Polish Academy of Sciences
Systems Research Institute
PhD Studies in Information Technology - Theory and Application

Ghent University
Department of Telecommunications and Information Processing
Database, Document & Content Management

Handling metadata in the scope of coreference detection in data collections

Zastosowanie metadanych przy wykrywaniu podobieństwa w kolekcjach danych

Marcin Szymczak

Dissertation submitted in accordance with the requirements for the joint degree of:
Doctor of Computer Science Engineering by Ghent University and
Doctor of Technical Sciences in the field of Computer Science by Systems Research Institute, Polish Academy of Sciences

Academic year 2014-2015

Polish Academy of Sciences
Systems Research Institute
PhD Studies in Information Technology - Theory and Application

Ghent University
Department of Telecommunications and Information Processing
Database, Document & Content Management

Supervisors:

Prof. Guy De Tré

Ghent University
Department of Telecommunications and Information Processing Database, Document & Content Management
St-Pietersnieuwstraat 41, B-9000 Gent, België Tel.: +32 9 264 34 12
Fax.: +32 9 264 42 95

Sławomir Zadrożny Ph.D., D.Sc.

Polish Academy of Sciences
Systems Research Institute
Newelska 6, 01-447 Warsaw, Poland Tel.: +48 22 38 10 275
Fax.: +48 22 38 10 105

Dissertation submitted in accordance with the requirements for the joint degree of:
Doctor of Computer Science Engineering by Ghent University and
Doctor of Technical Sciences in the field of Computer Science by Systems Research Institute, Polish Academy of Sciences

Academic year 2014-2015

Acknowledgement

First of all, I would like to thank my supervisors Professors Guy De Tré and Sławomir Zadrożny for giving me an opportunity to write this PhD thesis at the Ghent University and Systems Research Institute Polish Academy of Sciences, and for supporting the JOINT DIPLOMA program. I would also like to give a special voice of thanks to Professor Antoon Bronselaer who always found the time to answer my questions and gave many helpful suggestions. Last, but not least, I would like to thank my wife and little Sofia, for irreplaceable support throughout all my years of study. A special voice of thanks also goes to my parents, friends and family for their support.

Poznań, March 2015
Marcin Szymczak

This contribution is supported by the Foundation for Polish Science under International PhD Projects in Intelligent Computing. Project financed from The European Union within the Innovative Economy Operational Programme 2007-2013 and European Regional Development Fund.

Table of Contents

List of Figures

List of Tables

Nederlandse samenvatting

Het uitwisselen van informatie is uitermate belangrijk in de snel en onvoorspelbaar veranderende wereld. De hoeveelheid gegevens groeit heel snel en gegevens zijn dikwijls gedistribueerd over heterogene gegevensbronnen en databanken. Bijgevolg kan dezelfde informatie op verschillende manieren gemodelleerd zijn, wat *coreferentie* genoemd wordt: coreferente gegevens omschrijven dezelfde informatie op een verschillende manier. Om de *interoperabiliteit*, d.i. het vermogen van systemen en organisaties om samen te werken, te verbeteren is het belangrijk dat coreferentie kan worden gedetecteerd. Door de grote gegevensvolumes en de vereiste complexe analyse is een "manuele" detectie gewoonlijk zeer moeilijk en meestal zelfs onmogelijk. Een geautomatiseerde of semi-geautomatiseerde detectiemethode voor coreferentie is dus noodzakelijk.

Algoritmes voor het detecteren van coreferentie kunnen inwerken op twee hoofdniveaus, namelijk het metadata-niveau en het dataniveau. De algoritmes voor beide niveaus kunnen sterk met elkaar interageren. Enerzijds voorzien metadata, bijvoorbeeld onder de vorm van een ontologie of een taxonomie of onder de vorm van een databankschema dat de structuur en eigenschappen van de data vastlegt, in extra informatie over de data, wat het detecteren van coreferentie op het dataniveau kan ondersteunen. Anderzijds kunnen data, meer specifiek de data die beschreven zijn door metadata, worden gebruikt om coreferentie in metadata terug te vinden.

In het eerste deel van deze doctoraatsthesis stellen we twee nieuwe technieken voor het detecteren van coreferentie op metadata-niveau voor. Meer bepaald bestuderen we coreferentiedetectie in XML-databankschema's.

De eerste techniek maakt enkel gebruik van informatie die bekomen wordt uit de schema's zelf. Meer specifiek van de namen (tags) van schemaelementen (attributen) en van de sequenties (hier *paden* genoemd) waarin deze namen voorkomen, want elementen kunnen genest zijn in andere elementen. Het detecteren van coreferentie gebeurt hier op een hiërarchische wijze. De basis van de techniek bestaat erin om te bepalen of delen van paden (*stappen* genoemd) al dan niet coreferent zijn. Daarvoor wordt een "laag-niveau" methode voor het vergelijken van karakterstrings gebruikt. De techniek vergelijkt de elementnamen die voorkomen in de beschouwde stappen op een lexicale wijze en houdt rekening met de relatieve belangrijkheid van de stappen. Deze belangrijkheid hangt af van de positie van de stap in een pad. De verkregen informatie over de coreferentie van de stappen uit twee paden wordt gepast geaggregeerd om informatie te verkrijgen over de coreferentie van deze paden. Daarbij wordt uitgegaan van de volgende heuristiek: twee

paden zijn meer waarschijnlijk coreferent als ze achteraan meer stappen hebben die coreferent zijn. Het gebruik van deze heuristiek maakt de methode heel efficiënt. Informatie over de coreferentie van paden wordt dan op zijn beurt verder geaggregeerd om finaal te kunnen beslissen over de coreferentie van delen van XML-schema's of volledige XML-schema's.

De tweede techniek voor het detecteren van coreferentie in XML-databankschema's maakt enkel gebruik van inhoudelijke data (de XML-data) en bestaat uit een *verticale* en een *horizontale* schemavergelijkingsstap. Eerst worden de voorkomende waarden van attribuutparen lexicaal en statistisch vergeleken tijdens de verticale schemavergelijkingsstap. Daarna wordt er een horizontale schemavergelijking uitgevoerd die gebaseerd is op het zoeken naar coreferente tuples (stapwaarden en padwaarden). Met deze aanpak kunnen ook granulariteit- en bereikproblemen bij attributen worden opgevangen. Een *granulariteitprobleem* doet zich voor wanneer een attribuut uit het ene pad is opgesplitst in meerdere (sub)attributen in het andere pad. Het *bereikprobleem* duidt op situaties waarbij coreferente attributen niet noodzakelijk hetzelfde domein, d.i. verzameling van toegestane attribuutwaarden, hebben. De techniek laat het ook toe om automatisch, semantische verbanden tussen schema-attributen te detecteren door enkel te kijken naar de inhoudelijke data. Gevonden semantische verbanden kunnen op hun beurt nuttig aangewend worden bij het detecteren van coreferentie in de databankschema's. Dit is een duidelijk voordeel ten opzichte van technieken die louter gebruik maken van metadata en daardoor minder efficiënt kunnen zijn.

In het tweede deel van dit werk wordt een nieuwe, geautomatiseerde techniek "Dynamical Order Construction" (DOC) voorgesteld die het toelaat om een kennisbank op te bouwen met semantische informatie over de domeinen van de attributen waarvan de waarden kunnen worden gesorteerd op basis van een (partiële) orderelatie die een notie van veralgemening weergeeft. Een dergelijke kennisbank kan dan bijvoorbeeld worden gebruikt bij het semantisch vergelijken van attributen in coreferentiebepaling en het fusioneren van data (het samenbrengen van coreferente data in één enkele representatie). Om een partiële orderelatie over een domein te bepalen kunnen volgens de DOC-techniek verschillende maten worden gebruikt. Deze maten zijn gebaseerd op de frequentie waarmee (coreferente) domeinwaarden voorkomen in de inhoudelijke data en zijn alle uitgebreid geëvalueerd door middel van statische methodes. Onze nieuwe techniek heeft als voordeel dat er geen a priori taxonomische kennis over de attribuutdomeinen vereist is om de kennisbank op te bouwen. Deze kennis wordt daarentegen dynamisch geconstrueerd op basis van de inhoudelijke data. Wanneer deze data veranderen kan ook de kennisbank worden gecontroleerd en indien nodig worden bijgewerkt.

Het gebruik en de impact van de DOC-techniek op de verschillende aspecten van gegevensintegratie worden als volgt onderzocht. Ten eerste wordt de impact van de techniek op de detectie van coreferente databankrecords (tuples) bestudeerd. De geconstrueerde partiële orderelatie kan worden gebruikt als bijkomende bron van informatie voor de validatie van een gevonden coreferentie over tuples. Meer specifiek kunnen beschouwde attribuutwaarden als coreferent worden gezien indien ze via de geconstrueerde partiële orderelatie aan elkaar zijn gerelateerd.

Ten tweede wordt bestudeerd hoe de automatisch geconstrueerde partiële orderelatie kan worden gebruikt om coreferente representaties van een entiteit te fusioneren tot één enkele representatie in die gevallen waar een semantische vergelijking van de data nodig is. Selectie-gebaseerde fusiefuncties voor coreferente attribuutwaarden worden bestudeerd in het speciaal geval waarbij een gepaste orderelatie over het attribuutdomein niet gekend is of moeilijk te verkrijgen is. Een nieuwe fusiefunctie, die *gebalanceerde selectie* genoemd wordt, wordt voorgesteld. De gebruikte heuristiek bij deze functie is dat de meest specifieke waarde (volgens de orderelatie) van de kandidaatwaarden wordt teruggegeven, op voorwaarde dat deze waarde vergelijkbaar is met alle andere kandidaatwaarden. Verder wordt aandacht besteed aan het probleem van onbeslistheid, d.i. wanneer het gebruik van de heuristiek geen resultaat oplevert omdat voor elke kandidaatwaarde er een andere kandidaatwaarde bestaat waarmee deze niet kan worden vergeleken.

Ten derde wordt het gebruik en de impact van de met de DOC-techniek verkregen "specialisatie-generalisatie" hiërarchie over de domeinwaarden van een attribuut op het automatisch mappen van attribuutwaarden over heterogene datacollecties onderzocht. Er wordt een nieuwe techniek voorgesteld die semantische verbanden creëert tussen gerelateerde waarden in heterogene datacollecties. De techniek gaat uit van een uitbreidbare verzameling van mappingfuncties die gebaseerd zijn op de geconstrueerde tekstuele beschrijvingen van de beschouwde waarden en maakt verder gebruik van "information retrieval"-technieken. De gebruikte verzameling van mappingfuncties bestaat voor het grootste deel uit één-op-meerdere mappingfuncties, wat wil zeggen dat ze één enkele waarde uit de ene datacollectie gaan mappen op een verzameling van waarden uit de andere datacollectie. Elke één-op-meerdere mappingfunctie wordt daarom eerst omgezet naar een verzameling van één-op-één mappingfuncties, die indien mogelijk alle een waarde van hetzelfde abstractieniveau opleveren. Alle één-op-één mappingfuncties worden toegepast waarna het concept van meerderheid en gebalanceerde selectie worden gebruikt om de resulterende mapping te bepalen. Bijkomend wordt ook een techniek voorgesteld voor het vinden van onomastische verbanden tussen attribuutwaarden.

Alle voorgestelde methoden zijn extensief geëvalueerd over grote realistische datacollecties. Door de geselecteerde onderzoeksproblemen te bestuderen beogen we bij te dragen tot de wetenschappelijke ontwikkelingen in de gebieden van dataintegratie en interoperabiliteit.

Streszczenie po polsku

Wymiana informacji jest niezwykle istotna w nieprzewidywalnym i ciągle zmieniającym się świecie. Wciąż powiększa się ilość danych i są to często dane rozproszone w niejednorodnych systemach lub bazach danych. W konsekwencji, ta sama informacja może być przedstawiana na różne sposoby i dane ją na te różne sposoby przedstawiające nazywamy danymi *koreferentnymi* (ang. *coreferent data*). Dla sprawnego przetwarzania informacji pochodzących z różnych źródeł istotne jest automatyczne wykrywanie koreferencji na różnych poziomach reprezentacji danych oraz rozwiązanie problemu *interoperacyjności* - zdolności do efektywnej współpracy systemów i organizacji. Możemy wyróżnić dwa główne poziomy reprezentacji, na których można prowadzić wykrywanie koreferencji: poziom danych i poziom metadanych, przy czym działania te na obydwu poziomach są ze sobą silnie powiązane. Z jednej strony - metadane opisują dane i mogą wspomagać proces wykrywania ich podobieństwa. Z drugiej strony - dane, a konkretnie dane opisane przez metadane (np. schematy baz danych), mogą zostać wykorzystane do konstrukcji dodatkowych metadanych lub wykrywania koreferencji na poziomie metadanych.

W pierwszej części rozprawy zostały zaproponowane dwa nowatorskie rozwiązania problemu dopasowania metadanych i, na tej podstawie, wykrywania koreferencji na ich poziomie w przypadku danych zapisanych z użyciem języka XML. Pierwsza metoda operuje wyłącznie na informacji zawartej w schematach XML, a konkretnie na znacznikach (tagach) elementów schematu i ich sekwencjach, zwanych ścieżkami (ang. *paths*), prowadzących od korzenia dokumentu XML do danego elementu. Koreferencja jest tutaj rozważana w sposób hierarchiczny. Mianowicie, na poziomie podstawowym metoda ta porównuje znaczniki (części ścieżek) jako ciągi znaków, z uwzględnieniem ich zróżnicowanej ważności. Ważność znacznika zależy od jego pozycji na ścieżce: przyjęto, że dla ustalenia koreferentności ścieżek ważniejsze jest podobieństwo znaczników znajdujących się na ich końcach. Następnie, informacje na temat koreferencji par znaczników są odpowiednio agregowane w celu uzyskania informacji na temat koreferentności ścieżek, które z kolei są agregowane w celu otrzymania ostatecznej decyzji na temat koreferencyjności całych schematów. Dzięki intuicyjności i względnej prostocie implementacyjnej tej metody uzyskuje się efektywne i wydajne narzędzie do wykrywania koreferencji. Druga metoda operuje wyłącznie na danych, które są opisane przez porównywane metadane. Jest ona realizowana w dwóch krokach. Najpierw, w ramach pierwszego kroku, porównuje się wartości poszczególnych atrybutów z użyciem zarówno narzędzi statystycznych, jak i poprzez

leksykalne porównywanie pojedynczych wartości tychże atrybutów. Drugi krok bazuje na wykrytych w pierwszym kroku koreferentnych atrybutach i obiektach w bazie danych. Pozwala to przezwyciężyć *problem różnej granulacji informacji* (ang. *attribute granularity problem*), tzn. dekompozycji atrybutu na poszczególne (pod)atrybuty, oraz *problem stopnia pokrycia informacji* (ang. *data coverage problem*), tzn. gdy koreferentne atrybuty nie reprezentują w pełni tej samej informacji. Pozwoliło to na opracowanie sposobu semantycznego dopasowywania atrybutów porównywanych schematów. Jest to oczywista korzyść płynąca z zastosowania danych w procesie dopasowywania schematów, w porównaniu do metod stosujących tylko informacje zawarte w tychże schematach, co może być niewystarczające.

W drugiej części pracy zaproponowany został nowatorski algorytm, nazwany DOC, który automatycznie tworzy bazę wiedzy dotyczącą dziedziny wybranego atrybutu. Baza ta przyjmuje postać relacji porządku częściowego, określonej na dziedzinie danego atrybutu. Znajduje to zastosowanie przy porównywaniu elementów tej dziedziny, pozwalając na uwzględnienie większej lub mniejszej ich *ogólności*. Do określania tej relacji zaproponowano zastosowanie różnych miar odwołujących się do częstości występowania poszczególnych wartości w zbiorze danych. Miary te zostały przebadane statystycznie. W rozprawie przeanalizowano zastosowanie oraz wpływ takiej bazy wiedzy na różne aspekty integracji danych. Po pierwsze został przebadany wpływ metody DOC na skuteczność wykrywania danych koreferentnych. Tak skonstruowana relacja częściowego porządku może służyć jako dodatkowe źródło informacji w procesie walidacji koreferentności obiektów. To znaczy, wartości rozważanych atrybutów uznaje się za koreferentne wtedy, gdy powiązane są one w ramach skonstruowanej relacji. Po drugie, wyżej przedstawiona relacja porządku częściowego znajduje zastosowanie w procesie łączenia (fuzji) wielu koreferentnych reprezentacji danej informacji (ang. *data fusion*). *Selektywne funkcje fuzji danych* (ang. *selection fusion functions*) zostały przeanalizowane dla specjalnego przypadku, dla którego odpowiednia relacja porządku albo nie jest znana albo jej otrzymanie jest trudne. Zaproponowana została nowatorska metoda wyboru wartości atrybutu, która ma zastąpić zbiór wartości w procesie fuzji. Nazwano tę metodę *wyborem zrównoważonym* (ang. *balanced selection*). Metoda ta zwraca najbardziej specyficzną , w sensie rozważanej relacji porządku częściowego, spośród rozważanych wartości, pod warunkiem, że jest ona porównywalna ze wszystkimi pozostałymi wartościami w sensie tej relacji. Rozważa się również przypadek, gdy taka wartość nie istnieje i proponuje się odpowiednie rozszerzenie algorytmu. Po trzecie, przeanalizowano użycie i wpływ relacji częściowego porządku na mapowanie wartości atrybutów o niejednorodnych dziedzinach. Zaproponowano nowatorski algorytm tworzący mapowanie semantyczne pomiędzy wartościami pochodzącymi z niejednorodnych źródeł danych. W tym celu rozważa się zestaw metod mapujących, które są oparte na konstruowanych tekstowych opisach rozważanych wartości i stosują techniki wyszukiwania informacji (ang. *information retrieval*) do dalszego przetwarzania. Z pomocą tych metod uzyskuje się zbiór mapowań, składający się głównie z mapowań jeden-do-wielu. Oznacza to, że pojedynczej wartości z jednego źródła przypisanych zostaje wiele wartości z drugiego źródła. Następnie, zaproponowany

algorytm redukuje ten zbiór mapowań do zbioru mapowań jeden-do-jeden, przy czym o ile to możliwe przypisane sobie wartości z różnych źródeł znajdują się na tym samym poziomie ogólności w sensie rozważanej relacji porządku częściowego. Algorytm ten odwołuje się do częstości mapowań proponowanych przez poszczególne metody mapujące i stosuje wspomnianą metodę wyboru zrównoważonego. Ponadto, zaproponowano również algorytm znajdujący mapowanie specyficzne dla konkretnego obiektu, bazujący na nazwach własnych tychże obiektów (ang. *onomastic information*). Zaletą zastosowania zaproponowanej nowatorskiej metody DOC we wspomnianych wyżej podejściach jest fakt, iż pozwala ona stworzyć automatycznie bazę wiedzy, która znacznie podnosi efektywność realizacji rozważanych zadań, co pokazano w ramach przeprowadzonych eksperymentów obliczeniowych.

Wszystkie opisane metody zostały szczegółowo przetestowane na dużych zbiorach danych o charakterze praktycznym.

English summary

Exchange of information is extremely important in the rapidly and unpredictably changing world. The amount of data is growing very fast and is often distributed over heterogeneous systems or databases. As a consequence, the same piece of information can be represented in different ways, called *coreferent* data. This may be a serious problem in data processing, hampering the interoperability of distributed systems. Due to the volume of data processed and its required multilevel analysis, it is usually very difficult, and often just impossible, to remedy this problem "manually". Thus, it is important to identify coreference in automatic fashion on different levels to secure the *interoperability*, i.e. the ability of systems and organizations to work together. We can distinguish two major levels in coreference detection, namely, the metadata and the data level, which are strongly related to each other. On the one hand, metadata, e.g. a knowledge base (such as an ontology or taxonomy) or a database schema that defines structure and properties, provide additional information about data and can support the coreference detection on the data level. On the other hand, the data, more specifically data which are described by metadata, can be used to construct metadata or detect coreference in metadata.

In the first part of this dissertation we propose two novel schema matching techniques. The first technique is based only on XML schema information, more specifically on names (tags) of schema elements and their sequences, called here *paths*, as elements may be nested in other elements. Coreference is considered here in a hierarchical way. On the basic level, the coreference of steps (parts of paths) is determined by a low-level string comparison method. This method compares element names lexically and considers their relative importance. The importance depends on the position of a step in a path. In general, the following heuristic rule can be considered: two paths are more likely to be coreferent if they have more similar steps at the end. That makes it a very efficient solution. Then information on step coreference is properly aggregated to obtain information on paths coreference which is, in turn, further aggregated to finally decide on the coreference of whole schemas. The second schema matching technique is only based on content data and is a composition of a *vertical* and a *horizontal* schema matching. Firstly, attributes domains are statistically and lexically compared in the vertical matching. Secondly, a horizontal matching is applied which is based on detecting coreferent tuples. This allows to address the *attribute granularity* problem, i.e. decomposition of an attribute into a number of (sub)attributes, and *coverage* problem, i.e. coreferent attributes do not necessarily completely have to represent the same information. This allows to establish semantical schema

matching between corresponding schema attributes. This is a clear benefit of using content data in contrast to schema information-only-based methods which can be ineffective.

In the second part of this work a novel automated method (DOC) is proposed to construct a knowledge base with semantic information on the domain of an attribute. Such a knowledge base then supports the semantic comparison of domain values that can be sorted by means of an order relation reflecting a notion of generality. There are proposed different measures to introduce an order in the relation which are based on the frequency of values. These measures are extensively evaluated using statistical methods. Our novel technique has the advantage that there is no need for a priori taxonomical knowledge on the attribute domains. Instead, this knowledge is dynamically constructed and hence only depends on the content data, which means that it can be automatically reconstructed when these data change. The use and impact of this method on different aspects of data integration are investigated as follows. First, the impact of this method on the detection of coreferent tuples is studied. The constructed order relation can serve as an additional source of information to validate coreferency of tuples. More specifically, values of the attributes under consideration are coreferent if there exists a relation between them in the order relation. Second, the automatically constructed partial order relation is used to merge coreferent representations of an entity which may need semantical comparison to obtain a single representation, in the process called data fusion. Selectional fusion functions have been studied in the special case where a proper order relation is either unknown or difficult to obtain. The novel *balanced* selection is proposed. The heuristics returns the most specific value from among the candidate values provided that the selected value is comparable to all other candidate values. Attention is hereby given to the problem of tie breaking, i.e. if the algorithm does not yield a result because for each value under consideration there exists another value to which it can not be compared. Third, the use and impact of our specialization-generalization hierarchy on the mapping of attribute values across heterogeneous data collections are investigated. The novel algorithm creates semantical mappings between related values in heterogeneous data. To this aim, an extensible set of mappers are proposed that are based on the constructed textual descriptions of considered values and employs information retrieval techniques for further processing. The established mapping set consists mostly of one-to-many mappings which means that single value from the one source is mapped to more than one value in other source. Thus, the selection algorithm reduces the mapping set to a set of one-to-one mappings, if possible, on the same level of abstraction. It is based on the concept of majority (the frequency of mappings) and the balanced selection over the candidate mappings set. In addition, we also propose an algorithm to find a mapping specific for a particular object based on the onomastic information.

All proposed methods are extensively evaluated on large real-life data collections. By studying the selected research problems we aim to contribute to the scientific developments in the areas of data integration and interoperability.

1 Introduction

1.1 Coreferent data

Modern information standards and systems (e.g., relational databases, OO environments, XML) allow to manage information from the real world in a well-structured manner. Efficient and effective processing of this information requires dealing with various problems, notably those which have their roots in quality deficiencies of available information. One of these problems is the existence of *coreferent* data, which means that the same real-world entity is described multiple times within one (or across multiple) database(s). Due to errors, inaccuracies and lack of standardisation, coreferent data are not bound to be equal, which makes finding coreferent data a challenge. As an example, Table 1.1 shows six objects, each representing the same entity, i.e., the Belfry and Cloth Hall of Ghent which are located in the same building. Note that no two representations are equal one to another.

