown that while the (pre-)matching step scales linearly with the number of resources, the computational complexity of the *repair* module (hierarchical clustering) drastically worsens as this number increases. Furthermore, we need to discover matches between resources coming from multiple data sources simultaneously.

Chapter 6

Conclusion and Perspectives

Contents

This chapter concludes the dissertation by first summing up our main contributions compared to the research questions posed in Chapter 1. Then, we present some of the remaining issues not yet solved by our data linking system. Finally, we highlight several possible directions for future work.

6.1 Thesis Summary

This thesis presents a data liking method that captures identity links in the open context of the web. As stated in Chapter 1, we set out the following research problem:

what requirements must a data linking system fulfill to reduce as much as possible the user's intervention and how can it overcome the differences of identical resources and discover the similarity of distinct ones?

Indeed from this question, we identified three main challenges in Chapter 1. We investigated in depth each research question, we implemented solutions and we evaluated them through experiments in three chapters; each solution corresponds to a specific step of the global data linking process.

As a first contribution, we showed the first research question of *what are the different heterogeneity types and how are they organized* in Chapter 2. We extended the list of data heterogeneities in [38] and structured them in three main dimensions: *value*, *ontological* and *logical*. For these dimensions, we identified respectively 4, 4 and 2 heterogeneity types. We defined, analyzed and suggested some possible solutions for each of the considered heterogeneities. However, it is important to notice that best matching performance of *Legato* were achieved tackling only some of these heterogeneities. In particular, the current version of our data linking tool manage only the heterogeneities of *ontological* dimension. This work does not address the problem of *class* dimension. Indeed, we assume that ontological mappings already exist or the resources to be compared are of the same type. Even so, methods are still needed to automatically connect equivalent resources even if they are described with the same ontology. From our investigations, it appears that few approaches in the literature tackle the heterogeneities of the *ontological* dimension. An overview of the more relevant approaches dealing with these heterogeneities is presented in Sub-section 2.2. Nevertheless as discussed in the same chapter, the proposed solutions do not seem to be efficient. We related the problem of *vocabulary*, *structural* and *property depth* heterogeneities to the problem of *profiling* the resources. We established a representation making the resources comparable at the same level of granularity.

Considering the third research question of *how to make the linking process as automatic as possible*, the aim is to deal with the complexity of systems requiring a considerable user configuration effort. As explained in Chapter 5, we proposed a semi-supervised approach to capture match candidates by identifying the resources that share similar profiles. In fact, the only required user-configuration is the type of the resources to link. Choosing CBD-types and mapping arity remains optional. Regarding the similarity threshold, the parameter does not depend on the data. In the first step, the similarity threshold is set to 0.2 improving recall. The main objective of this process is to obtain the possible equivalence links even those that are generated between the least similar resources. It is clear that the threshold parameter affects the performance of our system since it may increase the number of false positive matching. The next step is to analyze the presence of such mappings. Especially, a source resource r_1 may be linked to one or n ($n >= 1$) target resources (r'_1, r'_2, ..., r'_n) while the expected mapping is between r_1 and only one target resource. At this stage, we decided to always set the threshold at 0.6 distinguishing these resources and increasing precision. These are design decisions that do not involve the end-user.

Due to the highly similar descriptions in the same data source, as stated in the fourth research question in Section 1.5 of Chapter 1, a large number of false positive matching may be generated. In the post-processing step, this issue is fixed by a key-based data linking approach. In Chapter 4, we proposed an unsupervised approach to select the best combination of attributes, called *key*, for distinguishing the resources across two RDF graphs. As explained in Sub-section 5.1.4, we select for each pair of similar clusters the best key to distinguish the similar resources and match only those having the same values for this key.

In Chapter 4, we showed that although the use of keys based on which data linking approaches attempt to align the resources sharing the same values for them, key discovery techniques may decrease their effectiveness for the matching task. Indeed, the keys are extracted from each dataset independently not guaranteeing that the resource pairs satisfying the matching condition are unique in their relations. In other words, a set of properties P identified as a key for the dataset A is not necessarily a key for the dataset B. Thus, comparing the resources from A and B using P may output a lot of matches that are actually non-matches.This problem raises the question of *how to identify the best key valid from both datasets to be compared*. In order to resolve this issue, we proposed an automatic approach which sorts the keys based on their effectiveness. The best key is the one which is able to provide as many correct `owl:sameAs` links as possible. We introduced a new support-based criterion computed for keys that are valid for both datasets to be compared.