Table 1.1 Example of coreferent objects (the source dataset S).

Key	Name	Lon.	Lat.	Category
1	Belfry	3.724923	51.053651	Belfry
2	Cloth Hall	3.725098	51.053552	Hist. Building
3	Belfrey	3.724911	51.053677	Tourist Attract
4	Cloth Hall	3.724837	51.053555	POI
5	Belfry of Ghent	3.721661	51.054897	POI
6	Belfry & Cloth Hall	3.724911	51.053653	Monument

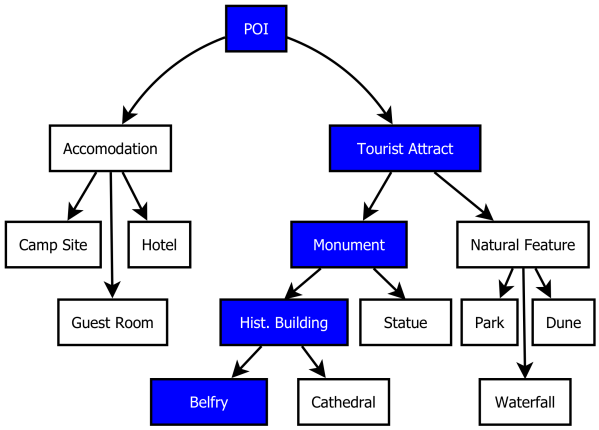
In the past decades, many authors proposed valuable and interesting methods for automated identification of coreferent objects (see [2] for an overview). Many of these methods have a probabilistic nature [3–7] and use statistical properties of variables common to a pair of objects to calculate the probability that these objects are coreferent. Approaches based on the possibility theory also have been proposed [8]. Moreover, rule-based approaches [9] use rules to specify conditions that should be met by two objects to declare them as coreferent.

Interestingly enough, regardless of the underlying mathematical framework, the general idea behind each method is that all modern information systems store objects as a somehow structured entity characterised by a set of *properties* [10] which represent the state of an object and are further divided into object *attributes* and *relationships*. A *relationship* represents an association between the entity represented by that object and entities represented by other objects, e.g., the relationship between an order and a client who placed it. An object *attribute* represents a fact about the entity represented by that object, it has a descriptive name and a value. As an example, in Table 1.1 the objects are described by means of four attributes: “name”, “longitude”, “latitude” and “category”. Starting from decomposition into attributes, techniques for the automated identification of coreferent objects will first compare corresponding attributes separately. Next, the results of these comparisons are combined into one result that can be used to decide whether the objects are coreferent or not. In [11] the comparison of attributes is studied and a distinction is made between two types of attribute comparison: *syntactical* and *semantical* comparison.

In the case of syntactical comparison, attributes are compared without any external knowledge. As a consequence, the comparison of attribute values focuses on similarities between the lexical form of these values. Such comparisons typically take into account typographical errors, inaccuracies in measurement, abbreviations, etc. In Table 1.1, the values of the attribute “name” can be compared purely syntactically by using, for example, the edit distance metric [12] which is defined as the minimal number of edit operations that are needed to transform one string into another. Similarly, the attribute values of “longitude” and “latitude” can be compared by using a metric on the set of real values.

In the case of semantical comparison, attributes are compared by using external knowledge, e.g., in the form of an ontology. Typically, this type of comparison tries to exploit connections between values that cannot be determined only by considering the values alone. Examples are synonyms and value generalisation/specialisation. For instance, in Table 1.1, the domain of the attribute “category” representing information on the type or function of a point of interest (POI) may be considered as partially ordered by means of a generalisation/specification relation and whose values should be compared by taking into account some hierarchical connections between them. A part of that partial order relation is presented

Figure 1.1 A part of the partial order relation for the category attribute from the source dataset S (Table 1.1).



in Figure 1.1, which contains categories from Table 1.1 in the filled squares. However, an important and relevant conclusion from [11] states that, in many cases, it is unfeasible to use a *predefined* knowledge base during coreference detection. The reasons are that such a knowledge base might not exist or that the database under consideration lacks standardisation, thus making it difficult to link the actual data to the knowledge base. As a consequence, dynamical construction of a knowledge base is an important aspect of improving the detection of coreferent objects.

Moreover, the knowledge base can be useful not only to detect coreferent objects but also to *fuse* values of the coreferent objects' attributes from homogeneous and heterogeneous data sources.

In the case of homogeneous data sources, where all data sources follow the same schema with the same domains for corresponding attributes, *data fusion* is another crucial operation in the maintenance of data quality which removes mutually duplicate (coreferent) tuples (i.e., tuples that mutually describe the same entity) and replaces them with a tuple that minimises information loss [13, 14]. In other words, it is a function that combines multiple tuples into one and resolves possible conflicts. For instance, provided that the tuples in Table 1.1 are coreferent tuples (duplicates), the next step in data cleansing, which is the process of detecting and correcting (or removing) corrupt or inaccurate records from a dataset, is to combine them into one tuple that best represents information about the referred location. This is usually done by projecting the tuples over their attributes and by treating the attributes separately. For the POI "name", a possible solution is to choose the most frequent name. For "longitude" and "latitude", a mediating function such as the median could be considered. However, for the POI "category", finding a representative value may be more challenging. An important question

is: What causes variations in the attribute values for duplicate POIs? In the case of the POI name, variations can be attributed to spelling errors, abbreviations, etc. In the case of the POI category, variations are caused by *subjectivity*. Whereas one agent decides that the "Belfry of Ghent" is a monument, another might decide that a monument is not an adequate category and will instead annotate it within the general category "POI". In this setting again the taxonomical connection between the values of the POI category in Figure 1.1 can be used as a basis for fusing them. For instance, for coreferent tuples in Table 1.1 the category "Belfry" can be selected because it is the most specific category among all of the candidates with respect to the taxonomy shown in Figure 1.1. The result of that fusion is presented in Table 1.2.

Table 1.2 Result of fusion operation on coreferent objects in Table 1.1 (the source dataset S).

Key	Name	Lon.	Lat.	Category
7	Belfry & Cloth Hall	3.724911	51.053653	Belfry

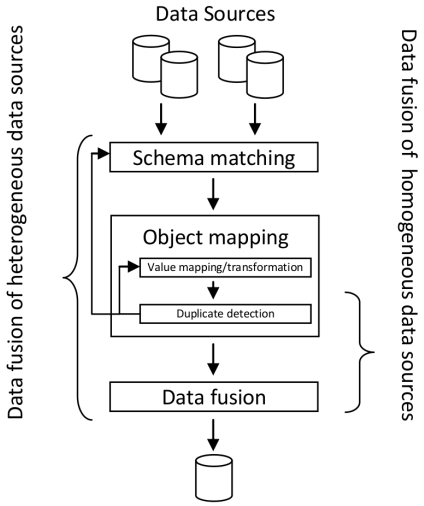
In contrast to homogeneous data sources, data fusion faces two additional types of challenges for heterogeneous data sources. Heterogeneity can exist at the schema level, where different data sources often describe the same domain using different schemas, and it can also exist at the instance level, where different sources can represent the same real-world entity in different ways [14], e.g., the category of church can be represented by a value "cathedral" in one data source or by a value "place of worship" in another data source. Thus, the whole data fusion process (also known as data integration) consists of three major steps and is shown in Figure 1.2.

The first step is called the *schema matching* problem which attempts to reconcile structural heterogeneity of data by matching schema elements across the data sources [1, 15–21] for heterogeneous data sources.

The second step, which is called the *object mapping* (record linkage, etc.) problem [3, 13, 22–28], resolves the semantic heterogeneity of data by mapping data instances across the heterogeneous data sources in substep *Value mapping - transformation* and detects coreferent tuples in substep *Duplicate detection* for homogeneous and across heterogeneous data sources. Detection of coreferent tuples, substep Duplicate detection, should be a consequence of the previous steps if we would like to detect coreferent tuples efficiently. However, detection of coreferent tuples can be performed in advance and used to establish schema matching or to map/transform values and unify different representations of the same entities.

The last step is called *data fusion* which, as described above, merges multiple, coreferent tuples into a single representation of a real-world entity. It is considered to be an optional post-processing step, because, e.g., a dataset without coreferent

Figure 1.2 Data fusion of heterogeneous and homogeneous data sources.



tuples does not need data fusion.

As an illustration, let us consider a data integration scenario of heterogeneous data sources in which the object in Table 1.2 (called the source dataset S) has to be merged with the objects in Table 1.3 (called the target dataset T). First, for successful data integration schema elements across the data sources have to be matched as in Figure 1.3. There is a need for syntactical and semantical matching of corresponding attributes. Different methods can be used to detect coreference in schemas. Some methods use only content data (e.g., coreferent tuples), others use only metadata (e.g., schema information, knowledge base), whereas hybrid methods use both data and metadata.

In the next step of data integration the semantic heterogeneity of data has to be resolved by established mappings which translate data from one representation to another. Generally, values of corresponding attributes which may need syntactical comparison, such as "name", may not be translated and might be imported to Table 1.3 without any additional processing. However, importing values of attributes which may need semantical comparison, such as "category", is less trivial because

Table 1.3 Example of coreferent objects (the target dataset T).

Key	POI name	GeoCoord1	GeoCoord2	Type
1	Belfrey	51.054898	3.721675	Bell Tower
2	St Bavo	51.054898	3.721675	Cathedral
3	Belfrey of Ghent	51.054898	3.721675	Old Building

they may often refer to the same concepts that are presented in a different way in both datasets. For instance, the concept "bell tower" is represented by the category "Belfry" in Table 1.2 and by the category "Bell Tower", "Old Building", etc. in Table 1.3. Thus, for successful data integration it is crucial not only to establish schema and data mappings but also to select proper mappings of values of the attributes representing categories where, again, a partial order relation can be useful, e.g., to select the most specific concept.

Afterwards, coreferent tuples are detected and are fused just as for homogeneous data sources. Table 1.4 contains the result of this integration. There are two objects; the first represents St Bavo cathedral in Ghent, which is an original object from Table 1.3, and the second represents the Belfry and Cloth Hall in Ghent, which is a combination of three candidates: an object from Table 1.2 and two objects with key 1 and 3 in Table 1.3. More precisely, it contains the longest name from all candidates, the average geographic coordinates and the type consistent with the target data source. Such operations help to maintain consistency and decrease the number of coreferent tuples in the integrated database, which has a major influence on data quality. This, in turn, decreases the cost of a database maintenance.

Figure 1.3 Example of schema matching of Table 1.2 and Table 1.3.

Key	Name	Lon.	Lat.	Category
7	Belfry & Cloth Hall	3.724911	51.053653	Belfry

Key	POI name	GeoCoord1	GeoCoord2	Type
1	Belfrey	51.054898	3.721675	Bell Tower
2	St Bavo	51.054898	3.721675	Cathedral
3	Belfrey of Ghent	51.054898	3.721675	Old Building

Table 1.4 Result of data fusion of objects in Table 1.2 and Table 1.3.

Key	POI name	GeoCoord1	GeoCoord2	Type
2	St Bavo	51.054898	3.721675	Cathedral
7	Belfry & Cloth Hall	51.053653	3.724911	Bell Tower

1.2 Metadata

In this dissertation we investigate how data and metadata can help to improve the detection and fusion of coreferent objects in both homogeneous and heterogeneous

data sources. Generally, metadata are data about data. More precisely, metadata can describe and define the structure and properties of any object. Thus, the National Information Standards Organization (NISO) [29] distinguishes among three types of metadata: *descriptive*, *structural* and *administrative*.

Descriptive metadata can help to discover and identify objects. In this context it can be an external knowledge base (e.g., ontology, order relation), or statistical information about instance data or keywords which describe an object. Structural metadata give a description and a definition of how the components (attributes) of an object (a tuple) are organised. For instance, a schema can define the attributes of an object, where each attribute is specified by its label, type and the set of constraints that apply to it. Finally, administrative metadata provide information to help manage the source. They refer to the technical information, including file type or when and how the file was created.

All of the above types of metadata can be used to find the real meaning of an entity. However, the techniques proposed in this dissertation are based only on the following aspects (types) of metadata. On the one hand, syntactical and semantical techniques for schema matching are proposed. In this context, database schema is considered as structural metadata. On the other hand, a partial order relation is considered as descriptive metadata which helps to increase the quality of data in the context of the detection of coreferent objects and data fusion.

Extending the notion of coreference, primarily defined with respect to two or more pieces of data (tuples, elements,...) representing the same real-world entity, we will also refer to two matched elements of some schemas, notably the attributes in relational databases and elements in XML, as *coreferent*. We will also call as coreferent two or more whole schemas for which a one-to-one matching is identified. Moreover, we will also name two values of corresponding nominal or ordinal attributes as coreferent if they denote equivalent concepts in domains of those attributes.

1.3 Problem statement

The purpose of the dissertation is to develop a set of novel algorithms which in a comprehensive way address various aspects of coreferent data detection using data itself, associated metadata or both data and metadata. The algorithms proposed give better or at least comparable results to the ones obtained using some well known approaches what is confirmed in extensive computational experiments. The following contributions of this dissertation may be distinguished.

Question 1: How does one establish schema matching based only on the schema itself?

A novel automatic method for detecting coreferent elements in XML schemas based only on metadata (schema information) and also, as a next step, a method for detecting coreference of XML schemas are proposed. More specifically, detection of coreferent elements in XML schemas based only on a comparison of the elements' names (tags) and their sequences (*paths*) is studied in this dissertation. Thus, this is a syntactical (lexical) schema matching. The detection of coreference of whole XML schemas takes into account the importance of coreferent elements. To determine the importance of elements we use other XML schema information, such as cardinality, as well as order and depth of elements in the schemas, etc. However, elements names may be not informative enough and lead to wrong conclusions regarding the coreference, e.g. when they are synonyms or homonyms. Thus, the second research question is considered.

Question 2: How does one establish schema matching based on content data?

A novel automatic method for detecting corresponding attributes in schemas based on content data is studied. More specifically, our proposed method for the detection of coreferent attributes in schemas is based on a statistical and lexical comparison of content data and already earlier detected coreferent tuples across multiple datasets, which increase the possibility of correct schema matching. We will show that knowledge of even a small number of coreferent tuples is sufficient to establish correct matching between corresponding attributes of heterogeneous schemas. The behaviour of the novel schema matching technique has been evaluated on several real life datasets, giving a valuable insight in the influence of the different parameters of our approach on the results obtained. However, even then the same information in content data can be represented in different ways. Thus, some knowledge base is necessary to resolve this problem. Hence, the third research question is investigated.

Question 3: How can a knowledge base be dynamically constructed and used to improve the detection of coreferent tuples and data fusion in homogeneous or heterogeneous data collections?

An algorithm is proposed for the automated construction of a partial order relation over the domain of an attribute whose values may need a semantical comparison and can be sorted by means of an order relation that reflects a notion of generality. Such an attribute will be referred to as a *category attribute*. The input for this

algorithm is a list of a number of known coreferent tuples in a dataset. The generated order relation can be used to improve detection of further coreferent tuples and fusion of duplicate values for that attribute. Our approach has the advantage that there is no need for a priori taxonomical knowledge on the attribute domain and that the order relation automatically adapts to the values in the dataset. Moreover, we also studied how a partial order relation can best be used in the context of data fusion. A new strategy is proposed called *balanced selection* which adopts the sort-and-select principle. Candidate values for the category attribute coming from a number of tuples being fused are combined into a single value in two steps. First, they are sorted using a generality relation. Next, the most specific value that is comparable to all others is chosen as a result of the fusion function. The behaviour and (dis)advantages of our methods are experimentally investigated on large real-life datasets. Moreover, an additional semantical mapping (transformation) between values of the heterogeneous data sources can be required. Therefore, the fourth research question is addressed.

Question 4: How does one establish semantical mappings between the values of attributes and how a partial order relation can help in the data fusion of heterogeneous data sources?

A novel approach for a specific part of the object mapping problem is proposed as an answer to this research question. More specifically, novel automatic value-mapping methods for values of categorical attributes are proposed under the assumption that the partial order relation is given. This novel mapping consists of two main phases. The first phase creates one-to-many mappings between coreferent values using *explicit* and *implicit* mappers which means that any value from one source can be mapped to more than one value in another source. Explicit mappers are based on dynamically constructed textual descriptions of attributes values and values' definitions that are extracted from the world wide web. These descriptions are pairwise compared using information retrieval techniques to establish mappings between coreferent values. The created explicit mappings and the known partial order relations are employed to derive additional mappings by implicit mappers. In the second phase the one-to-many mappings are transformed into one-to-one mappings, if possible at the same level of abstraction, using a novel heuristics. Beside these mappings, called *default mappings*, our approach creates mappings which are specific for the particular object using onomastic information. These proposed techniques are also evaluated on large real-life datasets.

1.4 Preliminaries

Before we present the details of elaborated algorithms we will first introduce some relevant basic concepts. We start with the notions related to the fuzzy set theory and the possibility theory. It plays an important role in all algorithms proposed in this dissertation. Namely, the possibility theory or, more specifically, the possibilistic truth values are used to assess the (un)certainty regarding the discovered coreference between particular data and metadata elements under consideration. Next, we briefly recapitulate the concepts and notations of the data models which are considered in this work. Then, we more formally define the problem of coreference detection, which is the main problem addressed in this dissertation.

1.4.1 Fuzzy sets and possibility theory

Fuzzy sets Let U be a universe of discourse, with a generic element of U denoted as u, $U = \{u\}$. A *fuzzy set* F in U [30] is characterized by a *membership function* $\mu_F(u)$ which associates with each element $u \in U$ a real number in the interval [0,1]. The values of $\mu_F(u)$ represent the "grade of membership" of u in F.

The basic operations of union, intersection and complement on fuzzy sets are defined as follows:

$$\mu_{A \cup B}(u) = \max(\mu_A(u), \mu_B(u)) \tag{1.1}$$

$$\mu_{A \cap B}(u) = \min(\mu_A(u), \mu_B(u)) \tag{1.2}$$

$$\mu_{\overline{A}}(u) = 1 - \mu_A(u) \tag{1.3}$$

Fuzzy restriction Let X be a variable which takes values in a universe of discourse U and let $X = u$ mean that X is assigned the value u, $u \in U$. Let F be a fuzzy set in U which is characterized by a membership function μ_F. Then F is a *fuzzy restriction on* X (or *associated with* X) if F acts as an elastic constraint on the values that may be assigned to X in the sense that the assignment of a value u to X has the form

$$X = u : \mu_F(u) \tag{1.4}$$

where $\mu_F(u)$ is interpreted as the degree to which the constraint represented by F is satisfied when u is assigned to X. Let $R(X)$ denote a fuzzy restriction associated with X. Then, to express that F plays the role of a fuzzy restriction in relation to X, we write $R(X) = F$ [31].

Possibility distribution Let F be a fuzzy set in U, with the grade of membership, $\mu_F(u)$, interpreted as the compatibility of u with the concept labeled F. Let X be a variable taking values in U, and let F act as a fuzzy restriction, $R(X)$, associated with X. Then the proposition "X is F" yields a *possibility distribution*

on U, π_X, which is postulated to be equal to μ_F, i.e., $\pi_X(u) = \mu_F(u)$. $\pi_X(u)$ is interpreted as a degree to which it is *possible* that the value of X is equal u [31].

Possibility measure Let A be a fuzzy set in U and let π_X be a possibility distribution associated with a variable X which takes values in U. The *possibility measure*[1] of A, denoted $\Pi_X(A)$ and expressing the possibility that the value of X is in A, denoted $Possibility(X \text{ is } A)$, is defined by

$$Possibility\{X \text{ is } A\} \triangleq \Pi_X(A) \triangleq \sup_{u \in U}(\mu_A(u) \wedge \pi_X(u)), \tag{1.5}$$

where $\mu_A(u)$ is the membership function of A, and $\wedge$ stands for the minimum operator [31].

Necessity measure In possibility theory, the certainty concerning the statement that the value of X is in A, denoted $Necessity(X \text{ is } A)$, is expressed by *the necessity measure* $\mathrm{N}_X(A)$, defined with respect to a possibility measure $\Pi_X(A)$ as follows:

$$Necessity(X \text{ is } A) \triangleq \mathrm{N}_X(A) \triangleq 1 - \Pi_X(\overline{A}) \tag{1.6}$$

with $\overline{A}$ denoting the complement of the fuzzy set A [31].

1.4.2 Multisets

Within the context of this work, the framework of the set theory will not suffice to present our approach. Instead, the more general framework of *multisets* (also called *bags*) will be used where necessary and the definitions by Yager [32] are adopted here.

A *multiset* A over a universe U is defined by a function $A : U \to \mathbb{N}$. For each $u \in U$, $A(u)$ is a non-negative integer denoting the *multiplicity* (i.e., the number of occurrences) of u in A. The set of all multisets drawn from a universe U is denoted $\mathcal{M}(U)$. Yager has defined some basic operations on multisets. The j-cut of a multiset A is a regular set, denoted as A_j and is given by $A_j = \{u | u \in U \wedge A(u) \geq j\}$. The counterparts of the classical set intersection and union operations are defined as follows:

$$\forall u \in U : (A \cup B)(u) = \max(A(u), B(u)) \tag{1.7}$$

$$\forall u \in U : (A \cap B)(u) = \min(A(u), B(u)). \tag{1.8}$$

The subsethood (inclusion) for multisets is defined as follows:

$$A \subset B \Leftrightarrow \forall u \in U \; A(u) < B(u) \tag{1.9}$$

$$A \subseteq B \Leftrightarrow \forall u \in U \; A(u) \leq B(u). \tag{1.10}$$

[1] The possibility measure may be defined in a more general, axiomatic, way without a reference to the concept of the possibility distribution but the definition given here will serve our purposes.

The theory of multisets provides also an addition operator and a subtraction operator:

$$\forall u \in U : (A \oplus B)(u) = A(u) + B(u) \tag{1.11}$$

$$\forall u \in U : (A \ominus B)(u) = \max(A(u) - B(u), 0). \tag{1.12}$$

The cardinality of a multiset A is calculated as the sum of all multiplicities:

$$|A| = \sum_{u \in U} A(u). \tag{1.13}$$

Finally, it is said that an element u belongs to the multiset A, denoted as $u \in A$, if $A(u) \geq 1$. Formally:

$$\forall_{A \in \mathcal{M}(U)} \forall_{u \in U}\ u \in A \Leftrightarrow A(u) \geq 1. \tag{1.14}$$

1.4.3 The data models

The relational and XML data models are extensively used in this dissertation.

The relational model

It is assumed that the reader is familiar with the relational database model as proposed by Codd [33]. For the sake of completeness, we shall briefly recapitulate the basic concepts and notations of this model that are relevant to the dissertation. Let $\mathcal{A}$ be a countable set of attributes. A *(relational) schema* $\mathcal{R}$ is defined by a non-empty and finite subset of $\mathcal{A}$. For each attribute $a \in \mathcal{A}$, let $\mathrm{dom}(a)$ denote the domain of a. That is, $\mathrm{dom}(a)$ represents the set of all possible values for attribute a. Having a relational schema $\mathcal{R} = \{a_1, ..., a_k\}$, a *relation* R over $\mathcal{R}$ is defined by a finite set $R \subseteq \mathrm{dom}(a_1) \times ... \times \mathrm{dom}(a_k)$. Each element of a relation R with schema $\mathcal{R}$ is called a tuple over $\mathcal{R}$. In the remainder, we shall denote an arbitrary tuple by t. For simplicity, we shall denote the combined universe $\mathrm{dom}(a_1) \times ... \times \mathrm{dom}(a_k)$ as $\mathrm{dom}(\mathcal{R})$. A *database schema* $\mathcal{D}$ is defined by a non-empty and finite set of relational schemas, i.e., $\mathcal{D} = \{\mathcal{R}_1, ..., \mathcal{R}_n\}$ and a database D over $\mathcal{D}$ is defined as a set of relations $\{R_1, ..., R_n\}$ where R_i is a relation over $\mathcal{R}_i$, for any i.