Finally, the output of this thesis is the implementation of an instance matcher prototype called *Legato* that overcomes all the issues raised beforehand. The processing pipeline of *Legato* is presented and described in Sub-section 5.1. Firstly, we proposed a naive strategy for cleaning the data before its processing relying on the notion of keys. According to our experiments, some attributes may make the comparison task difficult or may generate false-positive matches despite the fact that they are identified as keys. Such attributes are called *problematic properties* and defined as mono-property keys that are valid for both datasets. Thereby, the main goal of this step consists in dealing with the challenge of *descriptive heterogeneity* and therefore to improve the effectiveness of our system. Secondly, we defined the notion of *instance profiling* as representing each resource by all the literals in its $\mathcal{CBD}^*(r)$. This aims at addressing the challenges of *vocabulary*, *structural* and *property depth* heterogeneities by avoiding the property-based comparison and selecting the most relevant information to compare. Thirdly, the profiles are indexed and *Legato* considers a matching candidate of two resources by computing the correlation between their vectors. Two . Finally, *Legato* disambiguates between highly similar yet different resources by combining clustering and key-selection algorithms. In order to properly evaluate the performance of *Legato*, we decided to experimentally evaluate the suitability of each step with respect to three of its core features as follows:

- We analyzed through experiments the impact of *problematic properties* on the quality of the generated links as described in sub-section 5.3.2.
- In order to evaluate the suitability of instance profiling to pick up the most relevant data, we needed to perform experiments representing the resources by their $\mathcal{CBD}^*$. In other words, four executions were performed where for each of them the resources are described by one of their $\mathcal{CBD}s$, i.e., $\mathcal{CBD}$, $\uparrow \mathcal{CBD}$, $\downarrow \mathcal{CBD}$ or $\updownarrow \mathcal{CBD}$. For more details, please, see sub-section 5.3.3.
- We evaluated the impact of *Legato*'s repair module relied on the use of keys to efficiently distinguish highly similar resources by performing experiments (see sub-section 5.3.4) computing the proportion of repaired and the newly generated links.

The combination of these steps was evaluated on the bibliographic datasets from DOREMUS as well as on the OAEI 2015 and 2016 instance matching test datasets. When applied to these benchmarks, our system obtained comparable results and outperformed the best performing the state-of-the-art systems particularly on highly heterogeneous data as well as on highly similar ones (but yet distinct).

This empirical evaluation allowed us to learn the more suitable features for each dataset. For example, the $\updownarrow \mathcal{CBD}$ is more efficient for comparing the resources in

DOREMUS data using the information in their $\mathcal{CBD}$, that of their predecessors and that of their successors. In particular, this feature depends on the way the resources are modeled in the graph. Notice that *Legato*'s implementation has been made publicly available on a Github platform[1] for reuse.

6.2 Future Research

Several issues have been identified throughout this work which deserve further investigation. In this section, we present some directions for future research that appear to be feasible and effective for data linking.

Dealing with all Data Heterogeneities. In this thesis, we have motivated the investigation of data heterogeneity types as the main challenges to solve when linking data. We have identified the features of each of them as well as some relevant solutions that have to be explored. Particularly, we identified three main dimensions, namely *value*, *ontological* and *logical* dimensions. However, we mainly dealt with the *ontological* dimension. In fact, *Legato* is suitable to avoid the problem of *vocabulary* differences and to solve the problems of *structural*, *property depth* and *descriptive* heterogeneities. To improve the effectiveness of our system, the other dimensions require more attention and especially the heterogeneity at the *lingual* level. In fact, we have shown that when two resources are described with different natural languages, one solution to automatically compare them correctly is to translate one of them into the language that is used to describe the other resource.

Improving the Ranking Criterion. It is important to remember that discovering discriminant properties takes place in the final step of our linking process and any automatic key detection algorithm could be used for that purpose. To our knowledge, the most of state-of-the-art key generation systems often return a very large number of combinations of properties that uniquely identify the resources. In that context, we proposed an approach for automatically evaluating the generated keys and selecting the best one regarding the produced links in terms of F-Measure. Notice that the current key discovery algorithms perform an exact match operation on the resources values. Intuitively, the set of properties for those the instances do not share the same values represent a key. However, this is not necessarily true in all cases. Indeed, variations (errors, misspelling, abbreviations,

[1] `https://github.com/DOREMUS-ANR/legato`

acronyms and punctuations) in object values may lead to discover erroneous keys. Consider the example of values describing the name of the same person but written differently, the key identification tools still consider that the *name* represents a discriminant property. In other cases, there exists linguistic variations (a resource described in different natural languages) of the same value (e.g. école and school) of a property (e.g. describing a workplace). Hence, in a long term perspective, it is crucial to conceive a key ranking approach suitable for data showing *terminological* or *linguistic* heterogeneity. In particular, with respect to the ranking criterion, we plan to add the constraint of *discriminability* regarding the different heterogeneity types at the value dimension. The heterogeneities of the other two dimensions will be dealt with in an earlier step of the overall data linking process.