One of the standard querying languages of the relational data model is the *relational algebra*. It comprises a set of operations which take relations as arguments and produce other relations. A sequence of operations applied to appropriate relations makes it possible to retrieve the data needed by a user. Let us remind one of this operations, the *projection*. Let R be a relation over a schema $\mathcal{R}$ and consider a set of attributes $A \subseteq \mathcal{R}$. Then the result of the *projection operation* of R over A is a relation, denoted $R[A]$, with schema A that is obtained by taking the tuples in R and retaining only the values of attributes in A. The part of a tuple $t \in R$

comprising only attributes from A is denoted as $t[A]$ ($t[a]$ in the case where A is a singleton $\{a\}$). It is stressed here that the result of a projection, $R[A]$, is a *set*, i.e., if there are two tuples $t, s \in R$ such that $t[A] = s[A]$ then only one of them is preserved in $R[A]$.

The XML model

It is also assumed that the reader is familiar with the Extensible Markup Language (*XML*) [34]. For the sake of completeness, we shall briefly recapitulate the concepts of this model that are relevant to the dissertation.

An XML document represents the semi-structured information. The basic components of an XML document are *elements* which are composed of an opening and a closing *tag* and are used to mark up the sections of an XML document. Elements form a structured part of a document and may contain other nested elements or unstructured content. Tags are the names of elements and provide a clue on the content of an element. Elements may have attributes associated with them which usually provide some additional information on the content of an element.

There are some syntactical rules which have to be followed to obtain a *well-formed* XML document. For example, the closing tag of a nested element have to appear inside the host element. There may be also other rules stating, e.g., which elements may be nested in which elements. These rules form an XML document *schema*. A document following these rules is called *valid*.

An XML document forms a tree-like structure. The root corresponds to a root element which contains, nested within it, elements of the first level which in turn can contain nested elements, and so on. An element in such a tree, i.e., in an XML document, may be identified with a *path* from the root element down to the given element. The components of such a path, called later on as *steps*, are the names of the nested elements. As these names are usually assumed to be meaningful, thus a path may enrich the semantics of an element.

1.4.4 Object coreference detection

Considering a more abstract view of entity representation, denoting the universe of the i-th feature of an object by U_i, we can model the universe O of objects by:

$$O = U_1 \times \cdots \times U_n. \tag{1.15}$$

Two objects $o_1 \in O$ and $o_2 \in O$ are said to be *coreferent* (denoted $o_1 \leftrightarrow o_2$) if and only if they describe the same real world entity.

Two elementary operators play an important role in establishing the coreference of objects: a *comparison operator* working at the level of object features (or metadata features, e.g. tags, paths) and an *aggregation operator* combining the comparison scores obtained for particular features.

Definition 1 (Comparison operator)**.** *A comparison operator on the universe O is defined by a function $\mathcal{C}$:*

$$\mathcal{C} : O^2 \rightarrow \mathbb{L} \tag{1.16}$$

where $(\mathbb{L}, \leq)$ is a totally ordered and bounded lattice.

A comparison operator $\mathcal{C}$ compares (a feature of) two objects o_1 and o_2 and expresses the result of this comparison as a *matching degree*. This matching degree may be interpreted as expressing how certain it is that both objects are coreferent and belongs to a totally ordered and bounded lattice $\mathbb{L}$. In the case of probabilistic methods, $\mathbb{L}$ can be instantiated with the unit interval $[0, 1]$ in order to express the result of comparison as a probability of coreference of the objects. Other practical examples of $\mathbb{L}$ are the set of truth values $\mathbb{B} = \{T, F\}$, where T and F denote full certainty of the match and mismatch, respectively or the set of possibilistic truth values (PTVs) [8].

In our approach we use PTVs to express the confidence (certainty) in the validity of the mappings produced by an algorithm. Hereby, a PTV is a normalized possibility distribution [31] defined over the set of Boolean values $\mathbb{B}$ [35]:

$$\text{PTV} \longmapsto \pi : \mathbb{B} \rightarrow [0, 1]$$

A PTV expresses the uncertainty about the Boolean value of a *proposition* p. We will often use the notation $\mu(T)$ and $\mu(F)$ instead of $\pi(T)$ and $\pi(F)$ assuming that the (un)certainty as to the truth of a proposition is expressed as "certainly true", "true or false" etc., represented by appropriate fuzzy sets in $\mathbb{B}$; e.g., respectively, $\mu(T) = 1$ and $\mu(F) = 0$, and $\mu(T) = \mu(F) = 1$, for the previous examples. In the context considered here, the propositions p of interest are of the form:

$$p \equiv o_1 \text{ and } o_2 \text{ are coreferent}$$

where o_1 and o_2 are two objects.

Let P denote a set of all propositions under consideration. Then each $p \in P$ can be associated with a PTV denoted $\tilde{p} = \{(T, \mu_{\tilde{p}}(T)), (F, \mu_{\tilde{p}}(F))\}$, where $\mu_{\tilde{p}}(T)$ represents the possibility that p is true and $\mu_{\tilde{p}}(F)$ denotes the possibility that p is false. In what follows, PTVs are often noted as couples ($\mu_{\tilde{p}}(T)$, $\mu_{\tilde{p}}(F)$). It is assumed that each PTV is *normalized* which means that max($\mu_{\tilde{p}}(T)$, $\mu_{\tilde{p}}(F)$) = 1. The space of all possibilistic truth values is denoted $\mathcal{F}(\mathbb{B})$, i.e., it comprises all (normalised) fuzzy sets over $\mathbb{B}$.

Let us define the order relation $\geq$ on the set $\mathcal{F}(\mathbb{B})$ by:

$$\tilde{p} \geq \tilde{q} \Longleftrightarrow \text{if}((\mu_{\tilde{p}}(F) \leq \mu_{\tilde{q}}(F)) \text{ and } (\mu_{\tilde{p}}(T) = \mu_{\tilde{q}}(T) = 1)) \text{ or } (\mu_{\tilde{q}}(T) \leq \mu_{\tilde{p}}(T)) \tag{1.17}$$

Moreover, two thresholds ($threshold_T$ and $threshold_F$) are employed to decide on two objects coreference. If $\mu_{\tilde{p}}(F)$ is lower than the $threshold_F$ then

coreference is declared. If $\mu_{\tilde{p}}(T)$ is lower than the $threshold_T$ then a lack of coreference is declared. Finally, if both of the thresholds are exceeded then the coreference status is declared as being *unknown*.

Comparison of complex objects is usually a two-stage process. First, parts of objects, notably values of their features, are compared using a comparison operator. Thus, we extend our definition of the comparison operator (1.16) so as to make it applicable also to scalar feature values:

$$\mathcal{C}_i : U_i^2 \to \mathbb{L} \tag{1.18}$$

In this way a separate comparison operator $\mathcal{C}_i$ can be defined for each feature. Then, the results of those comparisons are aggregated to obtain an overall matching degree reflecting the coreference of the whole objects being compared. Therefore, another elementary operator, an *aggregation operator*, is needed.

Definition 2 (Aggregation operator)**.** *An aggregation operator on* $\mathbb{L}$ *is defined by a function* $\mathcal{A}$*:*

$$\mathcal{A} : \mathbb{L}^n \to \mathbb{L} \tag{1.19}$$

where $(\mathbb{L}, \leq)$ *is a totally ordered and bounded lattice.*

For more information on aggregation operators the reader is referred to [36]. We assume an aggregation operator $\mathcal{A}$ to be idempotent:

$$\forall\, l \in \mathbb{L} : \mathcal{A}(l, l, ..., l) = l \tag{1.20}$$

Besides that, we assume that $\mathcal{A}$ is monotone in the following sense:

$$\forall\, (\boldsymbol{l}, \boldsymbol{l'}) \in \mathbb{L}^n \times \mathbb{L}^n : \boldsymbol{l} \leq \boldsymbol{l'} \Rightarrow \mathcal{A}(\boldsymbol{l}) \leq \mathcal{A}(\boldsymbol{l'}) \tag{1.21}$$

where the relation $\leq$ is generalized from $\mathbb{L}$ to vectors from $\mathbb{L}^n$ in a point wise way.

Based on the definition of these two elementary operators, a comparison of two objects can be generally written as:

$$\mathcal{C}(o_1, o_2) = \underset{i=1}{\overset{n}{\mathcal{A}}} \left(\mathcal{C}_i \left(u_{i1}, u_{i2}\right)\right) \tag{1.22}$$

where u_{i1} and u_{i2} denote the value of the i^{th} feature of o_1 and o_2, respectively.

In our approach $\mathbb{L}$ is the space of all PTVs endowed with the relation given in (1.17). Aggregation of PTVs may be carried out using the *Sugeno integral for possibilistic truth values* as defined in [37]; cf. also [38] for the original, general definition of the Sugeno integral. This integral uses two *fuzzy measures* (γ^T and γ^F) which are defined below. Let us first remind briefly the definition of a *fuzzy measure*.

Definition 3 (Fuzzy measure)**.** *A* fuzzy measure *on a finite universe U is a set function $\gamma : \mathcal{P}(U) \rightarrow [0, 1]$ that satisfies the following properties:*

$$\gamma(\emptyset) = 0 \tag{1.23}$$

$$\gamma(U) = 1 \tag{1.24}$$

$$A \subseteq B \Rightarrow \gamma(A) \leq \gamma(B) \tag{1.25}$$

Then, in the context considered here, the measure $\gamma^T(A)$ (resp. $\gamma^F(A)$) provides the assessment of certainty that two complex objects are (not) coreferent, given that the set of (metadata) features A are (not) coreferent. As required by the definition of fuzzy measures, γ^T and γ^F are monotonic and satisfy the boundary conditions of a fuzzy measure.

Definition 4 (Sugeno integral for PTVs [37])**.** *Given a set of propositions $P = \{p_1, ..., p_n\}$ and a corresponding set of PTVs $\tilde{P} = \{\tilde{p}_1, ..., \tilde{p}_n\}$, let γ^T and γ^F be two fuzzy measures defined on P which satisfy the condition:*

$$\forall Q \subseteq P : \min(\gamma^T(Q), \gamma^F(\bar{Q})) = 0 \tag{1.26}$$

where $\bar{Q}$ denotes the complement of Q.

Then, the Sugeno integral of $\tilde{P}$ with respect to γ^T and γ^F is defined by:

$$S_{\gamma^{T,F}}(\tilde{P}) : \mathcal{F}(\mathbb{B})^n \rightarrow \mathcal{F}(\mathbb{B}) : \tilde{P} \mapsto \tilde{p}, \text{ where} \tag{1.27}$$

$$\mu_{\tilde{p}}(T) = 1 - \bigvee_{i=1}^{n} N_{\tilde{p}_{(i)}}\Big(F\Big) \wedge \gamma^F\Big(P_{(i)^F}\Big) \tag{1.28}$$

and

$$\mu_{\tilde{p}}(F) = 1 - \bigvee_{i=1}^{n} N_{\tilde{p}_{(i)}}\Big(T\Big) \wedge \gamma^T\Big(P_{(i)^T}\Big) \tag{1.29}$$

where $(\cdot)^T$ (respectively $(\cdot)^F$) is a permutation that orders the elements of $\tilde{P}$ non-increasingly (non-decreasingly), while $P_{(i)^F}$ and $P_{(i)^T}$ are sets of propositions p_j with, respectively, i largest values $\mu_{\tilde{p}_j}(F)$ and i largest values $\mu_{\tilde{p}_j}(T)$.

Remark. The motivation to use PTVs and the Sugeno integral is the following. We would like to show that taking into account similarity and dissimilarity of objects in each step separately may be advantageous. In fact, De Cooman [39] has shown in his formal analysis of PTVs that it is essential that possibilities for true and false can be measured separately. To this aim, the aggregated PTVs indicate

both the coreference and the lack of coreference of paths/schemas. The choice for the Sugeno integral is motivated by the ability of the related fuzzy measures to model complex preferences in the regular case, making the Sugeno integral a very powerful and flexible aggregation operator [37]. The research on the aggregation of *bipolar* information (here: for and against the coreference) is not that developed in the literature and the Sugeno integral is a prominent example of an aggregation operator adopted for this setting. Besides that, the experimental results confirm that this is a promising choice. An alternative aggregator can be, e.g., an Ordered Weighted Conjunction (OWC) [8] which is in fact a special case of the Sugeno integral.

1.4.5 Cardinality of a set of PTV qualified propositions

In this thesis we will often use (multi)sets of Boolean propositions, certainty of truth of which will be expressed with a PTV associated with each proposition. We will then use the concept of a kind of the cardinality of such a (multi)set which counts those propositions fully certain to be true (i.e., with a PTV (1,0) assigned) as 1, does not count at all propositions fully certain to be false (i.e., with a PTV (0,1) assigned), and counts the remaining propositions to some degree belonging to [0,1] and depending on how their PTVs are close to (1,0) or (0,1). In fact, this cardinality is similar to a fuzzy cardinality of fuzzy sets and will be expressed as a possibility distribution on the set of integers. We will denote this cardinality as $\pi_{\mathbb{N}}$ (provided that from the context it will be clear which set of propositions it concerns). We will call it also sometimes as a *fuzzy integer* due to the fact that its possibility distribution is assumed to be a convex function, in the same sense as membership functions of fuzzy numbers are assumed, i.e., every α-cut of this function (interpreted as a membership function of a fuzzy set) is an interval, i.e., contains all integers between the lowest and highest integers belonging to this α-cut.

In [40], a method is proposed to construct such a possibility distribution (fuzzy integer) for a (multi)set of propositions associated with PTVs P. In fact, this method may be treated as constructing a possibility distribution which expresses the possibility that an integer k represents the number of true propositions in P.

Definition 5 (Cardinality of a set of PTV qualified propositions)**.** *Let P be a multiset of independent Boolean propositions and let $\tilde{P}$ be the multiset of corresponding possibilistic truth values, i.e. $\forall p \in P : \tilde{p}$ is a PTV associated with p and expressing the (un)certainty as to the truth of p and let $\tilde{p}_{(i)}$ denote the i^{th} largest possibilistic truth value with respect to the order relation defined by Equation 1.17. The quantity of true propositions in P is given by the following possibility distribution on*

the set of all integers (fuzzy integer):

$$\pi_{\mathbb{N}}(k) = \begin{cases} \mu_{\tilde{p}_{(1)}}(F) & , k = 0 \\ \mu_{\tilde{p}_{(k)}}(T) & , k = |P| \\ \min\left(\mu_{\tilde{p}_{(k)}}(T), \mu_{\tilde{p}_{(k+1)}}(F)\right) & , \textit{else.} \end{cases} \tag{1.30}$$

This definition states that $\pi_{\mathbb{N}}(k)$ is the minimum of the possibility that at least k propositions are true and the possibility that at least $|P| - k$ propositions are false.

Let us define an order relation $\prec_{sup}$ on the set of such possibility distributions (fuzzy integers).

Definition 6 (Sup-order of fuzzy integers)**.** *The order relation $\prec_{sup}$ is defined for fuzzy integers as follows:*

$$\tilde{n} \prec_{sup} \tilde{m} \Leftrightarrow \sup \tilde{n}_\alpha < \sup \tilde{m}_\alpha \tag{1.31}$$

Hereby, $\tilde{n}_\alpha$ is the α-cut of $\tilde{n}$, which is treated here as a fuzzy set, and α is chosen such that:

$$\alpha = \sup\{x | \sup \tilde{n}_x \neq \sup \tilde{m}_x\} \tag{1.32}$$

Thus defined partial order coincides with the standard relation "<" on integers when fuzzy integers under consideration are regular integers, i.e., their possibility distributions are $\pi_{\mathbb{N}}(k) = 1$ for a $k \in N$ and $\pi_{\mathbb{N}}(l) = 0$ for $l \neq k$.

1.5 Outline of the thesis

The remainder of this dissertation is organised as follows. In the next two chapters, solutions are proposed for the schema matching problem. Namely, in Chapter 2, the schema matching approach based on schema information only is investigated, whereas the schema matching approach based on coreferent data is studied in Chapter 3.

In Chapter 4, an algorithm for Dynamical Order Construction (DOC), which constructs a partial order relation over the domain of an attribute whose values may need semantical comparison and can be sorted by the specialisation/generalisation relation, is introduced. Important features of this algorithm are pointed out and the parameters are discussed. Moreover, the algorithm's applicability for data coreference detection is studied. Usage of the DOC method in the context of data fusion is investigated and selection strategies are evaluated in Chapter 5. In Chapter 6, a partial order relation over the domain of an attribute is applied to automatically establish semantical mappings for attribute values with different domains in heterogeneous collections.

Finally, Chapter 7 summarises the most important contributions of this dissertation responding to the research questions.

1.6 Publications

A1: Papers in journals indexed by the Thomson Reuters Web of Science

- Szymczak, M., Zadrożny, S., Bronselaer, A. & De Tré, G. "Coreference detection in an XML Schema," Information Sciences (2015) pp. 237-262.
- Bronselaer, A., Szymczak, M., Zadrożny, S. & De Tré, G. "Dynamical Order Construction in Data Fusion," Information Fusion. (Under review)

P1: Papers in conference proceedings indexed by the Thomson Reuters Web of Science

- Szymczak, M., Zadrożny, S., & De Tré, G. "Coreference detection in XML metadata," IFSA World Congress NAFIPS Annual Meeting 2013, Proceedings. Edmonton, Canada, 2013.
- Szymczak, M., Bronselaer, A., Zadrożny, S., & De Tré, G. "Dynamical construction of binary relations in coreference detection," NAFIPS Annual Metting 2012, Proceedings. Berkeley, CA, USA, 2012.

C1: Conference paper and B2: Chapter in a book as author or co-author.

- Szymczak, M., Bronselaer, A., Zadrożny, S., & De Tré, G. "Selection of Semantical Mapping of Attribute Values for Data Integration," In P. Angelov et al. (Eds.), Advances in Intelligent Systems and Computing, Vol. 322, pp. 581-592, Switzerland: Springer 2014. Presented at the 7th IEEE International Conference Intelligent Systems, IS 2014, Warsaw, Poland.
- Szymczak, M., Bronselaer, A., Zadrożny, S. & De Tré, G. "Semantical Mapping of Attribute Values for Data Integration," IEEE 2014 Conference on Norbert Wiener in the 21st Century, NAFIPS Annual Meeting, Proceedings. Boston, USA, 2014.
- Szymczak, M., & Koepke, J. "Matching methods for semantic annotation-based XML document transformations," In K. Atanassov et al. (Eds.), New developments in fuzzy sets, intuitionistic fuzzy sets, generalized nets and related topics. Vol. II: Application, pp. 297—308, Warsaw, Poland, 2012.

2

Coreference detection in schema based on schema alone

The following publications have been based on the contents of this chapter:

- M. Szymczak, S. Zadrożny, A. Bronselaer, and G. De Tré, "Coreference detection in an XML schema," Information Sciences, vol. 296, pp. 237-262, 2015.
- M. Szymczak, S. Zadrożny, and G. De Tré, "Coreference detection in XML metadata," IFSA World Congress NAFIPS Annual Meeting, Proceedings. Edmonton, Canada, 2013.

2.1 Introduction

In this chapter, a novel approach is proposed for the first step of data integration, namely for schema matching. Our technique helps to discover coreferent schema elements in an automatic fashion based only on schema information and, more specifically, on the names of particular elements. Thus, this is a syntactical approach. The schema matching method is investigated in terms of importance of elements. An experimental evaluation of our method shows the role of its parameters and its performance in comparison to other well-known schema matching approaches.

XML [34] was chosen as the data model for multiple reasons. First, XML is one of the most popular formats to store and exchange data. Second, it consists of two

well-defined layers: a metadata layer (XML schema) and a data layer (XML document). Third, XML schemas can contain complex data structures, which makes the detection of coreferent schema elements[1] a challenging task. Moreover, it is platform-independent and often contains information that is represented in different ways. XML is the basis for many web services and especially for semantic web services offered in the framework of the Semantic Web. Moreover, XML allows to define models to control data quality from its content [41, 42]. Namely, these models use the power of XML Schema language to improve the representation of documents in the Web with semantic characteristics related to their quality and thus it is useful to search quality resources in XML format.

2.1.1 Problem illustration

The existence of duplicate data across multiple, related databases significantly lowers data quality and should be avoided. As an example, Figure 2.1 shows two data pieces (two instances with hierarchies of elements) resp. taken from FreeDB[2] (right tree) and Discogs[3] (left tree) which describe the same entity (an instance of the concept "compact disc", more precisely CD album "Metallica - Metallica"), but in different ways. An example of coreferent data is presented in Figure 2.1. Coreferent data (elements with values) which describe each object, i.e.: the title ("Metallica"), artist ("Metallica") or year ("1991"), are linked by arrows.

Equally important as detection of coreferent data is the detection of coreferent metadata (in this context these are the schema elements), which is investigated in this chapter. In the case of XML, metadata define the structure, hierarchy and constraints of data. The basic metadata are *tags*. A tag is also known as a *name of an XML element*. It provides the name for a specific element at the meta-level and also (as a consequence) at the data level. The tags are the filled rectangles in Figure 2.1 and in Figure 2.2, where they form the structure and hierarchy of the schema. The structure and hierarchy specify the parent-child relationships between the elements (tags). In consequence, a sequence of tags from the root to the leaf in an XML schema (called a *path*) precisely defines the context and location of a specific element, e.g., the sequence of tags "cddb/disc/dtitle" from the right schema in Figure 2.2. Paths can be considered as one of the main components of an XML schema and can be compared in a syntactical manner, as they are in this chapter.

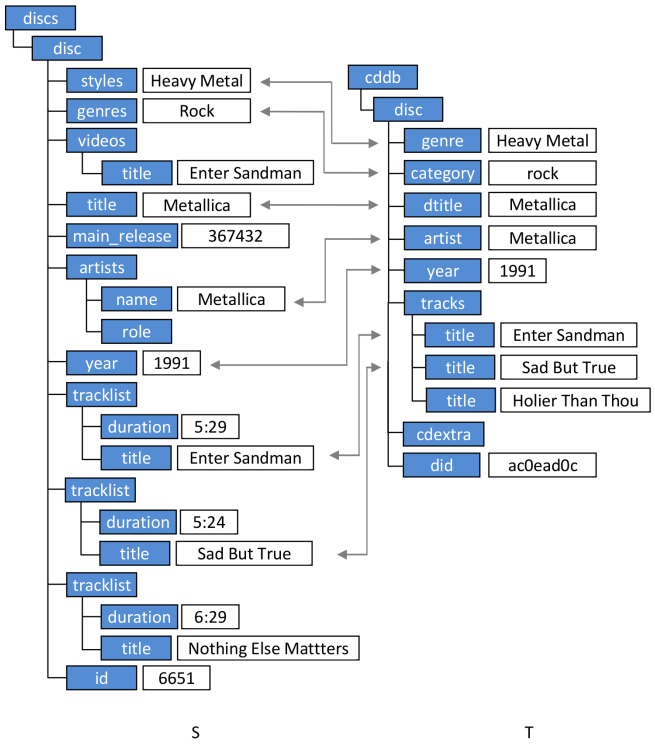
Detection of coreferent metadata (schema elements) makes it possible to establish at least a partial mapping between the data of two XML schemas. As an example, Figure 2.2 shows a mapping between elements of the metadata layer (hi-

[1] In the case of XML, schema elements are defined by XML elements or XML attributes and they are generally called *(XML) elements* in the scope of this chapter considering a more abstract point of view.

[2] FreeDB, http://www.freedb.org/

[3] Discogs, http://www.discogs.com/data/

Figure 2.1 Real-world data example: parts of XML documents from Discogs (left tree - S) and FreeDB (right tree - T) datasets. The filled boxes are the XML tags (elements) defined in the XML schema, while the not-filled are values of specific elements. Arrows represent mappings of actually coreferent data.

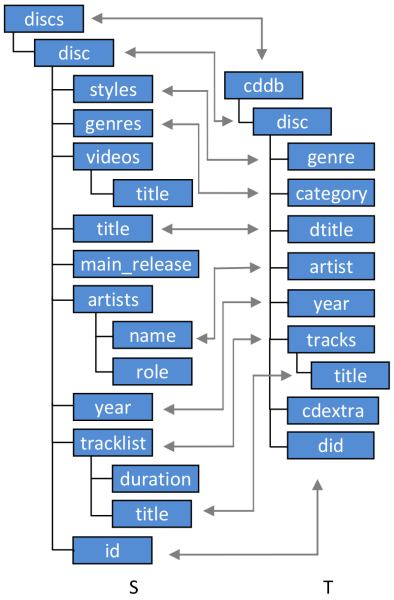


erarchy of data), respectively, taken from FreeDB (right tree) and Discogs (left tree). These mappings are in addition to mappings between coreferent XML data which are presented in Figure 2.1. For instance, in Figure 2.2 a mapping based on metadata is established between the elements “id” and “did” because the paths related to these elements may be recognized as coreferent. Moreover, the values present *on the paths* (content data) should help to identify the coreferent schema elements, but this is beyond the scope of this chapter and is investigated in Chapter 3. In this chapter, a novel syntactical schema matching approach based on path comparison only is proposed.

Many problems have to be addressed when devising such schema matching techniques. The most important among these are the following:

- How to establish schema matching when objects have different structures and a different organisation of the same content?

Figure 2.2 Real-world metadata example: XML schema elements with hierarchies extracted from Discogs (left tree - S) and FreeDB (right tree - T) datasets. Arrows represent matchings of coreferent metadata (schema elements).



- How to establish schema matching if abbreviations or typing errors exist in the tags?
- How to establish schema matching efficiently?

2.1.2 Contributions

The objective of this chapter is to propose a novel, automatic, syntactical method for detecting coreferent elements in XML schemas based only on metadata and, also, as a next step, to propose a method for detecting coreference of XML schemas. More specifically, the detection of coreferent elements in XML schemas based only on a comparison of the element names (tags) and their sequences (*paths*) is studied in this chapter. Detecting the coreference of whole XML schemas takes into account the coreference of their respective elements and the importance of the elements in the schema. To determine the importance of elements, some novel heuristics are proposed. Schema information, such as cardinality, order and element depth, is used. The novel heuristics allow to increase the quality of the results of the coreference detection method for XML schemas.

Our goal was to provide a novel method that would be general enough so that it could establish the matching of two (XML) schemas (or their parts/paths) with

limited information available (we assume that only the schemas are given). This may be useful, e.g., for two software agents trying to set up a mutual understanding based on their repositories of XML schemas representing information/data crucial for the purposes of their communication or for data integration of heterogeneous data sources, namely for matching of corresponding schema elements.

2.1.3 Outline

The rest of this chapter is structured as follows. Related work is presented in Section 2.2. In Section 2.3 it is presented how coreferent XML elements and coreferent XML schemas can be detected based on *paths*. Section 2.4 reports the experimental results. Finally, Section 2.5 summarises the contributions of this chapter.

2.2 Related work

There is a large body of work on schema matching which uses only schema (metadata) information (cf., e.g., [1,43–47]). In contrast to our method, most approaches are based on a few different *matchers* (methods which are able to establish a matching between corresponding elements), called combining matchers, which combine individual matching criteria, i.e., names and types of elements, constraints (value ranges restrictions, cardinalities, uniqueness constraints, referential integrity), descriptions, schema structure (i.e., parent, sibling, child, ancestor, descendant) or auxiliary information. Apart from this, combining matchers can be divided into two main subcategories: hybrid matchers and composite matchers. On the one hand, hybrid matchers use multiple matching criteria. On the other hand, composite matchers combine multiple match results that are obtained from independently executed matching algorithms.

2.2.1 Hybrid matchers

First of all, we present some important hybrid matchers which are based on schema information (cf.,e.g., [48, 49]). *Cupid* [17] is based on element, structure and linguistic level matchings, uses similarity of atomic elements (which capture a lot of semantics), exploits the internal structure and constraints, and creates context-dependent matches. Like many other approaches [21, 50–55], Cupid also uses a matching confidence level value (called similarity coefficient) which is a number from the interval [0,1]. Cupid matches elements based on structural and linguistic similarities and clusters the concepts with respect to their similarity. Besides that, Cupid is a generic approach to match different types of schemas: XML or relational. Similar to the above example is the XML Schema matching method

proposed in [56]. Like Cupid, it is a hybrid (structural and linguistic) schema matcher, but dedicated only to XML Schemas and uses predefined external knowledge bases (Wordnet [57], compatible data types table) to calculate the similarity between elements. In contrast Milne et al. [58] propose a method which calculates semantic relatedness between terms using links found within their corresponding Wikipedia articles while the most successful and well known of the corpus-based approaches is Latent Semantic Analysis (LSA) [59], which relies on the tendency for related words to appear in similar contexts. Corpus-based approaches obtains background knowledge by performing statistical analysis of large untagged document collections, thus, LSA can only provide accurate judgments when the corpus is very large, and consequently the pre-processing effort required is significant. That approaches to measure semantic relatedness can be applied for schema matching or knowledge modeling [60] and management [61].

Another hybrid approach is *SKAT* [62, 63]. SKAT is a rule-based approach. Rules are formulated in first-order logic to express match and mismatch relationships. SKAT supports element and structure matching by using a set of name matchers, inclusion relationships and structure matching. Besides that, SKAT creates 1:1 and 1:n mappings between attributes and is able to reuse general matching rules. However, a user must define the match and mismatch rules. Similar to SKAT is the ARTEMIS [55] schema integration tool that is a part of MOMIS [64] (a mediator system). ARTEMIS is a name, constraints and structure based matcher which clusters similar elements and exploits linguistic information provided by WordNet [57].

Also, Palopoli et al. propose algorithms [50–52] which (like ARTEMIS and S-Trans [65]) utilize element and structure level information. The algorithms are based on a set of user predefined synonyms and homonyms and create matchings by comparing the distances between the elements in the schema. Another approach of hybrid matching which is based on schema information is *TranScm* [66]. TranScm transforms input schemas into a graph representation (using, e.g., the *Similarity flooding algorithm* [67]) and applies predefined matching rules on each pair of nodes in top-down order to create matchings. This approach gives good results only if the top structure of the two schemas is quite similar. Other structural matchers are presented in [68–71] and concern algorithms to find mappings in a tree structure without synonym and hypernym recognition, but using only purely structural matches or twig matches [72]. Twig matching finds in an XML document tree all matches of a given twig, i.e., a subtree template. Modern twig query matching algorithms often first decompose individual path matches and then merge them to form twig matches [73].

In contrast, *DELTA* [74] does not use information about the schema structure. It groups all other available metadata about attributes into character string representations which are presented as a document. Next, information retrieval

techniques are applied on those documents to perform matching.

The approaches proposed in [75] and [76] are similar to our method - both are based on element names and paths. On the one hand, Rajesh et al. [75] present the architecture of a system which comprises a linguistic matcher and propose different heuristics to calculate similarity of elements depending on their position in the hierarchy. However, that method does not consider the importance of elements like our approach. On the other hand, *QMatch* [76] relies on the semantic and structural information encapsulated in an XML schema and extracted from auxiliary sources. In our approach we only use schema information, more specifically information on the lexical similarity of paths.

2.2.2 Composite matchers

The second group of matchers consists of composite schema matching systems (like LSD [54] or COMA [18]) which automatically combine match results.

COMA is a platform to combine results of different matchers in a flexible way. This approach copes with different aspects of match processing, i.e. aggregation of matcher specific results, match direction, match candidate selection, computation of combined similarity, different matcher usages, i.e. single matchers vs. matcher combinations, no-reuse vs. reuse approaches and also support an automatic or interactive mode. Moreover, in interactive mode a user is able to define and select a matcher and so the match or mismatch strategy for each elements couple separately. COMA is an iterative generic schema matching system. Each match iteration consists of three phases: an optional user feedback phase, the execution of different matchers and the combination of the individual match results. The algorithm executes independent matchers from a matchers library in the main step. It exploits different kinds of schema information (i.e. names, data types, synonyms tables, previous match results). The next step combines results obtained by individual matchers. Similarity values which are returned by matchers are aggregated. The aggregation takes, e.g., a maximum or average value of similarities. Finally, the algorithm chooses for each element a couple with the highest similarity value of all couples, in which the particular element occurs, exceeding a certain threshold. Besides that, COMA supports directional and undirectional matchings. Directional means that matching can only be applied to source from target or from target to source, but cannot be applied in both directions in contrast to undirectional matching. Directional schema matching might simplify the matching problem solution e.g., when one schema is significantly smaller than the other.

In Szymczak and Koepke [21] we present an automatic method to match similar elements of XML schemas based on *semantic annotations*. Semantic annotations of XML schemas allow for a semantic interpretation of the schema elements

and are addressed in the W3C recommendation SAWSDL [77]. We describe the schema elements (using an extended annotation method based on the SAWSDL specification [78]) in the form of expressions that consist of concepts and properties of a reference ontology. These expressions are directly added to the schemas. In order to check for semantic matching, the expressions are automatically translated to ontology concepts. Thus, the semantic matching is accomplished with reasoning support and knowledge from the reference ontology. A major time consuming and error-prone task in this method is the addition of semantic annotations to source and target XML schemas.

The approach presented in this work is less computationally expensive than approaches which are based on external knowledge sources, because it does not need user effort to create and manage additional sources of information or does not need manual effort to define match or mismatch rules. This is very important as it can be impossible to create a general source of knowledge which covers all different aspects. String-based similarity allows us to implement an approach which is simple, fast and based on common and available information (names of elements). Moreover, external knowledge sources can be (temporarily) unavailable (e.g., due to the Internet access failure) which decreases their usability in contrast to our approach which is self-contained. Our approach is somehow more general as it, to some extent, does not depend on the language in which the XML schema elements are named. Assuming the availability of an appropriate set of relevant knowledge bases for any language of interest may be too restrictive and not practical. Nevertheless, it is true that string-based matching is not able to deal with synonyms and polysemy. However, it is not necessary to use external knowledge base to solve these problems. As we will show in Chapter 3 coreference detection based on synonyms and polysemy is possible when instance data are used.

2.3 XML paths in coreference detection

An XML schema contains various types of metadata which define the structure of an XML document, data types, restrictions, etc. These metadata allow to reconstruct *paths*, by which we mean here sequences of elements' names connecting a root element with leaf elements in corresponding XML documents (without namespaces which will not be considered in this work).

Two elements in two XML schemas may refer to the same feature of a real-world entity (may be *coreferent*) even if they have different tags and are located at different levels of an XML file structure. This is the case of, for instance, elements represented by the path "/cddb/disc/artist" from the right schema tree and the path "/discs/disc/artists/name" from the left schema tree of Figure 2.2. Comparing their paths, as defined above, may help to discover their coreference. In this section, a novel method for detecting coreferent XML elements based only on comparing

Figure 2.3 Classification of schema matching approaches [1].

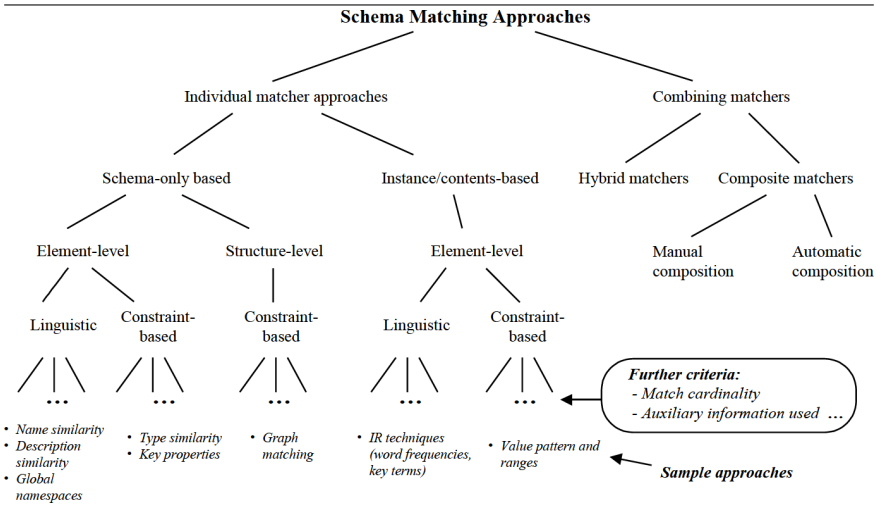
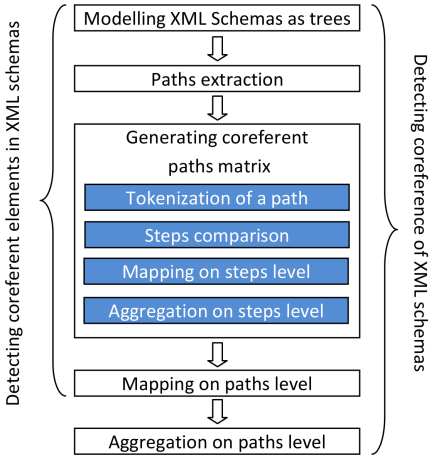


Figure 2.4 Overall scenario: XML paths in coreference detection



their paths and, as a consequence, coreference of XML schemas, is presented. Our approach is classified as the schema only based element level and name-based matching method according to the classification from [1], which is presented in Figure 2.3. The proposed method is purely syntactic and does not use any other external information sources such as ontologies or dictionaries. Using such extra information can increase the quality of coreference detection, but it also requires more computational resources and is not considered in this chapter.

Figure 2.4 presents the general steps in our approach. First, all input XML

schemas are modelled as the trees. Afterwards, paths are extracted for each leaf from a tree. Next, the coreferent paths matrix is generated. The matrix represents matchings between paths, while elements of the matrix express the (un)certainty degrees about the paths coreference (represented by PTVs) which are calculated in the following substeps: *Tokenisation of a path*, *Step comparison*, *Mapping at step level*, and *Aggregation at the step level*. Afterwards, for each path the best mapping with respect to the PTV is selected in phase *Mapping at the path level*. Finally, the PTVs corresponding to the mapping on path level are aggregated to obtain the certainty degree about schema coreference in the last step *Aggregation at the path level*. The above steps, except for the last step, concern the detection of coreferent elements in XML schemas, while the last step applies only to detecting the coreference of whole XML schemas.

The above mentioned steps will be now presented in detail.

2.3.1 Step 1: Model XML schemas as trees

The input XML Schema is modelled as a labelled unordered rooted tree [56] (as in Figure 2.2) obtained by the $getTree$ method in lines 1 and 2 of Algorithm 2.1. The original representation of the XML Schema (directed graph) in which *recursive definitions* (a leaf element refers to its ancestor) are represented by loops and *reference definitions* (simplifying schema by sharing of the common elements) are represented by cross edges is not appropriate in our approach for two reasons. First, recursive definitions result in an infinite number of paths. Second, graph matching is computationally expensive.

Algorithm 2.1 DETECTINGCOREFERENTXMLSCHEMAS

Require: Schema $\mathcal{R}_S$, Schema $\mathcal{R}_T$
Ensure: Propositions stating coreference of schemas p
1: $T_S \leftarrow$ getTree($\mathcal{R}_S$)
2: $T_T \leftarrow$ getTree($\mathcal{R}_T$)
3: $S \leftarrow$ extraction(T_S)
4: $T \leftarrow$ extraction(T_T)
5: $M_{paths} \leftarrow$ initialize(S,T)
6: **for all** $a^S \in S$ **do**
7: **for all** $a^T \in T$ **do**
8: $M_{paths_{a^S,a^T}} \leftarrow$ DetectCoreference($a^S.path$,$a^T.path$)
9: **end for**
10: **end for**
11: $Q_{paths} \leftarrow$ mappings(M_{paths})
12: $W_{Q_{paths}} \leftarrow$ calculateWeights(Q_{paths})
13: $\tilde{p} \leftarrow$ aggregate($\tilde{Q}_{paths}$,$W_{Q_{paths}}$)

The transformation is based on the top-down strategy (it starts from the root of the schema and ends at the leaves) and goes as follows. Each encountered element and attribute of the XML schema is translated into a *node* of the tree, preserving the relationships between the elements or attributes. The names of the elements or attributes (and other schema information, e.g., the cardinality) are the labels of the nodes.

However, reference and recursive definitions break the tree structure. Therefore, the former are transformed into a tree by duplicating the shared elements under the node that refers to it. Unfortunately, the latter, i.e., recursive definitions, cannot be solved as reference definitions because this generates an infinite loop which results in an infinite number of paths. Lu et al. [56] state that matching a recursively defined node is equivalent to matching the inner node that is being referred to. Therefore, if a node refers to its ancestor, then the connection is cut.

2.3.2 Step 2: Extraction

The algorithm extracts leaf nodes with metadata (paths to the root) from the input trees to be compared (lines 3 and 4 in Algorithm 2.1), cf., both trees in Figure 2.2. The path is constructed step by step by traversing the tree from the root to a leaf node. This is a sequence of names of traversed nodes which are separated by slashes.

For example, considering the right tree in Figure 2.2, the path "/cddb/disc/-tracks/title" is extracted for the node "title" of the tree on the right. Moreover, in our approach, only leaves are considered because leaves are the only nodes which are recommended as containing data.

Thus, two sets of paths are obtained. In our example (cf. Fig. 2.2), the set extracted from the schema tree on the left is denoted as S and set extracted from the schema tree on the right is denoted as T.

2.3.3 Step 3: Generation of a coreferent paths matrix

A coreferent paths matrix M_{paths} is generated based on the technique that was introduced in [79] (line 5 in Algorithm 2.1). For two input schemas with, respectively, m and n paths, M_{paths} is a $m \times n$ matrix, where element $M_{paths_{i,j}}$, $i = 1, \ldots, n$ and $j = 1, \ldots, m$ is a PTV resulting from the comparison of paths i and j from both schemas and reflecting the (un)certainty about their coreference. An example is given in Table 2.1. Each row corresponds to a path from S, whereas each column corresponds to a path from T. The matrix elements are computed using the following substeps (lines 6-10 in Algorithm 2.1).

Table 2.1 Example of a coreferent paths matrix. The bold values mean the best mappings for paths pair (their choice is discussed in Section 2.3.4).

Paths S / Paths T	/cddb/disc/genre	/cddb/disc/dtitle	/cddb/disc/year	/cddb/disc/tracks/title
/discs/disc/genres	**T:1.00** **F:0.06**	T:1.00 F:0.62	T:1.00 F:0.63	T:1.00 F:0.81
/discs/disc/title	T:1.00 F:0.65	**T:1.00** **F:0.12**	T:1.00 F:0.63	T:1.00 F:0.36
/discs/disc/year	T:1.00 F:0.63	T:1.00 F:0.63	**T:1.00** **F:0.06**	T:1.00 F:0.81
/discs/disc/tracklist/title	T:1.00 F:0.81	T:1.00 F:0.36	T:1.00 F:0.81	**T:1.00** **F:0.16**

2.3.3.1 Tokenisation of a path

In this step, each path is tokenised, which results in a list of substrings (elements on the path) which are called *steps* (lines 1 and 2 in Algorithm 2.2). Thus, *tokenisation of a path* transforms a path into a list of steps. In most cases, steps are separate words. In our approach, tokenisation of a path is equivalent to deleting all delimiting '/' characters in the paths. For instance, tokenisation of the path "/cddb/disc/tracks/title" results in the list $[cddb, disc, tracks, title]$. The reason why we introduce this tokenisation is that traditional string comparison methods (character based methods) are not efficient for long character strings [8]. A major advantage of tokenisation of a path is that it decreases complexity and enhances the effectiveness of string comparison in our approach. Moreover, as was described in Section 1.4.4, it allows us to introduce a specific aggregation method which helps to detect coreferent elements more effectively.

Remark: Compound versus simple steps. A special case of comparison concerns XML paths which consist of long or short steps (label length) describing the same real entity; for instance, a path which contains many short steps "/order/address/business/street/no" and a path which contains a few long steps "/order/addressBusiness/streetNo". The problem is how to detect strong matching of such paths. One solution can be tokenisation of the long steps (i.e., selecting single words). The change between lower and upper case, hyphen or underscore can be treated as a separator indicating the place to break a word.

Algorithm 2.2 DETECTINGCOREFERENTXMLPATHS

Require: Path $path_S$, Path $path_T$
Ensure: Propositions stating coreference of paths p
1: $path_S.steps \leftarrow$ tokenization($path_S$)
2: $path_T.steps \leftarrow$ tokenization($path_T$)
3: $M_{steps} \leftarrow$ initialize($path_S.steps$, $path_T.steps$)
4: **for all** $(step_S) \in path_S.steps$ **do**
5: **for all** $(step_T) \in path_T.steps$ **do**
6: $M_{steps_{S,T}} \leftarrow$ compare($step_S$, $step_T$)
7: **end for**
8: **end for**
9: $Q_{steps} \leftarrow$ mappings(M_{steps})
10: $W_{Q_{steps}} \leftarrow$ calculateWeights(Q_{steps})
11: $\tilde{p} \leftarrow$ aggregate($\tilde{Q}_{steps}$,$W_{Q_{steps}}$)

2.3.3.2 Steps comparison

The one-level string comparison technique proposed in [8] is used to compare the resulting steps from each pair of paths (line 6 in Algorithm 2.2). This low-level comparison method estimates the possibility that two given steps are coreferent and is based on an approximation of *weak string intersections* which is the set of longest common subsequences. It uses the concept of a moving window to construct the intersection of the two input steps. More specifically, the algorithm starts at the beginnings of both steps of a pair and moves a window over each of them. Each time common characters are detected under the moving windows they are added to the intersection, which is the largest set (in terms of set cardinality) that is a subset of both steps.

For example, consider a pair of steps (strings) s_1=*tracks* and s_2=*tracklist*. The construction of the intersection goes then as follows. We start with two one-character wide windows. Initially each window is at the beginning of a respective string and contains a character *'t'*. This character is common so it is added to the intersection and both windows move to their next position. Similarly for *'r'*, *'a'*, *'c'* and *'k'*. In the next step the windows contain different characters, *'s'* and *'l'*, respectively. Thus, the window size is increased by one. This is repeated until the windows contain a common character, here *'s'*, or there are no more characters in both of the strings. Next, the common character is added to the intersection, windows are shrunk to one character and moved to the position where the common character was found increased by 1. This construction of the intersection is repeated until the windows reach the ends of strings. Finally, the resulting intersection is *'tracks'*. The non common characters are counted (considered as errors) and decrease possibility that steps are coreferent.

This method marks out four different types of errors during comparison. These are *prefix*, *suffix*, *gap* and *mismatch*. The *prefix* is an error where one of the input strings contains a prefix before the matched substring, for instance a letter *'d'* is a prefix for ***d****title* and *title*. Analogously for *suffix*. The *gap* consists of missing characters in the middle of a string, for instance *'li'* is a gap and *'t'* is a suffix for strings *track****list*** and *tracks*. Finally, a *mismatch* is an umatched character in both strings. These errors have different importance and influence on the final matching result. Because of that, the importance of each error type is expressed by predefined weights between 0 and 1 and are problem dependent. The higher the weight of an error the lower degree of matching of two strings for which such an error occurs. In our case, where abbrevations are very popular, the crucial error types are *prefix* and *mismatch* so their weights are set to 1, *gap* has a weight 0.3 and *suffix* 0.1 is the minor error.