Automating the Instance Profiling. In the setting of CBD-based instance profiling, four representations $\mathcal{CBD}^*(r)$ can be considered for each resource r. In the current version of *Legato*, the parameter $*$, representing one of the four possible profiles listed in Sub-section 5.1.2, is defined by the user. Selecting the most relevant profile type can be very difficult for the user, since it depends on how the instances are modeled. If the user does not know the structure of the data, the default setting represents the instances only by the literals found in their $\mathcal{CBD}$. However, we have shown that considering different $\mathcal{CBD}^*$s produce different results (see sub-section 5.3.3). Representing the data items only by their $\mathcal{CBD}$ might not be sufficient to keep the most relevant description, while representing them by their $\uparrow \mathcal{CBD}$, their $\downarrow \mathcal{CBD}$ or their $\updownarrow \mathcal{CBD}$ is potentially error prone, i.e., may introduce some noise. Thus, we are interested in proposing ways to automatically select the most appropriate profile type depending on the data.

Setting Automatically the Arity Mapping. In the current version of *Legato*, the default setting is **1:n**. If the expected mapping by the user is **1:1**, *Legato* links each source resource with a target resource having the highest similarity score which is greater than a fixed threshold, i.e., 0.2. Consequently, the user could change the arity parameter of *Legato* to enable **1:1** mapping discovery. However, such user-based configuration can hardly be accomplished considering the wide possible representations of the real world entities. Thereby, it is necessary to provide a mechanism allowing to automatize this configuration. Such a mechanism requires further deep investigation on the data. A possible solution to this problem would be to provide a machine learning based approach relying on some training examples of correspondences.

Linking multiple datasets. In this thesis, we proposed an approach for automatically identifying correspondences between the resources across two datasets. It is important to remember that one of the main goals of DOREMUS project is to match musical works coming from more than two data sources, namely BNF (*Bibliothèque Nationale de France*), *Philharmonie de Paris* and *Radio France*. Hence, in the short term of future work, the transitive relations on the generated links can be exploited to discover matches between resources coming from multi data sources. For example, if an equivalence relation is identified between the resources *mw1* and *mw1'* and between the resources *mw1'* and *mw1"*, then an identity link is generated between the resources *mw1* and *mw1"*. In this case, we suppose that the three resources *mw1*, *mw1'* and *mw1"* are from three different datasets. In doing this, we reduce as much as possible the search space and thus we reduce the time needed for aligning all the resources.

Ontology Matching. The manual definition of ontology mappings is a time-consuming task especially when considering a very large number of properties and classes to be compared. In fact, each concept and each attribute has a label that defines its meaning. However, it sometimes happens that the designer of an ontology does not define them explicitly. For this reason, manually proceeding potentially lead to errors or to missing mappings. This is because the semantic of concepts and attributes may be misinterpreted. Therefore, we plan to extend our current matching solution to proceeding over both the ontologies and the data items. For example, we plan to investigate how to exploit our initial effort for aligning the ontological concepts relying on what we have proposed to align the resources. More precisely, in the future we aim at defining an automatic approach to determine the correspondence between the concepts and then to match the resources of the same type. This is of paramount importance because type alignment is an inevitable step from any data linking process.

Linking Large Scale Datasets. Notice that, considering the exponential growth in the size of the web of data, *Legato* should scale to very big datasets. For example, linking similar products on websites selling consumer electronic goods allows the user to quickly access the desired product and to compare its price which is a non-trivial task if he had to do it manually. Linking these datasets mostly challenges the *repair* module of *Legato*, which may become particularly time consuming according to many experiments we have conducted. In fact, this module performs data linking dealing with high similar descriptions. Particularly, it relies on clustering technique combined with the key ranking algorithm RANKey. *Legato* was experimentally evaluated relying on thousands triples with and without

performing the clustering algorithm. It appeared that our system takes much longer time to execute than when it does not perform the clustering algorithm. A solution consists in including a mechanism for accelerating the repair module while the current version of *Legato* does not ensure that.

Overall, we argue that the main purpose of this work is to show through experimental evaluations how heterogeneous data items can be effectively aligned. We analyzed the data and identified the most recurring problems. Then, we presented solutions to deal with certain challenges at the level of automation, data heterogeneities and similarity between non-identical resources. These solutions have been implemented in a data linking tool called *Legato*. The main advantage of *Legato* is that it remains agnostic to the nature of data, allows to reduce significantly the user effort of link specification and still guarantees a performance that is at least as good as, and in the presence of highly similar descriptions better than the existing off-the-shelf data linking systems.

Bibliography

[1] A. Ferrara, A. Nikolov, and F. Scharffe, "Data linking for the semantic web," *Semantic Web: Ontology and Knowledge Base Enabled Tools, Services, and Applications*, vol. 169, 2013.

[2] D. Allemang and J. Hendler, *Semantic web for the working ontologist: effective modeling in RDFS and OWL*. Elsevier, 2011.

[3] L. Ding, J. Shinavier, T. Finin, and D. L. McGuinness, "owl: sameas and linked data: An empirical study," 2010.

[4] L. Masinter, T. Berners-Lee, and R. T. Fielding, "Uniform resource identifier (uri): Generic syntax," 2005.