Our algorithm then compares pairs of steps from one input list with steps from the other input list. It generates PTVs which express the uncertainty about the coreference of the compared steps as described above. The possibility that a proposition p, stating that two steps are coreferent, is true ($\mu_{\tilde{p}}(T)$) and the possibility that p is false ($\mu_{\tilde{p}}(F)$) are calculated by the following equations:

$$\mu_{\tilde{p}}(T) = \frac{possT}{factor} \tag{2.1}$$

$$\mu_{\tilde{p}}(F) = \frac{possF}{factor} \tag{2.2}$$

where *possT*, *possF* and *factor* equal:

$$possT = \frac{|intersection|}{\max(s_1.length, s_2.length)} \tag{2.3}$$

$$possF = \sum_{i=0}^{|errors|} (errors_i.size \times w_i) \tag{2.4}$$

$$factor = \max(possT, possF) \tag{2.5}$$

where $|intersection|$ denotes the number of common chatacters, an $|errors|$ is the number of types of the errors, $errors_i.size$ is the number of the errors of a given type.

On the one hand, $possT$ is the ratio between the number of characters that are found to be common for a pair of steps (cardinality of the intersection) and the length of the longer step. On the other hand, $possF$ is computed as the sum of the product of the number of the errors of a given type (from errors that are found during comparison) and predefined weight w_i of specific error type. Finally, $factor$ is the maximum of $possT$ and $possF$ and is used to normalize both possibilities.

The PTVs resulting from the comparisons of all steps in the two paths are represented in a so-called *coreferent steps matrix* (lines 3-8 in Algorithm 2.2). An example of such a matrix is given in Table 2.2.

Our step comparison method thus takes into account misspellings and abbreviations and, moreover, has a low computational complexity. This is a great advantage for XML coreference detection because abbreviations are frequently used in XML paths.

This technique was chosen due to its efficiency [8]. In the literature a multitude of algorithms for string comparison has been proposed and these may also be employed here. An example of an interesting survey concerning strings in general is [2]. Work focused on coreference (duplicates) detection in the context of XML is [80]. An example of an approach employing fuzzy logic which might also be of interest to the reader is [81].

Table 2.2 Example of a coreferent steps matrix. Bold values represent the best mappings for the steps from both multisets (cf. step 3 substep 3 of the algorithm).

Steps	**cddb**	**disc**	**tracks**	**title**
discs	**T: 0.0** **F: 1.0**	T: 1.0 F: 0.01	T: 0.0 F: 1.0	T: 0.0 F: 1.0
disc	T: 0.0 F: 1.0	**T: 1.0** **F: 0.0**	T: 0.0 F: 1.0	T: 0.0 F: 1.0
tracklist	T: 0.0 F: 1.0	T: 0.0 F: 1.0	**T: 1.0** **F: 0.17**	T: 0.0 F: 1.0
title	T: 0.0 F: 1.0	T: 0.0 F: 1.0	T: 0.0 F: 1.0	**T: 1.0** **F: 0.0**

2.3.3.3 Mapping at the step level

This substep selects the best mapping between steps belonging to lists representing two paths in the coreferent steps matrix M_{steps} (cf. Table 2.2; the best mappings are marked in bold, line 9 in Algorithm 2.2). A general form of such a mapping problem is well known as the *assignment problem* [82] and in this case we apply the mapping algorithm proposed in [79] which works as follows. Let us assume a set S_S which contains steps of path $path_S$ and a set S_T which contains steps of path $path_T$ with $|S_S| \leq |S_T|$. The key idea is to create an injective mapping Q_{steps} from S_S to S_T based on PTVs expressing the certainty of matching between steps. Having the coreferent steps matrix M_{steps}, the largest PTVs are found iteratively, their locations in M_{steps} are added to the mapping and then the rows and columns of the locations are removed. The step mapping is conducted by Algorithm 2.3 which works as follows.

The mapping algorithm [79] first selects the largest PTV for each row of the

matrix M_{steps} (largest in the sense of the order relation (1.17)), as in step (a) in Figure 2.5, where the largest PTVs are bold; if the largest PTV for a row is not unique then a PTV of steps on the same (or similar) position should be chosen; if that condition does not help then one of the largest values is chosen randomly (lines 1-3 in Alg. 2.3). The functions r and c provide mappings of elements from S_S and S_T, respectively, to row and column indexes of these elements.

Next, the algorithm checks if **conflicts** occur (lines 4-15 in Alg. 2.3), i.e., if two (or more) rows exist, say r_1 and r_2, with the largest PTV in the same column. These conflicts are resolved one by one, for any pair of rows sharing the same column with maximal PTV. For resolving column conflicts the PTVs of rows preliminarily mapped to it, i.e., having their largest PTVs in it, are sorted in a non-increasing order and the conflict resolution starts with rows with largest PTVs. If that condition does not help then, by convention, in this case we choose first rows with lower indexes. Resolving each conflict consists in choosing one row which will be mapped to a given column and then the whole procedure of choosing a column to resolve conflicts is started from the beginning (the already mapped rows and columns are ignored).

More specifically, the conflict resolution works as follow. On the one hand, if the maximal PTV in such a column is unique, then the this column is mapped to the row containing this maximal PTV, say r_1, and the mapped column and row are

Algorithm 2.3 STEPMAPPING

Require: $(|S_S| \times |S_T|)$ matrix M_{steps}
Ensure: Injective mapping Q_{steps}
1: **for all** $s \in S_S$ **do**
2: $m[r(s)] \leftarrow \arg\max_{t \in S_T} M[r(s), c(t)]$
3: **end for**
4: **while** $\exists x \neq y \wedge m[r(x)] = m[r(y)]$ **do**
5: $d \leftarrow 0$
6: $\tilde{p}_1 \leftarrow M[r(x)][c(m[r(x)])]$
7: $\tilde{p}_2 \leftarrow M[r(y)][c(m[r(y)])]$
8: **if** $\tilde{p}_1 = \tilde{p}_2$ **then**
9: $d \leftarrow$ choose$(M[r(x)], M[r(y)])$
10: **end if**
11: **if** $\tilde{p}_1 < \tilde{p}_2 \vee d = r(x)$ **then**
12: $m[r(x)] \leftarrow$ search$(M[r(x)])$
13: **else**
14: $m[r(y)] \leftarrow$ search$(M[r(y)])$
15: **end if**
16: **end while**
17: $\forall s \in S_S : Q_{steps} = c^{-1}(m[r(s)])$

Figure 2.5 Example of objects mapping

$$a)\begin{bmatrix}(\mathbf{1,0.05}) & (1,0.8) & (0.3,1)\\ (\mathbf{1,0.2}) & (1,0.3) & (1,0.4)\\ (\mathbf{1,0.2}) & (1,0.3) & (1,0.8)\end{bmatrix}\ b)\begin{bmatrix}(\mathbf{1,0.05}) & (1,0.8) & (0.3,1)\\ & (\mathbf{1,0.3}) & (1,0.4)\\ & (\mathbf{1,0.3}) & (1,0.8)\end{bmatrix}$$

$$c)\begin{bmatrix}(\mathbf{1,0.05}) & (1,0.8) & (0.3,1)\\ & & (\mathbf{1,0.4})\\ & (\mathbf{1,0.3}) & (1,0.8)\end{bmatrix}$$

ignored for further mapping. Then, a subprocedure **search** is executed (lines 10-14 in Alg. 2.3) to find another column with largest PTV for the remained conflicted rows, in this case the row r_2. In our example from step (a) to (b) in Figure 2.5 there exist conflicts between all rows, because for each of the rows the maximum has been selected for the same column, i.e., column 1. Row 1 contains the largest PTV, so this conflict is resolved and row 2 and 3 are relocated.

On the other hand, if the PTVs are equal for two or more rows in a given column then the subprocedure **choose** is executed (lines 7-9 in Alg. 2.3). It decides which row should be passed to subprocedure **search**, where the new maximum will be selected (lines 10-14). If there are more than two rows with equal largest PTVs in the given column then by convention the subprocedure **choose** is executed for each pair of rows starting with rows with lower index. The subprocedure **choose** selects the remaining PTVs for each conflicting row from columns which have not yet been mapped to some rows, referred to as *enabled positions*, which results in multisets of PTVs, say M_1 and M_2. The three cases are considered. If $M_1 = M_2$, then both rows contain the same PTVs on enabled positions. By convention, in this case we choose the first row. If $M_1 \subset M_2$ or $M_2 \subset M_1$, i.e., M_2 contains the same PTVs as M_1 on enabled positions and some others PTVs or vice versa (see Section 1.4.2), then obviously the row corresponding to the largest multiset is chosen because it contains all information captured by the smaller multiset. If neither of these cases yield, we subtract $M_1 \cap M_2$ from both multisets and the multiset containing the largest PTV after subtraction is chosen. This way the largest possible PTVs are left for future maximum selection. Just as in our example from step (b) to (c) in Figure 2.5, a conflict occurs between row 2 and 3 - the selected PTVs are equal, so multisets $M_1 = \{(1, 0.4)\}$ and $M_2 = \{(1, 0.8)\}$. The last case is applied in our example and the largest PTV is selected because $(1, 0.4) \geq (1, 0.8)$.

Finally, no conflicts occur and the algorithm stops, like in step (c) in Figure 2.5.

Moreover, the example of mapping at the step level is illustrated with the coreferent steps matrix shown in Table 2.2. First, for each row the maximum is selected (the row is the first element in a pair, the column is the second element): (discs,disc), (disc,disc), (tracklist,tracks), (title,title). For rows ‘discs’ and ‘disc’, a conflict occurs as both of them best match column ‘disc’. However, row ‘disc’ contains a larger PTV as $(1, 0) \geq (1, 0.01)$, so column ‘cddb’ is chosen as match-

ing row 'discs' (column 'cddb' is the only one remaining unselected). Finally, no conflicts occur and the algorithm stops. The selected PTVs, indicating matching pairs of rows and columns, are shown in bold, e.g., the steps "tracklist" and "tracks" are chosen as matching and the PTV expressing their matching degree equals $(1.0, 0.17)$.

2.3.3.4 Aggregation at the step level

The last substep in generating coreferent paths matrix is aggregation of the PTVs in the coreferent steps matrix, hereby using the mappings that were obtained in the previous substep (line 11 in Algorithm 2.2). The result of this aggregation is a PTV for each pair of paths which expresses the uncertainty about the coreference of these two paths.

For the aggregation at the step level, the following issues should be dealt with.

- On the one hand, consider path "/cddb/disc/tracks/title" from the right schema and path "/discs/disc/tracklist/title" from the left schema of Figure 2.2, where not all steps are matched at the beginning of the paths but all are at the end match. In such a case both paths have a different context but may describe the same object as illustrated in our example. This means that if the steps at the end of the paths are coreferent, then this is a strong hint that the paths are coreferent.

- On the other hand, consider two paths where not all steps are matched at the end of the paths; cf. paths "/cddb/disc/publisher/telephone" and "/discs/disc-/publisher/address". Then, these elements are probably related but not similar and should not be matched.

- In contrast to the previous case, if one path, $path_T$, forms more or less a prefix of another path, $path_S$, i.e., most of the steps of $path_T$ (especially at the end) have matching steps at the beginning of $path_S$, then this supports the coreference of these paths. Indeed, $path_T$ may correspond to a concept which is a generalisation of the concept represented by $path_S$. The degree of certainty of coreference should be high but lower than the degree of certainty of coreference in case of an exact matching of the paths. For instance, consider as $path_T$ the path "/cddb/disc/artist" from the right schema and as path $path_S$ the path "/discs/disc/artists/name" from the left schema of Figure 2.2. Here the mismatch between the compared elements arises due to a possibly different *granularity* of information representation. In one schema the element "artist" represents all of the information, while in the second schema information about an artist is structured using the nested elements "name", "role", and possibly more.

Thus, it should be clear from these examples that not all steps in a path are equally important with respect to coreference detection (the importance of each step is expressed by a weight which is calculated in line 10 of Algorithm 2.2). Hence, in general, the following heuristic rule can be considered: two paths are more likely to be coreferent if they have more similar steps at the end. Because of this the aggregation takes into account the PTVs resulting from the coreference detection of the steps and the position of the steps in their respective paths. This rule is implemented by using the Sugeno integral for possibilistic truth values (1.27).

The aggregation operator for the comparison of two paths, S and T, is defined by the Sugeno integral for PTVs, where:

- $P = \{p_i\}$ is a set of propositions stating coreference of pairs of steps identified in the previous substep of the algorithm, i.e., 'mapping at the step level',
- $\tilde{P}$ is the set of PTVs corresponding to the above-mentioned propositions representing the uncertainty about their truth values computed as discussed in the previous substep and represented in a coreferent steps matrix,
- the fuzzy measure γ^T is defined by:

$$\gamma^T(Q) = \sum_{j=1}^{k} w_j, \qquad Q \subseteq P, Q = \{p_1, \ldots, p_k\} \tag{2.6}$$

where w_j is the weight of the jth pair (s_S^j,s_T^j) of steps, computed by:

$$w_j = \frac{r_j}{\sum_{i=1}^{k} r_i} \tag{2.7}$$

Hereby, r_j is in turn defined by:

$$r_j = \frac{1}{2} \times \left(\left(\frac{pos(s_S^j)}{len(p_S)} \right)^{exp(p_S, j)} + \left(\frac{pos(s_T^j)}{len(p_T)} \right)^{exp(p_T, j)} \right) \tag{2.8}$$

where step s_S^j belongs to path p_S, $pos(s)$ is a function which determines the position of step s in its path, i.e.: $pos(s) = 1$ for the first step in the path, $pos(s) = len(p_S)$ for the last step in the path, and $len(p_S)$ is the length of path p_S (the number of steps in this path) and

$exp(p_S, j) = len(p_S) - pos(s_S^j)$.

The exponentiation employed in (2.8) helps to spread the weights. More precisely, thanks to it the distribution of weights follows the power law and weights for steps at the beginning of a path are lower than weights for steps

at the end. This helps to implement the assumption that two paths are more likely to be coreferent if they have more similar steps at the end.

Moreover, a special procedure has to be applied in cases where the compared paths contain a different number of steps. For each unmatched step the algorithm adds a PTV (0,1) and its coresponding weight. This weight depends on the position of the unmatched step in the path. It means, if $len(p_S) \neq len(p_T)$ then for each step s^j belongs to path p_S (p_T) which is not matched to any step that belongs to path p_T (p_S, respectively) are assigned a PTV (0,1) and weight w. The weight w is calculated according to Equation (2.7) where r_j is calculated as follows:

$$r_j = \frac{1}{2} \times \left(\frac{pos(s^j)}{len(p)}\right)^{exp(p,j)} \tag{2.9}$$

where p is a path p_S or p_T, $pos(s^j)$ is a function which determines the position of step s^j in its path and $len(p)$ is the length of path p.

For instance, assume two paths, "/cddb/disc/title" and "/discs/disc/tracklist/-title". The unmatched step is "tracklist" and its weight equals:

$$r = \frac{1}{2} \times \left(\frac{3}{4}\right)^{(4-3)} = \frac{1}{2} \times 0.75 = 0.375 \tag{2.10}$$

$$w = \frac{0.375}{2.016} = 0.19. \tag{2.11}$$

- the fuzzy measure γ^F is defined by

$$\gamma^F(Q) = \begin{cases} 1 & \text{if } Q = P \\ 0 & \text{otherwise} \end{cases} \tag{2.12}$$

what is implied by condition (1.26) is that for each Q which is a subset of P (except $Q = \emptyset$), the fuzzy measure γ^T of Q ($\gamma^T(Q)$) is always greater than 0 because weights (on which γ^T depends) are greater than 0 (cf. Equations (2.7)-(2.8)). Therefore, the fuzzy measure γ^F of the complement of the set Q ($\gamma^F(\bar{Q})$) must be equal to 0 (except when $Q = P$) to satisfy condition (1.26).

Example. The aggregation at the step level for the paths "/cddb/disc/tracks/title" and "/discs/disc/tracklist/title" (cf. Table 2.2) goes as follows: first, the weights for the pairs of steps are computed using (2.7)-(2.8):

$$\begin{array}{lll} r(cddb, discs) & = \left(\frac{1}{4}\right)^{(4-1)} & = 0.0156 \\ r(disc, disc) & = \left(\frac{2}{4}\right)^{(4-2)} & = 0.25 \\ r(tracks, tracklist) & = \left(\frac{3}{4}\right)^{(4-3)} & = 0.75 \\ r(title, title) & = \left(\frac{4}{4}\right)^{(4-4)} & = 1 \end{array} \tag{2.13}$$

$$\begin{array}{lll} w(cddb, discs) & = 0.0156/2.0156 & = 0.008 \\ w(disc, disc) & = 0.2500/2.0156 & = 0.124 \\ w(tracks, tracklist) & = 0.7500/2.0156 & = 0.372 \\ w(title, title) & = 1.0000/2.0156 & = 0.496 \end{array} \tag{2.14}$$

These weights are then used to aggregate the PTVs from Table 2.2. Table 2.3 shows the calculations for $Nec_{\tilde{p}}(T)$. The rows are sorted in decreasing order by ranking the PTVs expressing the coreference between matched pairs of steps (cf. Definition 4). The first column (not counting the column with the heading i) shows matched steps. The second column shows the PTVs expressing their coreference, while column 4 gives the value for $Nec(\tilde{p}_{(i)^T} = T)$. For example, $Nec(\tilde{p}_{(i)^T} = T)$ in row 3 is computed by $Nec(\tilde{p}_{(i)^T} = T) = 1 - Pos_{\tilde{p}}(F) = 1 - 0.16 = 0.84$. The third column shows the weights. The fuzzy measure γ^T of the subsequent sets $P_{(i)^T}$ is shown in column 5 and is calculated by Equation (2.6) e.g., for the step pair $(tracks, tracklist)$ we obtain $\gamma^T(P_{(3)^T}) = w_1 + w_2 + w_3 = 0.12 + 0.5 + 0.37 = 0.99$. The last column shows the minimum (denoted by $\wedge$ in (1.29)) of $\gamma^T(P_{(i)^T})$ and $Nec_{\tilde{p}}(T)$, e.g., for $(tracks, tracklist)$ we obtain 0.84. Finally, $Nec_{\tilde{p}}(T)$ for paths "/cddb/disc/tracks/title" and "/discs/disc/tracklist/title" is the maximum value (denoted by $\vee$ in (1.29)) of the last column of Tables 2.3 (cf. Equation (1.27)) and is equal to 0.84 (the bold value in Table 2.3).

Table 2.3 $Nec_{\tilde{p}}(T)$ for paths "/cddb/disc/tracks/title" and "/discs/disc/tracklist/title".

i	**Step**	**PTV**	w_i	$Nec(\tilde{p}_{(i)^T})$	$\gamma^T\left(P_{(i)^T}\right)$	$\wedge$
1	disc disc	T: 1.00 F: **0.00**	0.12	1	0.12	0.12
2	title title	T: 1.00 F: **0.00**	0.5	1	0.62	0.62
3	tracks tracklist	T: 1.00 F: **0.16**	0.37	0.84	0.99	**0.84**
4	cddb discs	T: 0.00 F: **1.00**	0.01	0	1	0

$Nec_{\tilde{p}}(F)$ is calculated analogously to $Nec_{\tilde{p}}(T)$. This time the rows are sorted in increasing order by ranking the PTVs as shown in Table 2.4 (cf. Definition 4). $Nec(\tilde{p}_{(i)^F} = F)$ corresponding to a PTV representing the matching degree of two steps is shown in column 4; e.g., for the pair of steps (tracks, tracklist) in row 2: $Nec(\tilde{p}_{(i)^F} = F) = 1 - Pos_{\tilde{p}}(T) = 1 - 1 = 0$. The fuzzy measure γ^F is equal to 0 for all sets except for the whole set P, corresponding to the last row of the table. The last column shows the minimum (denoted by $\wedge$ in (1.28)) of $\gamma^F(P_{(i)^F})$ and $Nec(\tilde{p}_{(i)^F} = F)$, which equals 0 for all of the matched step pairs. Finally, the $Nec_{\tilde{p}}(F)$ for paths "/cddb/disc/tracks/title" and "/discs/disc/tracklist/title" is the

maximum of the values (denoted by $\vee$ in (1.28)) of the last column of Tables 2.4 (cf. Equation (1.27)) and is equal to 0 (the bold value in Table 2.4). Hence, the coreference between the paths

"/cddb/disc/tracks/title" and "/discs/disc/tracklist/title"

is expressed by the PTV $\tilde{p}$ with $\mu_{\tilde{p}}(T) = 1-0 = 1$ and $\mu_{\tilde{p}}(F) = 1-0.84 = 0.16$, where 0 and 0.84 are, respectively, the maximum value ($\vee$) of the last columns of Tables 2.4 and 2.3 (cf. Equation (1.27); respectively, $Nec_{\tilde{p}}(F)$ and $Nec_{\tilde{p}}(T)$).

Table 2.4 $Nec_{\tilde{p}}(F)$ for paths "/cddb/disc/tracks/title" and "/discs/disc/tracklist/-title".

i	**Step**	**PTV**	w_i	$Nec(\tilde{p}_{(i)^F})$	$\gamma^F(P_{(i)^F})$	$\wedge$
1	cddb discs	T: **0.00** F: 1.00	0.01	1	0	**0**
2	tracks tracklist	T: **1.00** F: 0.16	0.37	0	0	0
3	disc disc	T: **1.00** F: 0.00	0.12	0	0	0
4	title title	T: **1.00** F: 0.00	0.5	0	1	0

2.3.4 Step 4: Mapping algorithm at the path level

Once again, our algorithm establishes a mapping, but this time between paths (line 11 in Algorithm 2.1). The mappings of paths determined here with PTVs are inputs for the next step, where PTVs are aggregated to a single PTV which represents the (un)certainty of the coreference of the whole XML schemas.

In this step the best mapping between the paths of two XML schemas is determined, based on the coreferent paths matrix which is generated in the previous step and whose example is shown in Table 2.1. The procedure is analogous to the *'Mapping at the steps level'* as described above. Indeed, a coreferent paths matrix also consists of PTVs, of which the largest in each row is selected, thus, handling conflicts as described in Step 3, Substep 3. For example, the PTVs of the selected matched paths are set in bold in Table 2.1.

2.3.5 Step 5: Aggregation at the path level

Finally, the coreference of the whole XML schemas can be computed (line 13 in Algorithm 2.1). To this aim the aggregation of PTVs expressing the paths' coreference is done using a technique based on the Sugeno integral for PTVs [37], in a similar way as was proposed for the *'Aggregation at the step level'* as described

in Step 3, Substep 4. As was argued earlier, this approach makes it possible to adequately cope with the different importances of aggregated elements, i.e., paths in this case (it is calculated in line 12 in Algorithm 2.1). In [83] it is stated that not all attributes (elements, paths, generally metadata) are equally important and not all of them have the same role in coreference discovery. For instance, it can be assumed that a path "/cddb/disc/title" of the node "title" is more important than a path "/cddb/disc/id" of the node "id", but this is, of course, context-dependent. Because of this, mapped elements are classified into subsets from the most to the least important. This classification can be done manually or be based on heuristics or knowledge stored in a knowledge base. Thus, PTVs expressing coreference of all matched pairs of paths can be aggregated using the Sugeno integral, as defined in Definition 4. The resulting aggregated PTV then expresses the (un)certainty about the coreference between the two input XML schemas, such as the left and right schemas represented in Figure 2.2.

In the next paragraph, heuristics are presented to compute the importance of each XML leaf element (path). They are based on expert opinions that the most important information is unique and required, and also given closer to the root. This allows to increase the quality of the results of the proposed approach.

2.3.5.1 Heuristics

In this paragraph four novel heuristics are proposed which help to determine the importance of the elements (paths, objects). They are using the following criteria: *element requirement*, *element uniqueness*, *r-distance descendant*, *k-closest descendant*. Each heuristic considers different metadata (schema level information) but returns a normalised *score* which is expressed by a value in the unit interval [0,1] (see the example below of the heuristics: (2.17)-(2.20)). A higher score specifies higher importance of an element. Afterwards, for each element the scores are summed up to the $finalScore$:

$$\forall a^S \in S, finalScore(a^S) = \frac{sumScores(a^S)}{\max_{i \in S}(sumScores(a^i))} \tag{2.15}$$

where $sumScores$ equals:

$$sumScores(a^S) = \sum_{k \in H} getScore_k(a^S) \tag{2.16}$$

where a^S with a $path$ is a leaf element from a set S (T, respectively), H is a set of all heuristics, $getScore_k(a^S)$ represents the score of a^S obtained using the k-th heuristics from H, a set of considered heuristics, and returns a value between 0 and 1. The *finalScore* is normalised to a value between 0 and 1 by dividing it by the maximum of all *sumScores* values of elements from S (T, respectively).

The final score is used to classify each element with respect to its importance. This importance is expressed by a *weight* which is a normalised value between 0 and 1; a higher weight means that the element is more important. Moreover, the weight (importance) of a pair of mapped elements equals the minimum of the importance weights of these elements.

The heuristics under consideration are the following:

- *Element requirement* is based on the minimum cardinality constraint (called *minOccurs* for short), which can be defined in a schema and takes a value that is greater than or equal to 0. The value 0 means that the element is optional, the value 1 means that the element is required and has to occur minimum one time, and so on. With this heuristic elements with minOccurs values equal 0 are the least important. So, this heuristic prefers elements which are neither optional nor repeatable and returns 0 if an element is optional, 1 if it is required or not repeated, and $1/minOccurs$ in other cases:

$$\forall a^S \in S, req(a^S) = \begin{cases} 0, & \text{if } minOccurs \text{ is } 0 \\ minOccurs^{-1}, & \text{else.} \end{cases} \tag{2.17}$$

- *Element uniqueness* is based on the maximum cardinality constraint (called *maxOccurs* for short) which can be defined in the schemas and takes a value that is greater than or equal to 0. This value specifies the maximum number of occurrences for the element. An element is least important if the maxOccurs value equals *unbounded* (∞), which means that there is no limit on the maximum number of its occurrences. On the other hand, the element is the most important if its maxOccurs value equals 1. This heuristic prefers elements which are not repeated, so it returns 0 if an element has no maximum number of occurrences and $1/maxOccurs$ in other cases:

$$\forall a^S \in S, unique(a^S) = \begin{cases} 0, & \text{if } maxOccurs \\ & \text{is } \infty \text{ or } 0 \\ maxOccurs^{-1}, & \text{else.} \end{cases} \tag{2.18}$$

Remark. The values of these indicators are inherited from their ancestors. An inherited minOccurs (maxOccurs) equals the product of all minOccurs (maxOccurs) of the current element and its ancestors. For instance, in Table 2.5, the maxOccurs defined in the schema of the element "title" with the path "/discs/disc/tracklist/title" equals 1, but the maxOccurs of the parent element "tracklist" with the path "/discs/disc/tracklist" can be more than 1 (i.e. unbounded), so the inherited (real) maximum occurrence of that element is then unbounded; minOccurs of the element "title" with the path

"/discs/disc/videos/title" equals 1, but the minOccurs of the parent element "videos" with the path "/discs/disc/videos" equals 0, so the real minimum occurrence of that element then equals 0.

Table 2.5 Example of minimum and maximum occurrence indicators of elements which are defined in XML Schema and their paths from the left schema S of Figure 2.2.

Element	Path	Min Occurs	Max Occurs
discs	/discs	1	1
disc	/discs/disc	1	∞
tracklist	/discs/disc/tracklist	1	∞
title	/discs/disc/tracklist/title	1	1
videos	/discs/disc/videos	0	∞
title	/discs/disc/videos/title	1	1

- *R-distance descendant* is based on the number of steps in a path. It reflects that leaves which are close to the root (with short paths) are the most important. This heuristic score is calculated by the following equation:

$$\forall a^S \in S, rDist(a^S) = 1 + \frac{1 - depth(a^S.path)}{DepthMax} \tag{2.19}$$

where $DepthMax$ is the maximum depth in the tree and $depth$ returns the depth of a path (number of steps in the path). For instance, Table 2.6 shows the depths of a few elements (leaves) and their paths from the left tree A of Figure 2.2.

Table 2.6 Example of depth (*depth(p)*) and position (*pos(p)*) of elements and their paths from the left schema S of Figure 2.2.

Element	Path *p*	*depth(p)*	*pos(p)*
styles	/discs/disc/styles	3	1
genres	/discs/disc/genres	3	2
title	/discs/disc/videos/title	4	3
title	/discs/disc/title	3	4
mainrelease	/discs/disc/mainrelease	3	5

- *K-closest descendant* is based on the position of an element (precisely a leaf) in the tree. It is based on an assumption that the order of elements is important, i.e., more important elements first occur in a schema. The score of this heuristic is calculated by the equation:

$$\forall a^S \in S, kClosest(a^S) = 1 + \frac{1 - pos(a^S.path)}{|S|} \tag{2.20}$$

where $|S|$ is the number of elements (leaves) in the XML tree S; $pos(path)$ is the position of the leaf from this tree which is pointed by a path and the numbering is based on the depth-first search.

For instance, Table 2.6 contains the position of a few elements (leaves) and their paths from the left tree S of Figure 2.2.

Remark. It should be noted that there is a crucial difference between the importance of steps (tags) in a path (Equation (2.8)) and the importance of elements (leaves) in the schema (Equation (2.20)). In the first case the importance of steps (tags) in a path is considered in the context of a single element. On the other hand, in the latter case the order of the element is considered in the context of the whole schema.

Example. Calculating the *finalScore* of a leaf element a_1^S "title" with $path_S$ "/discs/disc/title" goes as follows. First, the importance of the element with the given path is computed using heuristics (2.17)-(2.20):

$$\begin{aligned} req(a_1^S) &= \tfrac{1}{1} = 1, \text{ because inherited minOccurs = 1} \\ unique(a_1^S) &= 0, \text{ because inherited maxOccurs =}\infty \\ rDist(a_1^S) &= 1 + \tfrac{1-3}{4} = 0.5 \\ kClosest(a_1^S) &= 1 + \tfrac{1-4}{11} = 0.73. \end{aligned} \tag{2.21}$$

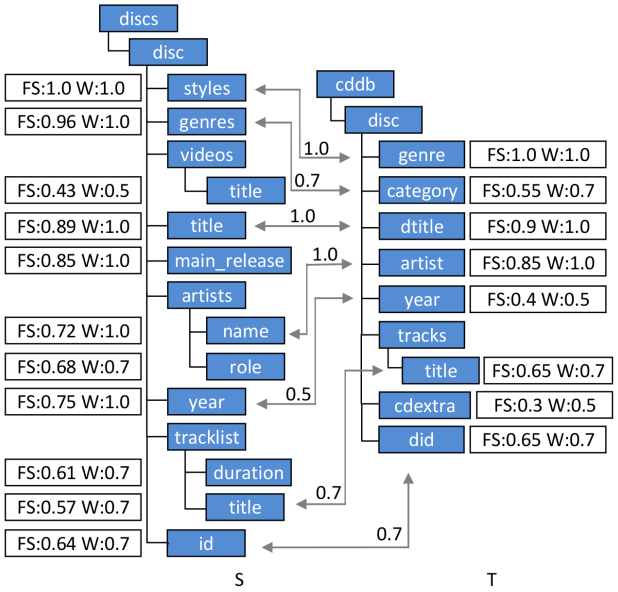
The normalised final score is calculated by Equation (2.15) and equals 0.89, because the maximum importance for all considered leaf elements (paths) equals 2.5.

In contrast, the *finalScore* of a leaf element a_2^S "title" with a $path_S$ "/discs/disc/tracklist/title" goes as follows. First, the importance of the element with the path is computed:

$$\begin{aligned} req(a_2^S) &= \tfrac{1}{1} = 1, \text{ b/c inherited minOccurs = 1} \\ unique(a_2^S) &= 0, \text{ b/c inherited maxOccurs = } \infty \\ rDist(a_2^S) &= 1 + \tfrac{1-4}{4} = 0.25 \\ kClosest(a_2^S) &= 1 + \tfrac{1-10}{11} = 0.18. \end{aligned} \tag{2.22}$$

Hence, the normalised final score equals 0.57. The resulting weight properly reflects the expert opinion that the "title" of a compact disc is more important than

Figure 2.6 Real-world data example: XML schema elements with hierarchies extracted from Discogs (left tree - S) and FreeDB (right tree - T) datasets with final scores (FS), subset weights (W) and weights of mapped elements (shown on arrows).



the "title" of a single track. Figure 2.6 contains two real-world schema trees with final scores (abbr. *FS*) calculated for each leaf element.

Moreover, all leaf elements (based on final scores) are classified to the predefined subsets (with the predefined weights), ranking these elements from most to least important. We can assume three importance subsets: *the most important* with a weight of 1.0 which contains elements with final scores $\in$]0.7,1], *average important* with a weight 0.7 which contains elements with final scores $\in$]0.5,0.7], and *the least important* with a weight of 0.5 which contains elements with final scores $\in$ [0,0.5]. The weights expressing the importance (abbrev. *W*) of each element are presented in Figure 2.6. For instance, the element "year" from S with a final score of 0.75 is classified into the most important subset with a weight of 1.0, but a coreferent element from T with a final score of 0.4 is classified into the least important subset with a weight of 0.5. Moreover, the final weights (which express the importance) of the mapped pairs of elements are shown on the mapping arrow and are equal to the minimum of weights of the mapped elements, i.e. it equals 0.5 for the considered elements.

2.3.6 Algorithm features

The algorithm creates mappings between XML leaf elements. One element (which is represented by a path) from the first schema is matched to at most one element (a path) in the second schema. Moreover, the method to detect coreferent XML paths applies a *bottom-up* strategy. It checks all combinations of fine-grained elements and finds a match even if elements at a higher level are not similar. Because of this, no coreferent elements get lost. This strategy is more expensive than the *top-down* strategy, where the matching of parent element restricts the choices for all descendant elements. However, the top-down solution might mislead matchers when higher structures are different, but lower ones are quite similar.

2.4 Evaluation and discussion

Our system is compared with the popular schema matching systems COMA 3.0 [18, 84], Cupid [17], QMatch [76], HMAT [75], a method by Lu et al. [56] (referred to in what follows also as MatchingLu), and also with a two-level string matcher [8] (referred to in what follows also as StringMatcher II). The datasets used in this comparison are described in what follows.

Figure 2.7 Purchase order schemas pair 1 (PO1).

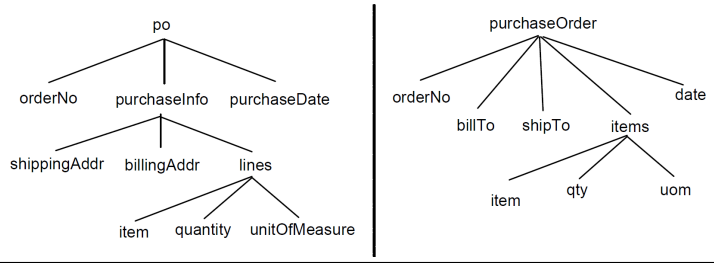


Figure 2.8 Purchase order schemas pair 2 (PO2).

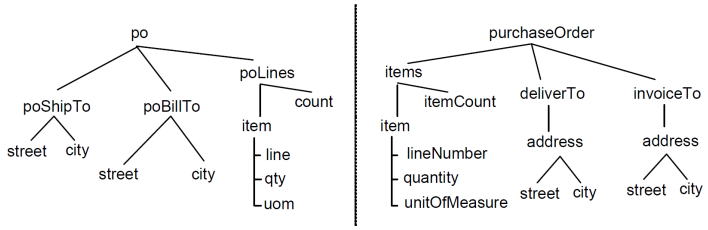


Figure 2.9 Purchase order schemas pair 3 (PO3).

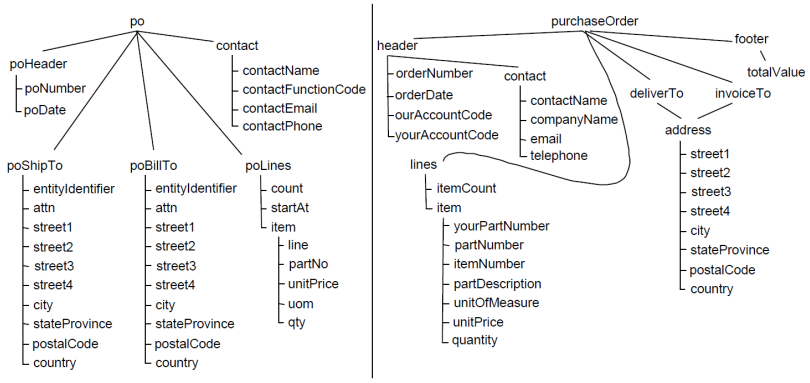
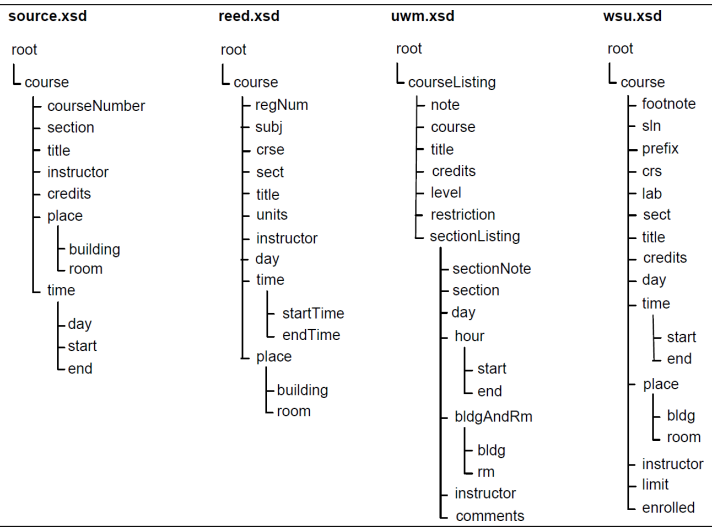


Figure 2.10 University courses schemas.



2.4.1 Datasets

To illustrate the proposed approach we consider three different real-world datasets, respectively, containing information about ‘compact discs’, ‘purchase order’, ‘university courses’ and a synthetic dataset which is generated for the purpose of execution time evaluation.

Compact disc data are represented by two schemas (also being used in our running example). The first schema is extracted from FreeDB and consists of 8 leaf elements. The second schema is extracted from Discogs and consists of 33 leaf elements, of which 7 have been identified manually as being coreferent with

the FreeDB elements. The real coreferent elements are presented in Table 2.10.

The purchase order data comprise three schema pairs that were used in evaluating other schema matching approaches [17,76]. Each pair has a different level of complexity. The first pair shown in Figure 2.7 is the simplest one, where the difference mostly consists in the tags' abbreviations and the structure. Each schema consists of 7 leaf elements, all of which have been identified manually as being coreferent. The second pair shown in Figure 2.8 differs (besides the tags' abbreviations and structure) in the usage of synonyms. Each schema consists of 8 leaf elements, all of which have also been identified manually as being coreferent. Finally, the schemas of the last pair shown in Figure 2.9 are similar to the second pair, but the number of leaf elements is larger. Each schema of the last pair consists of 33 leaf elements, of which 28 are coreferent. The actual coreferent elements of the first and second purchase order pairs are presented in Table 2.10. The coreference of the last pair is not presented due to limited space and similarity to the truly real coreferent elements of the second pair.

The university course data comprise four schemas, one schema (called *source*) is from [76] and three other are from the AnHai Doan repository[4], which are derived from the websites of three universities: Reed College (Reed), University of Wisconsin-Milwaukee (UWM) and Washington State University (WSU). These schemas are presented in Figure 2.10 and are considered in pairs: (Source, WSU) contains 10 pairs of coreferent elements, (Source, UWM) 11 and (Source, Reed) 10. Truly coreferent elements were manually identified by experts based on the schemas and instance data and are presented in Table 2.9.

Table 2.7 contains a summary of these datasets. The number of elements in the data vary between 9 and 43, while the number of leaves vary between 7 and 33. The maximum depth for these datasets equals 5, while the average depth does not exceed 4.

The synthetic data comprise six pairs of schemas which are generated by multiplexing of purchase order pair 3 (PO3, PurOrd3) shown in Figure 2.9 and replacing the elements' names by random words of length 3 through 10. For instance, the pair (PO3-2, PurOrd3-2) consists of two copies of the pair (PO3, PurOrd3) with new elements' names, the pair (PO3-3, PurOrd3-3) consists of three copies of the pair (PO3, PurOrd3), etc. The number of elements in the pairs vary between 42/44 and 602/632, while the number of leaves vary between 33 and 495. Table 2.8 contains a summary of these datasets. The input size represents multiplication of the leaf counts of the two trees.

[4]Doan, AnHai, http://www.cs.washington.edu/research/xmldatasets

Table 2.7 Real-world datasets.

Datasets	# elements	# leaves	Max depth	Average depth	# coreferent elements
A: FreeDB	11	8	4	3.1	7
B: Discogs	39	33	5	3.9	
A: PO1	10	7	4	3.1	7
B: PurOrd1	9	7	3	2.4	
A: PO2	13	8	4	3.4	8
B: PurOrd2	15	8	4	3.9	
A: PO3	41	33	4	3.2	28
B: PurOrd3	43	33	4	3.8	
A: Source	14	10	4	3.5	10
B: REED	16	12	4	3.3	
A: Source	14	10	4	3.5	10
B: UWM	21	15	5	3.9	
A: Source	14	10	4	3.5	10
B: WSU	20	16	4	3.3	

2.4.2 Experiment: Path comparison

Goal. This experiment was conducted to show the advantages of using our method (referred to as *Paths Matcher*) and the two-level string comparison method as proposed in [8] (referred to as *String Matcher II*).

Procedure. A set of path pairs was compared by Paths Matcher and String Matcher II, accordingly. The latter method was chosen in this comparison because it is similar to *Paths Matcher* while the main difference is that StringMatcher does not take into account the positions of the path elements.

Result. Table 2.11 presents the PTVs denoting the (un)certainty of coreference for selected pairs of paths, calculated using both methods. The columns correspond to particular methods and rows contain selected paths from the CD and university course datasets. Path pairs from the CD datasets have equal number of steps and different steps at the beginning, but are more similar at the end. As Paths Matcher takes into account this feature of coreferent paths, it gives much better results for these pairs than the other method. However, when the paths are similar at the beginning the results obtained using Paths Matcher are only slightly better. Thus, our method properly implements the assumption that not all steps are equally important and that differences at the beginning of the paths do not exclude the paths' coreference.

Table 2.8 Synthetic datasets.

Datasets	# elements	# leaves	Max depth	Average depth	Input size
A: PO3-1	42	33	5	4.2	1089
B: PurOrd3-1	44	33	5	4.8	
A: PO3-2	82	66	5	4.2	4356
B: PurOrd3-2	86	66	5	4.8	
A: PO3-3	122	99	5	4.2	9801
B: PurOrd3-3	128	99	5	4.8	
A: PO3-5	202	165	5	4.2	27225
B: PurOrd3-5	212	165	5	4.8	
A: PO3-10	402	330	5	4.2	108900
B: PurOrd3-10	422	330	5	4.8	
A: PO3-15	602	495	5	4.2	245025
B: PurOrd3-15	632	495	5	4.8	

Paths Matcher properly deals with the differences in the schema structures (e.g., the fourth row of Table 2.11), abbreviations (e.g., "tracklist" and "tracks" in the fourth row) and misspellings. This is confirmed by the larger PTVs returned by our method for truly coreferent elements (cf., e.g., "/cddb/disc/year" and "/discs/disc/year" in the first row of Table 2.11) and lower PTVs for elements that are not coreferent, c.f. for instance "/root/courseListing/restrictions" and "/root/course/instructor" in row 5 of Table 2.11.

2.4.3 Evaluation settings

To determine the quality of our approach *Paths Matcher*, we compared results obtained by *Paths Matcher* against the manually derived results from the Tables 2.9 and 2.10. We can distinguish three sets. The first set, true positive B, contains the truly coreferent objects which are automatically derived by the approach. The second set, false negative A, contains truly coreferent objects which are not identified. The last set, false positive C, are objects falsely identified as coreferent. Based on the cardinality of these sets we specified three quality measures of precision, recall and F-Measure. These are commonly used in Information Retrieval [85].

Precision is one of the important measures of classifiers quality and is defined in our case as the fraction of truly coreferent objects among all objects classified by a given algorithm as being coreferent:

$$\text{Precision} = \frac{|B|}{|B| + |C|}. \tag{2.23}$$

Table 2.9 Real coreferent paths of university courses datasets.

Source vs Reed	Source vs UWM	Source vs WSU
/root/course/courseNumber	/root/course/courseNumber	/root/course/title
/root/course/crse	/root/courselisting/course	/root/course/title
/root/course/section	/root/course/title	/root/course/instructor
/root/course/sect	/root/courselisting/title	/root/course/instructor
/root/course/title	/root/course/credits	/root/course/time/start
/root/course/title	/root/courselisting/credits	/root/course/times/start
/root/course/instructor	/root/course/section	/root/course/time/end
/root/course/instructor	/root/courselisting/sectionlisting/section	/root/course/times/end
/root/course/place/building	/root/course/time/day	/root/course/place/building
/root/course/place/building	/root/courselisting/sectionlisting/days	/root/course/place/bldg
/root/course/credits	/root/course/time/start	/root/course/place/room
/root/course/units	/root/courselisting/sectionlisting/hours/start	/root/course/place/room
/root/course/place/room	/root/course/time/end	/root/course/time/day
/root/course/place/room	/root/courselisting/sectionlisting/hours/end	/root/course/days
/root/course/time/start	/root/course/place/building	/root/course/courseNumber
/root/course/time/starttime	/root/courselisting/sectionlisting/bldgandrm/bldg	/root/course/crs
/root/course/time/end	/root/course/place/room	/root/course/credits
/root/course/time/endtime	/root/courselisting/sectionlisting/bldgandrm/rm	/root/course/credits
/root/course/time/day	/root/course/instructor	/root/course/section
/root/course/days	/root/courselisting/sectionlisting/instructor	/root/course/sect

High precision means that the method returns more relevant (coreferent) than irrelevant results. Recall is another important quality measure which in our case can be defined as the fraction of true positive objects among all coreferent objects present in a test dataset:

$$\text{Recall} = \frac{|B|}{|A| + |B|}. \tag{2.24}$$

High recall means that a method is capable of discovering most of the actually relevant (coreferent) objects. F-Measure is the last of the measures. It is the harmonic mean of precision and recall:

$$\text{F-Measure} = 2 \times \frac{Precision \times Recall}{Precision + Recall}. \tag{2.25}$$

Neither precision nor recall alone can accurately express matching quality. On the one hand, precision can be usually maximised at the expense of poor recall by reducing the number of returned objects. On the other hand, high recall can be achieved at the expense of poor precision by increasing the number of returned objects. Therefore, the F-Measure is used in our evaluation as a measurement for the balance between precision and recall.