[5] A. N. Ngomo and S. Auer, "LIMES - A time-efficient approach for large-scale link discovery on the web of data," in *IJCAI 2011, Proceedings of the 22nd International Joint Conference on Artificial Intelligence, Barcelona, Catalonia, Spain, July 16-22, 2011*, pp. 2312–2317, 2011.

[6] J. Volz, C. Bizer, M. Gaedke, and G. Kobilarov, "Silk-a link discovery framework for the web of data.," *LDOW*, vol. 538, 2009.

[7] A. Nikolov, V. S. Uren, E. Motta, and A. N. D. Roeck, "Integration of semantically annotated data by the knofuss architecture," in *Knowledge Engineering: Practice and Patterns, 16th International Conference, EKAW 2008, Acitrezza, Italy, September 29 - October 2, 2008. Proceedings*, pp. 265–274, 2008.

[8] D. Symeonidou, V. Armant, N. Pernelle, and F. Saïs, "Sakey: Scalable almost key discovery in RDF data," in *The Semantic Web - ISWC 2014 - 13th International Semantic Web Conference, Riva del Garda, Italy, October 19-23, 2014. Proceedings, Part I*, pp. 33–49, 2014.

[9] T. Soru, E. Marx, and A. N. Ngomo, "ROCKER: A refinement operator for key discovery," in *Proceedings of the 24th International Conference on World*

Wide Web, WWW 2015, Florence, Italy, May 18-22, 2015, pp. 1025–1033, 2015.

[10] P. Shvaiko and J. Euzenat, "Ontology matching: State of the art and future challenges," *IEEE Trans. Knowl. Data Eng.*, vol. 25, no. 1, pp. 158–176, 2013.

[11] D. Ngo and Z. Bellahsene, "YAM++ : A multi-strategy based approach for ontology matching task," in *Knowledge Engineering and Knowledge Management - 18th International Conference, EKAW 2012, Galway City, Ireland, October 8-12, 2012. Proceedings*, pp. 421–425, 2012.

[12] D. Faria, C. Pesquita, E. Santos, M. Palmonari, I. F. Cruz, and F. M. Couto, "The agreementmakerlight ontology matching system," in *On the Move to Meaningful Internet Systems: OTM 2013 Conferences - Confederated International Conferences: CoopIS, DOA-Trusted Cloud, and ODBASE 2013, Graz, Austria, September 9-13, 2013. Proceedings*, pp. 527–541, 2013.

[13] E. Jiménez-Ruiz and B. C. Grau, "Logmap: Logic-based and scalable ontology matching," in *The Semantic Web - ISWC 2011 - 10th International Semantic Web Conference, Bonn, Germany, October 23-27, 2011, Proceedings, Part I*, pp. 273–288, 2011.

[14] M. Gulic, B. Vrdoljak, and M. Banek, "Cromatcher: An ontology matching system based on automated weighted aggregation and iterative final alignment," *J. Web Sem.*, vol. 41, pp. 50–71, 2016.

[15] R. Isele and C. Bizer, "Active learning of expressive linkage rules using genetic programming," *J. Web Sem.*, vol. 23, pp. 2–15, 2013.

[16] A. N. Ngomo and K. Lyko, "EAGLE: efficient active learning of link specifications using genetic programming," in *The Semantic Web: Research and Applications - 9th Extended Semantic Web Conference, ESWC 2012, Heraklion, Crete, Greece, May 27-31, 2012. Proceedings*, pp. 149–163, 2012.

[17] S. Araujo, J. Hidders, D. Schwabe, and A. P. De Vries, "Serimi-resource description similarity, rdf instance matching and interlinking," *arXiv preprint arXiv:1107.1104*, 2011.

[18] C. Shao, L. Hu, J. Li, Z. Wang, T. L. Chung, and J. Xia, "Rimom-im: A novel iterative framework for instance matching," *J. Comput. Sci. Technol.*, vol. 31, no. 1, pp. 185–197, 2016.

[19] M. Kejriwal and D. P. Miranker, "Semi-supervised instance matching using boosted classifiers," in *The Semantic Web. Latest Advances and New Do-*

mains - 12th European Semantic Web Conference, ESWC 2015, Portoroz, Slovenia, May 31 - June 4, 2015. Proceedings, pp. 388–402, 2015.

[20] F. Scharffe, Y. Liu, and C. Zhou, "Rdf-ai: an architecture for rdf datasets matching, fusion and interlink," in *Proc. IJCAI 2009 workshop on Identity, reference, and knowledge representation (IR-KR), Pasadena (CA US)*, 2009.

[21] S. Rong, X. Niu, E. W. Xiang, H. Wang, Q. Yang, and Y. Yu, "A machine learning approach for instance matching based on similarity metrics," in *The Semantic Web–ISWC 2012*, pp. 460–475, Springer, 2012.