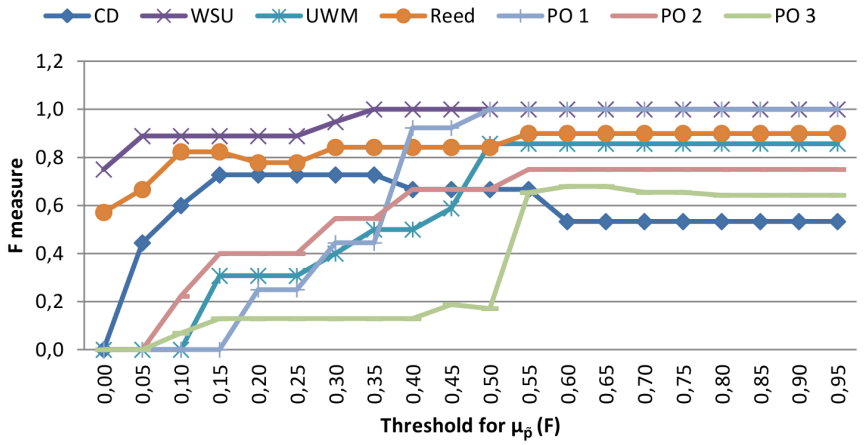
Our algorithm employs two thresholds ($threshold_T$ and $threshold_F$) to decide on the paths' coreference (see Section 1.4.4). Thresholds are set to maximise quality measures (precision, recall and F-Measure[5]). We have observed that the

[5]Paths with an unknown coreference status are considered as non-coreferent for the calculation of

Table 2.10 Real coreferent paths of Purchase Order datasets PO1 and PO2 and also CD datasets.

Pair PO1	Pair PO2	Discogs vs FreeDB
/po/orderNo /purchaseOrder/orderNo	/po/poShipTo/street /purchaseOrder/deliverTo/address/street	/discs/disc/id /cddb/disc/did
/po/purchaseInfo/shippingAddr /purchaseOrder/shipTo	/po/poShipTo/city /purchaseOrder/deliverTo/address/city	/discs/disc/artists/name /cddb/disc/artist
/po/purchaseInfo/billingAddr /purchaseOrder/billTo	/po/poBillTo/street /purchaseOrder/invoiceTo/address/street	/discs/disc/title /cddb/disc/dtitle
/po/purchaseInfo/lines/item /purchaseOrder/items/item	/po/poBillTo/city /purchaseOrder/invoiceTo/address/city	/discs/disc/genres /cddb/disc/category
/po/purchaseInfo/lines/quantity /purchaseOrder/items/qty	/po/poLines/item/line /purchaseOrder/items/item/lineNumber	/discs/disc/styles /cddb/disc/genre
/po/purchaseInfo/lines/unitOfMeasure /purchaseOrder/items/uom	/po/poLines/item/qty /purchaseOrder/items/item/quantity	/discs/disc/year /cddb/disc/year
/po/purchaseDate /purchaseOrder/date	/po/poLines/item/uom /purchaseOrder/items/item/unitOfMeasure	/discs/disc/tracklist/title /cddb/disc/tracks/title
	/po/poLines/count /purchaseOrder/items/itemCount	

Figure 2.11 The F-Measure of Paths Matcher for different datasets depends on the setting of threshold for $\mu_{\tilde{p}}(F)$.



quality measure of matching for each dataset depends on the setting of $threshold_F$ for $\mu_{\tilde{p}}(F)$, while $threshold_T$ for $\mu_{\tilde{p}}(T)$ does not depend on the dataset and can easily be fixed to 0.5. Figure 2.11 shows that the F-Measure for each dataset depends on the setting of the threshold for $\mu_{\tilde{p}}(F)$. For instance, the best results for the CD data are achieved for $threshold_F$ between 0.15 and 0.35.

The schema matching systems that are used in our comparison have a set of tunable parameters. To determine the optimal setting for the Cupid and QMatch

precision and recall.

Table 2.11 Comparison of two algorithms for detecting paths coreference.

Paths	**Paths Matcher**	**String Matcher II**
/cddb/disc/year	**T: 1.00**	T: 1.00
/discs/disc/year	**F: 0.06**	F: 0.25
/cddb/disc/genre	**T: 1.00**	T: 1.00
/discs/disc/genres	**F: 0.06**	F: 0.25
/cddb/disc/dtitle	**T: 1.00**	T: 1.00
/discs/disc/title	**F: 0.12**	F: 0.25
/cddb/disc/tracks/title	**T: 1.00**	T: 1.00
/discs/disc/tracklist/title	**F: 0.17**	F: 0.42
/root/course/instructor	T: 1.00	T: 1.00
/root/courseListing/restrictions	F: 0.56	F: 0.12
/root/course/title	T: 1.00	T: 1.00
/root/courseListing/title	F: 0.19	F: 0.04
/root/course/days	**T: 1.00**	T: 1.00
/root/courseListing/sectionListing/days	**F: 0.33**	F: 0.67

algorithms, a set of experiments was run [76]. The threshold of the label matcher, which is used in both algorithms to determine whether two labels should be considered as a match, is set to 0.45. Next, the significance value attributed to the match value of the label when computing the path match is 0.8 for Cupid and 0.43 for QMatch; the significance value attributed to the quality of matching of children when computing the paths match is 0.2 for Cupid and 0.285 for QMatch. Moreover, the overall threshold value used to make a decision on whether or not to increment or decrement leaf similarity values based on ancestor values is 0.6 for Cupid and 0.7 for QMatch. The increment and decrement constants for both algorithms by which the similarity values are increased or decreased are fixed to 0.1 and 0.075 respectively. QMatch uses two more parameters. The first, i.e. the significance value of the property set (i.e. max occurrence, order of elements, data type) attributed to the match degree when computing the paths match is 0.285. The second parameter, i.e. the weight attributed to the path difference when computing the paths match for non-leaf nodes is 0.3. Moreover, to secure a fair comparison of QMatch and Cupid, a modified version of the former, denoted QMatch', is used which limits the property match to a data type match.

The parameters used to fine tune the HMAT system are the following: linguistic similarity threshold (lt), root similarity threshold (rt), descendant similarity threshold (dt), and sibling similarity threshold (st). Only similarity measures which exceed these predefined thresholds are considered for evaluation. The sys-

tem gave optimum performance for the following values: lt=0.4, rt=0.5, dt=0.4, and st=0.4 [75]. In contrast, MatchingLu employs one threshold which is fixed to 0.28 in our experiments.

The last matching system which we compared with our Paths Matcher is COMA. A key feature of COMA is its flexible support for multiple, independently executable matchers that can be executed within a user-controlled match process or workflow. We used the workflows *OnlyNodesW* and *AllContextW*, which returned the best results. AllContextW identifies and matches all contexts by considering all paths (sequences of nodes) for a shared element from the root to the element, while OnlyNodesW creates matching without contexts. Also, both workflows apply predefined matchers, such as label matcher, path matcher, structure matcher and synonyms matcher.

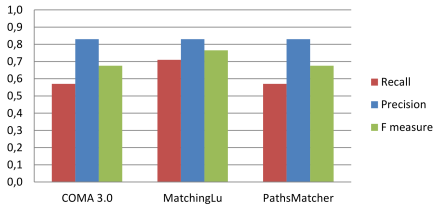
2.4.3.1 Experiment: Comparison results

Goal. In this experiment, accuracy of Paths Matcher is evaluated by comparing its results with a ground truth taxonomy and with alternative approaches.

Procedure. The accuracy of our method and the alternative approaches is measured by precision, recall and F-Measure. First, the result of Paths Matcher is compared to COMA 3.0 and MatchingLu on the CD dataset. Next, our method is evaluated by comparing the result with all alternative approaches on the PO datasets and the university datasets.

Result. Figure 2.12 presents the precision, recall and F-Measure for the compact disc data used in our experiments and calculated for the results obtained using our algorithm (Paths Matcher), COMA 3.0 and MatchingLu. Paths Matcher and COMA returned ex aequo results for the CD dataset, while MatchingLu was slightly better. However, it should be noted that our method is a simple matcher based only on labels, in contrast to other methods which use a set of predefined matchers, synonyms, abbreviations mappings tables (in the case of COMA) and a WordNet dataset (in the case of the MatchingLu method).

Figure 2.12 Precision, recall and F-Measure of our method (Paths Matcher), COMA 3.0 and MatchingLu for CD data set.



Moreover, our approach is compared with Cupid, QMatch', QMatch, HMAT, and with COMA 3.0 and MatchingLu using other datasets. Figure 2.13 and Figure 2.14 show the results for three pairs of schemas of the Purchase Order and University Courses datasets, respectively. All of the methods return fairly good results. However, as was mentioned above, our approach does not use synonyms and abbreviations mappings as does COMA or QMatch. This explains the worse results of Paths Matcher for pairs PO2 and PO3 in Figure 2.13, where knowledge about synonyms is crucial in order to detect true positive mappings. Surprisingly, MatchingLu returns worse results for these two sets even though it uses WordNet because it has failed to map the elements' names with *noise* characters, e.g., "poShipTo" and "poBillTo" with "deliverTo" and "invoiceTo", respectively. Nonetheless, Paths Matcher returns better results than other methods for some pairs of schemas that differ the most in structure and names (abbreviations) of their elements, i.e. for pair PO1 in Figure 2.13 or Source vs WSU in Figure 2.14.

Figure 2.13 Precision, recall and F-Measure of Purchase Order data for our method (PathMatcher), Cupid, QMatch', QMatch, COMA 3.0, HMAT and LuMatching.

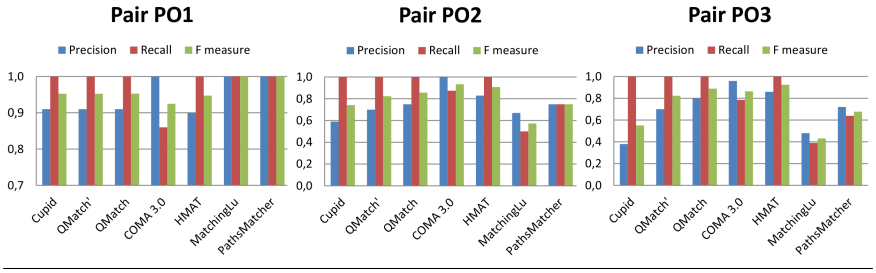
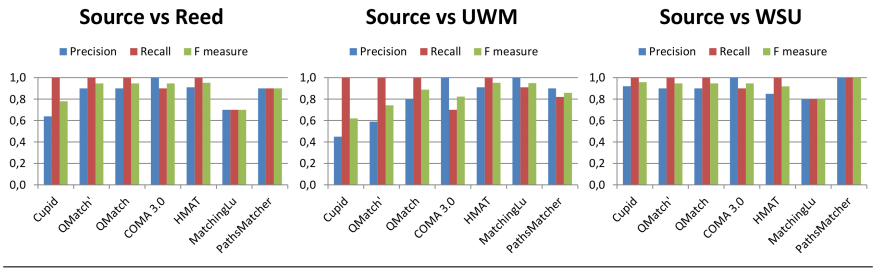


Figure 2.14 Precision, recall and F-Measure of University Courses data for our method (PathMatcher), Cupid, QMatch', QMatch, COMA 3.0, HMAT and LuMatching.

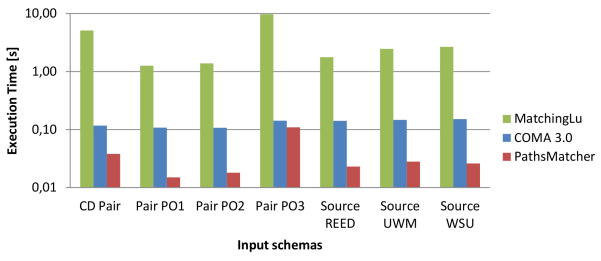


2.4.3.2 Experiment: Execution time

Goal. In the last experiment, we investigated to what extent the use of the Paths Matcher method introduces a computational overhead with respect to alternative approaches.

Procedure. The execution time of ours and alternative approaches (COMA 3.0 and MatchingLu) was measured in this experiment. These approaches were selected for the comparison for the following reasons. COMA 3.0 is a well-known approach that uses different schema information and decides on matching based on the combined results of several individual matchers. On the other hand, MatchingLu decides on matching by using an external source of information, i.e. WordNet. Thus both methods implement rather complex and extra data demanding matching algorithms, in contrast to our matcher which implements a simple approach which basically boils down to matching schema elements based only on path comparison. This experiment is composed of two sub-experiments. The first sub-experiment is conducted to measure the execution time for matching small schemas (the number of leaves is not larger than 33) which are used in the literature to check the quality of schema matching methods. For each schema pair, the matching procedure is repeated 100 times in order to obtain mean time evaluation. The second sub-experiment is conducted to collect the execution times of the schema matching methods under comparison for larger schemas, where the number of leaves varies between 33 and 495. We subdivided the input size, represented by multiplication of the leaves count of the two schema trees into intervals and then calculated the average execution times for each interval (each experiment is repeated 100 times).

Figure 2.15 Execution time of matching small schemas for our method (PathMatcher), COMA 3.0 and MatchingLu.



Result. The results of the first sub-experiment are shown in Figure 2.15 (for each schema pair separately) and in Figure 2.16 (average execution time). Our approach needs, on average, only 37 ms to establish matching, whereas COMA 3.0 does the

same almost 3 times slower. However, MatchingLu generates matching with a similar quality (confirmed by F-Measure in Figure 2.16) on average more than 90 times slower than our approach and 25 times slower than COMA 3.0. Computing semantic similarities by MatchingLu is a very expensive task: given two words, the program algorithm checks all of their relations stored in WordNet and tries to find the highest ranked connection. Even though they restrict the relation to synonymy and hypernymy only, the searching space in WordNet is still huge. Moreover, the normalized rate between execution time and F-Measure is calculated. Figure 2.16 shows these rates for the evaluated methods. Namely, it equals 0.007 for the MatchingLu, that is based on WordNet, 0.225 for COMA and 0.768 for our method.

The results of the second sub-experiment are shown in Figure 2.17. Our method also copes well with large schemas. Paths Matcher outperforms COMA 3.0, especially for the largest schemas.

Figure 2.16 Normalized rate between execution time and F-Measure for our method (PathsMatcher), COMA 3.0 and MatchingLu.

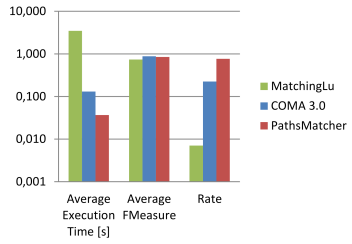
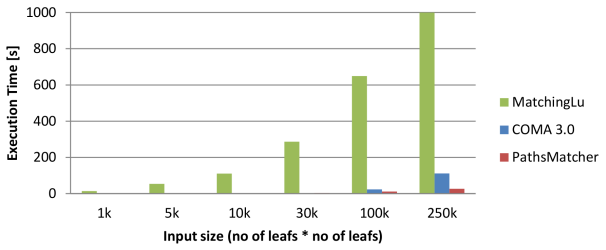


Figure 2.17 Execution time of matching large schemas for our method (PathMatcher), COMA 3.0 and MatchingLu.



2.4.4 Experiment: Schemas coreference

Goal. In this experiment the coreference of whole schemas is evaluated.

Procedure. PTVs expressing coreference of all matched paths are aggregated using the Sugeno integral, as defined in Definition 4. The resulting aggregated PTV then expresses the (un)certainty about the coreference between the two input XML schemas.

Result. Table 2.12 presents aggregated PTVs of XML schema pairs from our datasets. For truly coreferent schema pairs we have obtained high certainty degrees about their coreference, and for non-coreferent these degrees were low. The highest certainty about coreference of schema pairs was obtained for (Source,Reed) and (Source,WSU), which in fact are the most similar. On the other hand, other schema pairs have lower certainty of coreference because they significantly differ in terms of tags names and structure.

Table 2.12 Schemas coreference.

Datasets	**FreeDB Discogs**	**Source Reed**	**Source UWM**	**Source WSU**	**Pair PO1**	**Pair PO2**	**Pair PO3**
PTV	T: 1.00 F: 0.20	T: 1.00 F: 0.00	T: 1.00 F: 0.19	1.00 0.00	1.00 0.33	1.00 0.30	1.00 0.20

2.5 Conclusions

In this chapter we propose a novel method for finding coreferent schema elements based on coreferent paths, which may help to further decide on the coreference of the whole XML schemas. Paths are one of the most crucial metadata in XML files. Detection of coreferent paths requires recognising coreferent steps which paths are composed of. We treat coreference as a binary notion, i.e., two paths are either coreferent or not. However, we assume that the results of coreference detection may be uncertain, which is represented by employing possibilistic truth values.

Coreference is considered here in a hierarchical way. On the basic level, the coreference of steps (parts of XML paths) is determined by a low-level string comparison method [8].

Then information on step coreference, with an explicit representation of its related uncertainty using PTVs, is properly aggregated to obtain information on paths coreference which is, in turn, further aggregated to finally decide on the coreference of whole XML schemas. We apply a Sugeno integral for PTVs to aggregate information on the coreference of subsequent levels of this hierarchy. This allows us to explicitly cope with the position of the elements in a path and with the relative importance of paths within their schema which are set up by our novel techniques.

The presented novel schema matching approach is the answer for the first research question set in this PhD which concerns syntactical matching of corresponding schema elements of heterogeneous datasets based only on schema information. In contrast, in the next chapter a novel schema matching approach which is based on content data is proposed as the answer for the next research question. More specifically, statistical analysis and lexical comparison of content data, and efficiently detected coreferent tuples across heterogeneous datasets are used to establish semantical matching between corresponding schema elements. As will be explained this schema matching approach improves the results of the schema matching at the expense of using additional information.

3

Coreference detection in schema based on coreferent content data

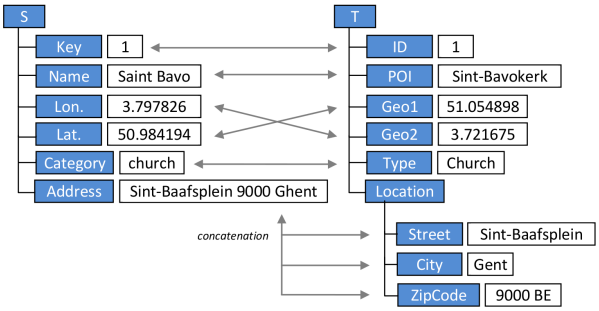
3.1 Introduction

The existence of coreferent content data (coreferent tuples, duplicates) which describe the same entity but in a different way across multiple, related databases significantly lowers data quality and should be avoided. However, a small number of coreferent tuples can be useful in the data integration process, which involves importing data from one source to another. Namely, coreferent tuples may be helpful in establishing a true matching between the corresponding attributes of heterogeneous database schemas. This is known as the *schema matching* problem, which is the first step in data integration and is investigated in this chapter. In contrast to the schema matching approach in the previous chapter which is based only on schema information, here we exploit also data to match the schemas.

3.1.1 Problem illustration

As a motivating example let us consider the schema matching scenario in which corresponding attributes (schema elements) in the source dataset S in the leftmost XML tree in Figure 3.1 and the target dataset T in the rightmost XML tree in Figure 3.1 have to be aligned as in Figure 3.1. Without loss of generality, XML data can be represented in Tables such as the source dataset in Table 3.1 and the

Figure 3.1 POI data example: parts of XML documents from the source dataset S (left tree) and the target dataset T (right tree). The filled boxes are the attributes (elements) defined in the schema, while the not-filled ones are their values. Arrows represent mappings of coreferent schema elements (metadata).



target dataset in Table 3.2[1]. The attributes "Key", "Name", "Lat", "Lon" and "Category" in the left-hand tree in Figure 3.1 and Table 3.1 have to be matched to the attributes "ID", "POI", "Geo1", "Geo2" and "Type" in the right-hand tree in Figure 3.1 and Table 3.2, respectively. It is obvious that matching techniques which are based on the attributes' names (such as Paths Matcher in Chapter 2) are not capable to establish all of these matchings. Semantical matching of corresponding attributes has to be established as coreferent attributes may have different names. Moreover, an attribute "Address" in Figure 3.1 and in Table 3.1 is decomposed into a number of (sub)attributes: "Street", "City", "ZipCode" in Figure 3.1 and in Table 3.2. Thus, a one-to-many matching between these attributes is required; namely, a concatenation function has to be applied to solve the *attribute granularity* problem. In general also the *coverage* problem of matched attributes exists, i.e. coreferent attributes do not necessarily completely have to represent the same information; for example, the attribute "Address" in Figure 3.1 and in Table 3.1 does not contain information about the country. Moreover, due to errors, inaccuracies and lack of standardisation, coreferent data are not bound to be equal, i.e. the Belfry in Ghent has a different category in the considered tables. It should be clear that detected coreferent tuples do not guarantee perfect schema matching, i.e. the attributes "Name" and "Type" may contain similar values, e.g. *cafe* or *theatre*, which may mislead the matching system. Therefore, all of this makes the finding of coreferent data in schemas using content data a challenging task.

Examples of coreferent tuples are the objects described in the first, second, fourth, fifth and sixth rows in Table 3.1 and Table 3.2, respectively. They have slightly different names, similar geographic coordinates and different categories

[1]The order of datasets does not matter, i.e. there exists schema matching between corresponding attributes from the source dataset and the target dataset, and vice versa.

Table 3.1 Example of objects extracted from the source dataset S

Key	Name	Lon.	Lat.	Category	Address
1	Belfry & Cloth Hall	3.724911	51.053653	Tourist Attract	Sint-Baafsplein, 9000 Ghent
2	Saint Bavo	3.797826	50.984194	church	Sint-Baafsplein, 9000 Ghent
3	Cafe-Restaurant De Ster	4.050876	51.281777	restaurant	Grotestraat 91, 7471 BL Goor
4	Het Kouterhof	3.665122	51.034331	lodging	Stoopkensstraat 24, 3320 Hoegaarden
5	Borluut B&B	3.657992	51.018882	lodging	Kleine Gentstraat 69, 9051 St-Denijs-Westrem
6	Gravensteen Hotel	3.719741	51.056485	hotel	Jan Breydelstraat 35,9000,Ghent
7	Carlton Hotel	3.713951	51.036280	lodging	Chartreuseweg 20, 8200 Brugge
8	Vlaamse Opera	3.722336	51.049746	theater	Schouwburgstraat 3, 9000 Ghent

Table 3.2 Example of objects extracted from the target dataset T

ID	POI	Geo1	Geo2	Type	Street	City	ZipCode
1	Belfort en Lakenhalle	51.054898	3.721675	Bell Tower	Emile Braunplein	Gent	9000 BE
2	Sint-Bavokerk	51.054898	3.721675	Church	Sint-Baafsplein	Gent	9000 BE
3	Cafe Theatre	51.049830	3.722015	Restaurant	Schouwburgstraat 5-7	Gent	9000 BE
4	Het Kouterhof	51.034379	3.665140	Hotel	Stoopkensstraat 24	Hoegaarden	3320 BE
5	Borluut Bed Breakfast	51.018938	3.657975	Hotel	Kleine Gentstraat 69	St-Denijs-Westrem	9051 BE
6	Hotel Gravensteen	51.056465	3.719741	Hotel	Jan Breydelstratt 35	Gent	9000 BE

and addresses, but they are still describing coreferent objects. These detected coreferent tuple pairs in the considered datasets are used to derive schema matching, known as *horizontal matching*. The same or similar attribute values among coreferent tuple pairs imply coreference of the corresponding attributes of the schemas.