[22] K. Nguyen, R. Ichise, and B. Le, "Slint: a schema-independent linked data interlinking system," *Ontology Matching*, p. 1, 2012.

[23] M. Achichi, M. Ben Ellefi, D. Symeonidou, and K. Todorov, "Automatic key selection for data linking," in *Knowledge Engineering and Knowledge Management: 20th International Conference, EKAW 2016, Bologna, Italy, November 19-23, 2016, Proceedings 20*, pp. 3–18, Springer, 2016.

[24] D. Beckett and B. McBride, "Rdf/xml syntax specification (revised),"

[25] D. Beckett, T. Berners-Lee, and E. Prudhommeaux, "Turtle-terse rdf triple language," *W3C Team Submission*, vol. 14, no. 7, 2008.

[26] J. Grant and D. Beckett, "Rdf test cases-n-triples," tech. rep., Technical report, W3C Recommendation, 2004.

[27] D. Beckett, "New syntaxes for rdf," *WWW 2004*, 2004.

[28] M. B. Ellefi, Z. Bellahsene, F. Scharffe, and K. Todorov, "Towards semantic dataset profiling," in *Proceedings of the 1st International Workshop on Dataset PROFIling & fEderated Search for Linked Data co-located with the 11th Extended Semantic Web Conference, PROFILES@ESWC 2014, Anissaras, Crete, Greece, May 26, 2014.*, 2014.

[29] B. Fetahu, S. Dietze, B. P. Nunes, M. A. Casanova, D. Taibi, and W. Nejdl, "A scalable approach for efficiently generating structured dataset topic profiles," in *The Semantic Web: Trends and Challenges - 11th International Conference, ESWC 2014, Anissaras, Crete, Greece, May 25-29, 2014. Proceedings*, pp. 519–534, 2014.

[30] C. Böhm, J. Lorey, and F. Naumann, "Creating void descriptions for web-scale data," *J. Web Sem.*, vol. 9, no. 3, pp. 339–345, 2011.

[31] M. Atencia, J. David, and F. Scharffe, "Keys and pseudo-keys detection for web datasets cleansing and interlinking," in *Knowledge Engineering and Knowledge Management*, pp. 144–153, Springer, 2012.

[32] F. Naumann, "Data profiling revisited," *ACM SIGMOD Record*, vol. 42, no. 4, pp. 40–49, 2014.

[33] M. B. Ellefi, Z. Bellahsene, S. Dietze, and K. Todorov, "Dataset recommendation for data linking: An intensional approach," in *The Semantic Web. Latest Advances and New Domains - 13th International Conference, ESWC 2016, Heraklion, Crete, Greece, May 29 - June 2, 2016, Proceedings*, pp. 36–51, 2016.

[34] T. Lesnikova, J. David, and J. Euzenat, "Interlinking english and chinese rdf data sets using machine translation," in *Proc. 3rd ESWC workshop on Knowledge discovery and data mining meets linked open data (Know@ LOD), Hersounisos (GR)*, vol. 2013, 2014.

[35] T. Lesnikova, J. David, and J. Euzenat, "Interlinking english and chinese RDF data using babelnet," in *Proceedings of the 2015 ACM Symposium on Document Engineering, DocEng 2015, Lausanne, Switzerland, September 8-11, 2015*, pp. 39–42, 2015.

[36] C. Bizer, T. Heath, and T. Berners-Lee, "Linked data-the story so far," *Semantic Services, Interoperability and Web Applications*, pp. 205–227, 2009.

[37] D. Symeonidou, N. Pernelle, and F. Saïs, "KD2R: A key discovery method for semantic reference reconciliation," in *On the Move to Meaningful Internet Systems: OTM 2011 Workshops - Confederated International Workshops and Posters: EI2N+NSF ICE, ICSP+INBAST, ISDE, ORM, OTMA, SWWS+MONET+SeDeS, and VADER 2011, Hersonissos, Crete, Greece, October 17-21, 2011. Proceedings*, pp. 392–401, 2011.

[38] A. Ferrara, D. Lorusso, S. Montanelli, and G. Varese, "Towards a benchmark for instance matching," in *Proceedings of the 3rd International Workshop on Ontology Matching (OM-2008) Collocated with the 7th International Semantic Web Conference (ISWC-2008), Karlsruhe, Germany, October 26, 2008*, 2008.

[39] G. A. Miller, "Wordnet: A lexical database for english," *Commun. ACM*, vol. 38, no. 11, pp. 39–41, 1995.

[40] Y. Yamamoto, A. Yamaguchi, H. Bono, and T. Takagi, "Allie: a database and a search service of abbreviations and long forms," *Database*, vol. 2011, 2011.

[41] C. Li, L. Ji, and J. Yan, "Acronym disambiguation using word embedding," in *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA.*, pp. 4178–4179, 2015.

[42] A. Simon, R. Wenz, V. Michel, and A. D. Mascio, "Publishing bibliographic records on the web of data: Opportunities for the bnf (french national library)," in *The Semantic Web: Semantics and Big Data, 10th International Conference, ESWC 2013, Montpellier, France, May 26-30, 2013. Proceedings*, pp. 563–577, 2013.

[43] K. D. Bollacker, R. P. Cook, and P. Tufts, "Freebase: A shared database of structured general human knowledge," in *Proceedings of the Twenty-Second AAAI Conference on Artificial Intelligence, July 22-26, 2007, Vancouver, British Columbia, Canada*, pp. 1962–1963, 2007.

[44] F. Narducci, M. Palmonari, and G. Semeraro, "Cross-lingual link discovery with TR-ESA," *Inf. Sci.*, vol. 394, pp. 68–87, 2017.

[45] A. Gangemi, S. Leonardi, and A. Panconesi, eds., *Proceedings of the 24th International Conference on World Wide Web Companion, WWW 2015, Florence, Italy, May 18-22, 2015 - Companion Volume*, ACM, 2015.

[46] T. Saveta, E. Daskalaki, G. Flouris, I. Fundulaki, M. Herschel, and A. N. Ngomo, "LANCE: piercing to the heart of instance matching tools," in *The Semantic Web - ISWC 2015 - 14th International Semantic Web Conference, Bethlehem, PA, USA, October 11-15, 2015, Proceedings, Part I*, pp. 375–391, 2015.

[47] M. Achichi, Z. Bellahsene, and K. Todorov, "A survey on web data linking," *Revue des Sciences et Technologies de l'Information-Série ISI: Ingénierie des Systèmes d'Information*, 2016.

[48] M. Nentwig, M. Hartung, A.-C. Ngonga Ngomo, and E. Rahm, "A survey of current link discovery frameworks," *Semantic Web*, vol. 8, no. 3, pp. 419–436, 2017.

[49] T. Hastie, R. Tibshirani, J. Friedman, and J. Franklin, "The elements of statistical learning: data mining, inference and prediction," *The Mathematical Intelligencer*, vol. 27, no. 2, pp. 83–85, 2005.

[50] A. Tversky, "Features of similarity," *Psychological Review*, pp. 327–351, 84/4 (July 1977).

[51] M. Konrath, T. Gottron, S. Staab, and A. Scherp, "Schemexefficient construction of a data catalogue by stream-based indexing of linked data," *Web*

Semantics: Science, Services and Agents on the World Wide Web, vol. 16, pp. 52–58, 2012.

[52] G. Salton, A. Wong, and C.-S. Yang, "A vector space model for automatic indexing," *Communications of the ACM*, vol. 18, no. 11, pp. 613–620, 1975.

[53] J. Euzenat, P. Shvaiko, *et al.*, *Ontology matching*, vol. 333. Springer, 2007.

[54] J. A. Bondy and U. S. R. Murty, *Graph theory with applications*, vol. 290. Macmillan London, 1976.

[55] R. Navigli and S. P. Ponzetto, "Babelnet: The automatic construction, evaluation and application of a wide-coverage multilingual semantic network," *Artif. Intell.*, vol. 193, pp. 217–250, 2012.

[56] A. N. Ngomo, J. Lehmann, S. Auer, and K. Höffner, "RAVEN - active learning of link specifications," in *Proceedings of the 6th International Workshop on Ontology Matching, Bonn, Germany, October 24, 2011*, 2011.

[57] M. Kejriwal and D. P. Miranker, "An unsupervised instance matcher for schema-free RDF data," *J. Web Sem.*, vol. 35, pp. 102–123, 2015.

[58] T. Hastie, R. Tibshirani, and J. H. Friedman, *The elements of statistical learning: data mining, inference, and prediction, 2nd Edition*. Springer series in statistics, Springer, 2009.

[59] R. Isele and C. Bizer, "Learning linkage rules using genetic programming," in *Proceedings of the 6th International Conference on Ontology Matching-Volume 814*, pp. 13–24, CEUR-WS. org, 2011.

[60] A. Nikolov, M. d'Aquin, and E. Motta, "Unsupervised learning of link discovery configuration," in *The Semantic Web: Research and Applications - 9th Extended Semantic Web Conference, ESWC 2012, Heraklion, Crete, Greece, May 27-31, 2012. Proceedings*, pp. 119–133, 2012.

[61] A. N. Ngomo, K. Lyko, and V. Christen, "COALA - correlation-aware active learning of link specifications," in *The Semantic Web: Semantics and Big Data, 10th International Conference, ESWC 2013, Montpellier, France, May 26-30, 2013. Proceedings*, pp. 442–456, 2013.

[62] J. Lehmann, R. Isele, M. Jakob, A. Jentzsch, D. Kontokostas, P. N. Mendes, S. Hellmann, M. Morsey, P. van Kleef, S. Auer, and C. Bizer, "Dbpedia - A large-scale, multilingual knowledge base extracted from wikipedia," *Semantic Web*, vol. 6, no. 2, pp. 167–195, 2015.

[63] F. M. Suchanek, G. Kasneci, and G. Weikum, "Yago: a core of semantic knowledge," in *Proceedings of the 16th International Conference on World Wide Web, WWW 2007, Banff, Alberta, Canada, May 8-12, 2007*, pp. 697–706, 2007.

[64] Y. Raimond, C. Sutton, and M. B. Sandler, "Automatic interlinking of music datasets on the semantic web.," *LDOW*, vol. 369, 2008.

[65] M. Rowe, "Interlinking distributed social graphs," in *Proceedings of the WWW2009 Workshop on Linked Data on the Web, LDOW 2009, Madrid, Spain, April 20, 2009.*, 2009.

[66] A. Jaffri, H. Glaser, and I. Millard, "Managing URI synonymity to enable consistent reference on the semantic web," in *Proceedings of the 1st IRSW2008 International Workshop on Identity and Reference on the Semantic Web, Tenerife, Spain, June 2, 2008*, 2008.

[67] D. Fleischhacker, H. Paulheim, V. Bryl, J. Völker, and C. Bizer, "Detecting errors in numerical linked data using cross-checked outlier detection," in *The Semantic Web - ISWC 2014 - 13th International Semantic Web Conference, Riva del Garda, Italy, October 19-23, 2014. Proceedings, Part I*, pp. 357–372, 2014.

[68] D. Wienand and H. Paulheim, "Detecting incorrect numerical data in dbpedia," in *The Semantic Web: Trends and Challenges - 11th International Conference, ESWC 2014, Anissaras, Crete, Greece, May 25-29, 2014. Proceedings*, pp. 504–518, 2014.

[69] J. Volz, C. Bizer, M. Gaedke, and G. Kobilarov, "Discovering and maintaining links on the web of data," in *The Semantic Web - ISWC 2009, 8th International Semantic Web Conference, ISWC 2009, Chantilly, VA, USA, October 25-29, 2009. Proceedings*, pp. 650–665, 2009.

[70] J. C. dos Reis, C. Pruski, and C. Reynaud-Delaître, "State-of-the-art on mapping maintenance and challenges towards a fully automatic approach," *Expert Syst. Appl.*, vol. 42, no. 3, pp. 1465–1478, 2015.

[71] S. Auer, S. Dietzold, J. Lehmann, S. Hellmann, and D. Aumueller, "Triplify: light-weight linked data publication from relational databases," in *Proceedings of the 18th international conference on World wide web*, pp. 621–630, ACM, 2009.

[72] F. Liu and X. Li, "Using metadata to maintain link integrity for linked data," in *2011 IEEE International Conference on Internet of Things (iThings) &*

4th IEEE International Conference on Cyber, Physical and Social Computing (CPSCom), Dalian, China, October 19-22, 2011, pp. 432–437, 2011.

[73] N. Popitsch and B. Haslhofer, "Dsnotify - A solution for event detection and link maintenance in dynamic datasets," *J. Web Sem.*, vol. 9, no. 3, pp. 266–283, 2011.

[74] M. Pourzaferani and M. A. Nematbakhsh, "Repairing broken RDF links in the web of data," *Int. J. Web Eng. Technol.*, vol. 8, no. 4, pp. 395–411, 2013.

[75] H. Paulheim and C. Bizer, "Improving the quality of linked data using statistical distributions," *Int. J. Semantic Web Inf. Syst.*, vol. 10, no. 2, pp. 63–86, 2014.

[76] E. Rajabi, S. S. Alonso, and M. Sicilia, "Analyzing broken links on the web of data: An experiment with dbpedia," *JASIST*, vol. 65, no. 8, pp. 1721–1727, 2014.

[77] C. B. Neto, D. Kontokostas, S. Hellmann, K. Müller, and M. Brümmer, "Assessing quantity and quality of links between link data datasets.," in *LDOW@ WWW*, 2016.

[78] H. Paulheim, "Identifying wrong links between datasets by multi-dimensional outlier detection," in *Proceedings of the Third International Workshop on Debugging Ontologies and Ontology Mappings, WoDOOM 2014, co-located with 11th Extended Semantic Web Conference (ESWC 2014), Anissaras/Hersonissou, Greece, May 26, 2014.*, pp. 27–38, 2014.

[79] F. Angiulli and C. Pizzuti, "Fast outlier detection in high dimensional spaces," in *Principles of Data Mining and Knowledge Discovery, 6th European Conference, PKDD 2002, Helsinki, Finland, August 19-23, 2002, Proceedings*, pp. 15–26, 2002.

[80] Z. He, X. Xu, and S. Deng, "Discovering cluster-based local outliers," *Pattern Recognition Letters*, vol. 24, no. 9-10, pp. 1641–1650, 2003.

[81] M. M. Breunig, H. Kriegel, R. T. Ng, and J. Sander, "LOF: identifying density-based local outliers," in *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data, May 16-18, 2000, Dallas, Texas, USA.*, pp. 93–104, 2000.

[82] H. Kriegel, P. Kröger, E. Schubert, and A. Zimek, "Loop: local outlier probabilities," in *Proceedings of the 18th ACM Conference on Information and Knowledge Management, CIKM 2009, Hong Kong, China, November 2-6, 2009*, pp. 1649–1652, 2009.

[83] M. Amer and M. Goldstein, "Nearest-neighbor and clustering based anomaly detection algorithms for rapidminer," in *Proc. of the 3rd RapidMiner Community Meeting and Conference (RCOMM 2012)*, pp. 1–12, 2012.

[84] M. Amer, M. Goldstein, and S. Abdennadher, "Enhancing one-class support vector machines for unsupervised anomaly detection," in *Proceedings of the ACM SIGKDD Workshop on Outlier Detection and Description*, pp. 8–15, ACM, 2013.

[85] L. Guizol, M. Croitoru, and M. Leclère, "Aggregation semantics for link validity," in *Research and Development in Intelligent Systems XXX, Incorporating Applications and Innovations in Intelligent Systems XXI Proceedings of AI-2013, The Thirty-third SGAI International Conference on Innovative Techniques and Applications of Artificial Intelligence, Cambridge, England, UK, December 10-12, 2013*, pp. 359–372, 2013.

[86] D. Faria, C. Pesquita, B. S. Balasubramani, C. Martins, J. Cardoso, H. Curado, F. M. Couto, and I. F. Cruz, "OAEI 2016 results of AML," in *Proceedings of the 11th International Workshop on Ontology Matching co-located with the 15th International Semantic Web Conference (ISWC 2016), Kobe, Japan, October 18, 2016.*, pp. 138–145, 2016.

[87] M. Atencia, J. David, and J. Euzenat, "Data interlinking through robust linkkey extraction.," in *ECAI*, pp. 15–20, 2014.

[88] P. Shvaiko and J. Euzenat, "Ontology matching: state of the art and future challenges," *IEEE Transactions on knowledge and data engineering*, vol. 25, no. 1, pp. 158–176, 2013.

[89] M. Achichi, R. Bailly, C. Cecconi, M. Destandau, K. Todorov, and R. Troncy, "Doremus: Doing reusable musical data," in *ISWC PD*, 2015.

[90] L. Rokach and O. Maimon, "Clustering methods," in *The Data Mining and Knowledge Discovery Handbook.*, pp. 321–352, 2005.

[91] D. E. Knuth, "The analysis of algorithms," in *Actes du congres international des Mathématiciens*, vol. 3, 1970.

[92] R. S. Boyer and J. S. Moore, *The correctness problem in computer science.* Academic Press, Inc., 1982.

[93] C. H. Papadimitriou, *Computational complexity.* John Wiley and Sons Ltd., 2003.

[94] S. Firdaus and M. A. Uddin, "A survey on clustering algorithms and complexity analysis," *International Journal of Computer Science Issues (IJCSI)*, vol. 12, no. 2, p. 62, 2015.

[95] R. Sibson, "Slink: an optimally efficient algorithm for the single-link cluster method," *The computer journal*, vol. 16, no. 1, pp. 30–34, 1973.

[96] D. Defays, "An efficient algorithm for a complete link method," *The Computer Journal*, vol. 20, no. 4, pp. 364–366, 1977.

[97] T. Saveta, E. Daskalaki, G. Flouris, I. Fundulaki, M. Herschel, and A. N. Ngomo, "Pushing the limits of instance matching systems: A semantics-aware benchmark for linked data," in *Proceedings of the 24th International Conference on World Wide Web Companion, WWW 2015, Florence, Italy, May 18-22, 2015 - Companion Volume*, pp. 105–106, 2015.

[98] A. Khiat, M. Benaissa, and M. A. Belfedhal, "STRIM results for OAEI 2015 instance matching evaluation," in *Proceedings of the 10th International Workshop on Ontology Matching collocated with the 14th International Semantic Web Conference (ISWC 2015), Bethlehem, PA, USA, October 12, 2015.*, pp. 208–215, 2015.

[99] E. Jiménez-Ruiz, B. C. Grau, A. Solimando, and V. V. Cross, "Logmap family results for OAEI 2015," in *Proceedings of the 10th International Workshop on Ontology Matching collocated with the 14th International Semantic Web Conference (ISWC 2015), Bethlehem, PA, USA, October 12, 2015.*, pp. 171–175, 2015.

[100] B. Settles, "Active learning literature survey," *University of Wisconsin, Madison*, vol. 52, no. 55-66, p. 11, 2010.

[101] J. R. Koza, *Genetic programming: on the programming of computers by means of natural selection*, vol. 1. MIT press, 1992.