However, detecting coreferent tuple pairs without having knowledge about the correspondences between the attributes of heterogeneous schemas (known as schema alignment) is time-consuming and error prone. It requires the comparison of the values of each attribute from one schema with the values of each attribute from the other schema. Thus, one of the main challenges in the efficient detection of coreferent tuple pairs is the reduction of the set of attributes involved in the comparison to those that may correspond to each other. For this purpose our content data-based approach statistically and lexically compares the attributes' domains and selects potentially corresponding (coreferent) attributes which are called *candidate* attributes. This method is known as vertical matching. It significantly decreases the number of comparisons and increases the quality of coreferent tuple pairs detection. Candidate attributes give the first tips of the coreference among attributes, which is confirmed or rejected by detected coreferent tuples or even may be the basis to establish schema matching in case of a lack of coreferent tuples. However, it should be clear that vertical matching is necessary but not sufficient for efficient attribute coreference identification. Thus, our approach is a combination of vertical and horizontal schema matching methods used to establish the matching of corresponding attributes.

Many problems have to be addressed while devising such a schema matching algorithm. To sum up, the most important among them are the following:

- How can content data be useful in schema matching?

- How can coreferent tuples be detected in unaligned schemas?
- How can one-to-one and one-to-many semantic matching be established between corresponding attributes?
- How can attribute granularity and the coverage matching problems occurrence be recognized?

3.1.2 Contributions

The objective of this chapter is to propose a novel automatic semantical matching method of corresponding (coreferent) attributes in schemas based on data and metadata. More specifically, the detection of coreferent attributes in schemas is based on statistical and lexical analysis of content data and detected coreferent tuples across pairs of datasets, which increases the confidence in schemas matching. In other words, our method is a combination of vertical and horizontal schema matching techniques that applies possibilistic truth values (PTVs) and a kind of cardinality of a set of PTV to express the uncertainty about the matchings. Apart from this, our approach copes with the attribute granularity problem and the information coverage problem.

We will show that even a small number of coreferent tuples is sufficient to establish a correct matching between corresponding attributes of heterogeneous schemas. Such methods can then later be used to improve the coreference detection of data described by schema which are considered as metadata of content data.

3.1.3 Outline

The remainder of this chapter is organized as follows. In Section 3.2, an extensive overview of work related to the topic of this chapter is provided. Next, in Section 3.3.2, an overview of our novel content data-based schema matching algorithm is presented. In Sections 3.4-3.6, the details of the algorithm are studied. In Section 3.7 an experimental study of the proposed methods and techniques is reported. Finally, Section 3.8 summarizes the most important contributions of this chapter.

3.2 Related work

Schema matching can be established by using different methods. Some methods use only data (e.g. duplicates [19, 54]), others use only metadata (e.g. schema information [15–17,86], knowledge base [21]), whereas other methods use both data and metadata, e.g. [18,20]. In this chapter the content data-based schema matching approach is proposed which uses coreferent tuples. There is a large body of work

on schema matching which uses content data [1]; for example, LSD [54] extracts information from a training set and consists of a learning and classification phase. More specifically, given a user-supplied mapping between schema elements, the learning step looks at content data to train the learner, thereby discovering characteristic content data patterns and matching rules. Next, these patterns and rules can be applied to match other schema elements. As opposed the approach in [87] does not require training or learning as in the learning-based or neural network techniques, but captures valuable knowledge about the domain of the attribute. This approach uses regular expression as a formalism to characterize a set of attribute values. Having this regular expression, the corresponding attributes are detected by matching the regular expression of one attribute with the value of another attribute using the *match* function (Java built in function: java.util.regex). In many cases it is still not clear which attributes correspond to each other. Thus, regular expressions are a valuable and useful tool but should be supported by other techniques. As opposed to most instance-based solutions which use summary information (e.g., average value) for attribute classification, we derive schema matching from detected coreferent tuples in the datasets. One schema matching approach using duplicates is ILA [88], which is a domain-independent program that learns the meaning of external information by explaining it in terms of internal categories. ILA considers a pair of objects as duplicates if both objects contain at least one attribute value in common and relies on a high extensional overlap (a number of coreferent tuples). In our opinion these assumptions are unrealistic.

IMap [20] is based on both schema and instance information as well as on a domain ontology and uses past matchings. The duplicates are identified by the user and only exact matches of attribute values are considered by a matcher. IMap copes with various attribute granularities, as in Chua et al. [89] and Lu et al. [28], but the focus is on numerical and differently scaled data (as opposed to our approach which focuses on textual data). Statistical analysis is employed to data in duplicates which are assumed to be identified by a common ID attribute. This means that at least one attribute is already aligned. The approach of Chua et al. [89] classifies attributes into domain classes (e.g. categorical) and forms attribute groups (sets of attributes from the same relation which may be corresponding) based on predefined rules. Then, the correspondence scores of pairs of attribute groups are calculated. Finally, attributes are matched based on these scores. The approach of Lu et al. [28], one the other hand, uses correlation analysis techniques (supervised by the user) to identify attributes which are potentially semantically related; secondly, they apply regression analysis to generate the relevant conversion function that allows the attribute values of one database to be transformed into attribute values of the other database.

DUMAS [19], just as in our approach, drops several of the assumptions that were made in the above works: coreferent tuples are automatically detected using

unaligned schemas; a few coreferent tuples being sufficient to establish schema matching (low extensional overlap). Moreover, it does not use any external source of information, such as an ontology; it is a content data-based approach. In contrast to our approach, DUMAS does not apply possibility theory and does not combine vertical and horizontal schema matching methods to detect coreferent tuples and establish schema matching.

3.3 Content data-based schema matching

Before we continue to describe our method for schema matching, first of all we should define the problem more formally.

3.3.1 Problem definition

As a reminder, within the scope of this chapter it is assumed that entities from the real world are described as objects (tuples) which are characterised by a number of *attributes* (features). A *schema* $\mathcal{R}$ of a given dataset, which consists of tuples, is identified by a set of attributes A. For each attribute $a \in A$, let $dom(a)$ denote the domain of a (the set of possible values for attribute a) and let $dom'(a)$ denote the subset of $dom(a)$ comprising the values of a that are actually present in the (tuples of the) dataset.

Two datasets are considered. The source dataset over the schema $\mathcal{R}_S$ with the set of attributes $A_S = \{a_1^S,\ldots,a_n^S\}$ is denoted as S, while the target dataset over the schema $\mathcal{R}_T$ with the set of attributes $A_T = \{a_1^T, \ldots, a_m^T\}$ is denoted as T. The one-to-many schema matching is defined as follows.

Definition 7 (One-to-many schema matching)**.** *A relation M is a schema matching if:*

$$M \subseteq 2^{A_S} \times 2^{A_T} \times \tilde{M} \tag{3.1}$$

where $M = \{m_i\} = \{(A'_S, A'_T, \tilde{m})\}$, $A'_S \subseteq A_S$, $A'_T \subseteq A_T$ and A'_S or A'_T is a singleton set, $\tilde{M}$ is the set of PTVs and $\tilde{m} \in \tilde{M}$ expresses the certainty degree to which A'_S matches A'_T.

Some additional properties may be associated with each matching $m \in M$. For example, the *local matching cardinality*, denoted $card_m^l$, is the number of matched attributes in m, i.e., $card_m^l$ = $|A'_S|$+$|A'_T|$ (e.g. $card_m^l$ is equal to 2 for a one-to-one matching). Moreover, particular matchings may be classified to a $type$. The following matching types are distinguished:

- *full matching* (coverage level 1): corresponding attributes have the same meaning and cover completely the same concept, e.g. “Name” and “POI” or “Type” and “Category” in Figure 3.1;

- *inclusion matching* (coverage level 0.5): corresponding attributes have partially the same meaning and do not cover completely the same concept, e.g. "Address" in the source S in Figure 3.1 represents the address of a POI which consists of a street, house number, city and zip code, and this is a part of the concatenation of the attributes "Street", "City" and "ZipCode" in the target T in Figure 3.1, which consists of the same information as address from the source but is extended by a country code. "Street" in the target T in Figure 3.1 represents only a part of the address from the source. Thus, two sub-types of matching are considered: the *source is a part of the target* and the *target is a part of the source*, respectively;

- *has a common part matching* (coverage level 0.3): corresponding attributes have partially the same meaning, do not cover completely the same concept and are not an inclusion matching. E.g. the matching between "Address" in the source S and "ZipCode" in the target T in Figure 3.1. "Address" represents the address of a POI which consists of a street name, house number, city and zip code without the country code; while "ZipCode" in the target T in Figure 3.1 represents the zip code and country code of a POI, thus only the zip code is a common part.

- *unknown* (coverage level 0): if attributes do not match.

- *concatenation*. This is a special case of attribute matching which combines two or more attributes. Combining matching types might result in another matching type. For instance, a combination of two inclusion matchings may give a full matching (of attributes) or an inclusion matching. E.g. let assume inclusion matchings between attributes from the source S and the target T in Figure 3.1: "Address" and "Street"; "Address" and "City"; "Address" and "ZipCode". Concatenation of these matchings gives a full matching.

The matching $m \in M$ can be interpreted as a one-to-one matching of corresponding attributes if the cardinalities of A'_T and A'_S are equal to 1.

Furthermore, in the context of a one-to-many schema matching M, we consider a set D of coreferent tuple pairs which is defined as follows.

Definition 8 (A set D of coreferent tuple pairs)**.** *A set D of coreferent tuple pairs consists of 4-tuples $d = (t^S, t^T, M_V, \tilde{d})$ where t^S and t^T are coreferent tuples from the source and target datasets, respectively, M_V is a set of attributes matchings for which there are coreferent values in both particular tuples t^S and t^T, and $\tilde{d}$ is a PTV representing the (un)ceratinty that two tuples $t^S \in S$ and $t^T \in T$ are coreferent.*

3.3.2 Algorithm

The novel content data-based schema matching Algorithm 3.1 creates matchings between corresponding attributes A_S and A_T of the source dataset S and the target dataset T, respectively, using content data. Therefore, the inputs for the algorithm are the source and target datasets (S and T, respectively), and a set of parameters (P_V, P_H and P_C for each phase) which are used to establish a schema matching. The objective of our algorithm is to establish as many valid one-to-one or one-to-many schema matchings M for coreferent attributes as possible. Our approach is classified as the hybrid content only based matching method according to the classification from [1], which is presented in Figure 2.3.

Algorithm 3.1 SCHEMAMATCHINGALGORITHM

Require: Dataset S, Dataset T, Parameters P_V, Parameters P_H, Parameters P_C
Ensure: Schema Matching M
1: $M_V \leftarrow$ getVerticalMatchings($\{dom'(a^S)\}_{a^S \in A_S}$, $\{dom'(a^T)\}_{a^T \in A_T}$,$P_V$)
2: $M \leftarrow$ getHorizontalMatchings(S,T,M_V,P_H)
3: $M \leftarrow$ resolveConflicts(M,P_C)

The Algorithm 3.1 is composed of three main phases. First, vertical schema matchings are established by the method getVerticalMatchings which compares the domains of particular attributes (line 1 in Algorithm 3.1, which is further discussed in Section 3.4). Second, the established vertical matchings M_V are used to detect coreferent tuple pairs in the heterogeneous data sources which, in turn, constitute a basis to generate horizontal schema matchings M by using the method getHorizontalMatchings (line 2 in Algorithm 3.1, which is further discussed in Section 3.5). Finally, conflicts are resolved by the method resolveConflicts (line 3 in Algorithm 3.1, which is further discussed in Section 3.6). These steps are described in detail in the following sections.

3.4 Phase I: Vertical matching

The first phase of our novel schema matching approach is the generation of one-to-one and one-to-many vertical matchings between corresponding attributes. These matchings are established by Algorithm 3.2 based on statistical analysis and lexical comparison of attribute domains. Thus the input for the algorithm are the subsets of the domains consisting of these values that actually occur in tuples of respective datasets, $\{dom'(a^S)\}_{a^S \in A_S}$ and $\{dom'(a^T)\}_{a^T \in A_T}$, and also a set P_V of parameters which define the thresholds and submatcher settings and is detailed further on. This phase consists of three steps.

In the first step, "Statistical analysis of content data", the subsets of the attribute

domains are statistically compared by the *statistical matcher*, and if particular subsets are coreferent, then the matching between their corresponding attributes is established (lines 2-18 in Algorithm 3.2); otherwise the attribute domains are lexically compared in the second step called "Overlapping" by the *lexical matcher* (lines 19-25 in Algorithm 3.2). More specifically, each pair of attributes is processed sequentially by the following techniques. First, the results of the statistical analysis of the subsets of the attribute domains (such as the analysing the average length, average values, called attribute properties) are compared, which is a relatively computationally non-expensive statistical technique. Second, only if coreference between two attributes is not declared then the intersection of the subsets of their domains, which are represented by multisets of terms, is calculated based on the equality relation, i.e. two terms are added to the intersection if they are equal. Thus, two attributes are considered as coreferent if a cardinality of the intersection exceeds threshold. Third, if coreference between the attributes is still not declared, then the subsets of their domains are calculated analogously to the second technique but based on the low-level string comparison technique [8] instead of the equality relation. This is the most computationally expensive method of the three, but it is also the most valuable because non-equal but coreferent terms can be detected. The established matchings are added to the set $M_V^{1:1}$ of the one-to-one schema matchings. Next, in the third step, called "Generalization", from the established one-to-one schema matchings in $M_V^{1:1}$, a one-to-many schema matching ($\in M_V^{1:n}$) is generated (line 28 in Algorithm 3.2). Finally, the vertical schema matching M_V is composed of the one-to-one schema matchings $M_V^{1:1}$ and the one-to-many schema matchings $M_V^{1:n}$ (line 29 in Algorithm 3.2). These steps are described in detail in the following subsections.

3.4.1 Step 1: Statistical analysis of content data

In the first step the attribute domains of each schema are statistically analysed separately using predefined *Data Analysers* $P_V.AN$ (lines 2-8 in Algorithm 3.2). This returns a set of *properties* for each attribute which are considered as a basis for some heuristics for determining the coreference of attributes. There is a large body of work of such properties and heuristics [90–93]. Thus, we give only some examples of such properties and also give an example of an application. These aspects are subject to further research and outside the scope of the work. The proposed examples of such heuristics are the following:

- average, minimum and maximum length as a number of characters in a value without white spaces (numbers are considered as character strings, e.g., telephone numbers, bank accounts, etc.);
- average, minimum and maximum number of tokens for alphabetic and alphanumerical data types. Each value is tokenised, which results in a set of

Algorithm 3.2 VERTICALMATCHINGALGORITHM

Require: $\{dom'(a^S)\}_{a^S \in A_S}$, $\{dom'(a^T)\}_{a^T \in A_T}$, Parameters P_V
Ensure: Schema Matching M_V

Schema Matching $M_V^{1:1} \leftarrow$ null
Properties $P_S[]$, $P_T[]$
for all $a^S \in A_S$ **do**
    $P_S[a^S] \leftarrow$ getProperties($dom'(a^S)$,$P_V.AN$)
end for
for all $a^T \in A_T$ **do**
    $P_T[a^T] \leftarrow$ getProperties($dom'(a^T)$,$P_V.AN$)
end for
for all $a^S \in A_S$ **do**
    for all $a^T \in A_T$ **do**
        Matching $m \leftarrow$ null
        if compareStats($P_S[a^S]$,$P_T[a^T]$) $> P_V.thrStats$ **then**
            $m.A'_S \leftarrow a^S$
            $m.A'_T \leftarrow a^T$
            $m.\pi_{\mathbb{N}} \leftarrow \pi^1_{\mathbb{N}}$
            $M_V^{1:1} \leftarrow M_V^{1:1} \cup m$
            continue
        end if
        $m.\pi_{\mathbb{N}} \leftarrow$ compareDom($dom'(a^S)$,$dom'(a^T)$,P_V)
        FuzzyInteger $\pi^{thr}_{(dom^S,dom^T)} \leftarrow$ getThr($P_V.thrOverlap$)
        if $\pi^{thr}_{(dom^S,dom^T)} \prec_{sup} m.\pi_{\mathbb{N}}$ **then**
            $m.A'_S \leftarrow m.A'_S \cup a^S$
            $m.A'_T \leftarrow m.A'_T \cup a^T$
            $M_V^{1:1} \leftarrow M_V^{1:1} \cup m$
        end if
    end for
end for
Schema Matching $M_V^{1:n} \leftarrow$ getGeneralization($M_V^{1:1}$)
$M_V \leftarrow M_V^{1:1} \cup M_V^{1:n}$

substrings which are called tokens. In most cases, the tokens are separate words. In our approach, the tokenisation of a value is equivalent to subdividing the value in a multiset of tokens and deleting all white spaces in a value;

- average, minimum and maximum value for numerical data types;
- data type: numerical (values contain only numbers), alphabetic (values contain only letters and special characters), alphanumerical (values contain any characters);

Next, attributes a^S and a^T that have similar properties are considered as potentially coreferent (candidate attributes, line 12 in Algorithm 3.2) and the established matching m between them is added to the set of matchings $M_V^{1:1}$ (lines 13-16 in Algorithm 3.2, similarity for all properties is assumed here). The statistical criteria are very strict, thus this matching is assigned a full certainty which is expressed by the fuzzy integer $\pi^1_{\mathbb{N}}$ that $\forall x \in \mathbb{N}\ \pi^1_{\mathbb{N}}(x) = 1$. The basis to decide if properties are similar is the similarity function. We use a simple function that calculates the similarity of properties for particular attributes as a normalised difference of property values. The returned values are within the unit interval $[0, 1]$, where 1 means strong similarity and 0 means a complete lack of similarity. Properties with a similarity above threshold $P_V.thrStats$ are considered to be similar. The similarity function is defined by Equation (3.2) and is applied for all properties, except for data type property which is considered similar only if compared data types are the same.

$$\text{simProp}(a^S, a^T) = 1 - \frac{|\text{propVal}(a^S) - \text{propVal}(a^T)|}{|\text{propVal}(a^S)| + |\text{propVal}(a^T)|} \tag{3.2}$$

Hereby $a^S \in A_S$ and $a^T \in A_T$, and *propVal* is a method which gets the value of a particular property, e.g., the maximum length of the values for an attribute a^S (or a^T).

Remark. *Information from the schema, e.g. maximum value, etc., is not considered because it might be too general and may mislead the matching algorithm. For instance, let assume a database of students with an attribute "Age" of type INTEGER in the range of* -2^{31} *to* $2^{31} - 1$. *This statistical analysis of the values of the attribute "Age" which are actually presented in the database may return a range of* 20 *to* 29. *This information can be more useful than data type restriction which is defined in the database schema.*

Table 3.3 Properties of attribute domains from the source dataset in Table 3.1. *Num* means numerical datatype, *str* means alphabetic data type, and *str-num* means alphanumeric datatype.

Property	Name	Lon.	Lat.	Category	Address
Min length	10	8	9	5	27
Max length	23	8	9	15	44
Avg length	14.86	8	9	8	31.36
Min #tokens	2	-	-	1	3
Max #tokens	4	-	-	2	5
Avg #tokens	2.38	-	-	1.13	3.86
Min value	-	3.657992	50.984194	-	-
Max value	-	4.050876	51.281777	-	-
Avg value[2]	-	3.756594	51.064419	-	-
Data type	str	num	num	str	str-num

Example

Let us consider the attributes from the source dataset in Table 3.1 and the target dataset in Table 3.2. The calculated properties of the subsets of the attribute domains from these datasets are presented in Table 3.3 and Table 3.4, respectively. Next, the similarities between these properties are calculated by Equation 3.2, e.g. for the attributes a^S = "Lon" and a^T = "Geo2" we obtain:

$$\begin{aligned}
&\text{MinLength:} && \text{simProp}(a^S, a^T) && = 1 - \tfrac{|8-8|}{|8|+|8|} = 1 \\
&\text{MaxLength:} && \text{simProp}(a^S, a^T) && = 1 - \tfrac{|8-8|}{|8|+|8|} = 1 \\
&\text{AvgLength:} && \text{simProp}(a^S, a^T) && = 1 - \tfrac{|8-8|}{|8|+|8|} = 1 \\
&\text{MinValue:} && \text{simProp}(a^S, a^T) && = 1 - \tfrac{|3.657992-3.657975|}{|3.657992|+|3.657975|} = 0,999998 \\
&\text{MaxValue:} && \text{simProp}(a^S, a^T) && = 1 - \tfrac{|4.050876-3.722015|}{|4.050876|+|3.722015|} = 0.957691 \\
&\text{AvgValue:} && \text{simProp}(a^S, a^T) && = 1 - \tfrac{|3.756594-3.701370|}{|3.756594|+|3.701370|} = 0.992595
\end{aligned} \tag{3.3}$$

Assuming that the threshold $P_V.thrStats$ is equal to 0.8, these attributes are considered as being coreferent because the similarity of all properties of these attribute domains exceeds 0.8. The same holds for the attribute pair "Lat" and "Geo1". Thus, these two attribute pairs determine two one-to-one matchings which are added to the set $M_V^{1:1}$ of one-to-one matchings. However, the other attribute pairs are not coreferent based on the statistical information, therefore they are further processed in the next step of our algorithm.

Table 3.4 Properties of attribute domains from the target dataset 3.2. *Num* means numerical datatype, *str* means alphabetic data type, and *str-num* means alphanumeric datatype.

Property	POI	Geo1	Geo2	Type	Street	City	ZipCode
Min length	12	9	8	5	15	4	7
Max length	21	9	8	10	20	17	7
Avg length	16.17	9	8	8.83	18.17	7.17	7
Min #tokens	1	-	-	1	1	1	2
Max #tokens	3	-	-	2	3	1	2
Avg #tokens	2.16	-	-	1.17	2.16	1	2
Min value	-	51.018938	3.657975	-	-	-	-
Max value	-	51.056465	3.722015	-	-	-	-
Avg value[3]	-	51.044901	3.701370	-	-	-	-
Data type	str	num	num	str	str-num	str	str-num

3.4.2 Step 2: Overlapping

The attributes a^S and a^T, whose statistical properties are not similar enough, are considered in this step. More specifically, a lexical comparison of the subsets

$dom'(a^S)$ and $dom'(a^T)$ of the attribute domains is conducted using soft strings comparison. For that purpose, the method compareDom is used which works as follows (lines 19-25 in Algorithm 3.2).

First, special characters that appear in the values of $dom'(a^S)$ and $dom'(a^T)$, i.e. dash, semicolon, dot, etc., are replaced by a space character, which results in strings of terms separated by space. This way, each attribute is described by a multiset of obtained terms (W_{a^S} and W_{a^T}, respectively). Next, the intersection I of these multisets is calculated according to formula (1.7). Thus, multiset I contains the common terms of W_{a^S} and W_{a^T}, which are assigned a PTV (1,0) and are the basis for further checking whether the particular attributes a^S and a^T are coreferent or not. Namely, these associated PTVs multiplied by the term multiplicity form a multiset $\tilde{P}$ which is used to construct a possibility distribution $\pi_{\mathbb{N}}$ (a fuzzy integer) introduced in Definition 5.

The fuzzy integer $\pi_{\mathbb{N}}$ of intersection I reflects the possibility that two attribute domains are coreferent. Hence, if $\pi_{\mathbb{N}}$ is greater than the threshold $\pi^{thr}_{(dom^S,dom^T)}$ with respect to the order relation of Definition 6, then the attributes $a_S \in A_S$ and $a_T \in A_T$ are considered to be potentially coreferent (candidate attributes, line 21 in Algorithm 3.2), and the established matching m between them is added to the set $M_V^{1:1}$ of matchings (lines 22-24 in Algorithm 3.2). The threshold $\pi^{thr}_{(dom^S,dom^T)}$ is a fuzzy integer and is dynamically calculated by the method getThr (lines 20 in Algorithm 3.2). This threshold depends on the particular attribute domains and the predefined parameter $P_V.thrOverlap$, which specifies the percentage of domain terms that overlap. More specifically, $\pi^{thr}_{(dom^S,dom^T)}$ is constructed from n PTVs (1,0), where n is calculated by the following equation:

$$n = \lfloor \min(|W_{a^S}|, |W_{a^T}|) \times P_V.thrOverlap \rfloor \tag{3.4}$$

However, if a fuzzy integer $m.\pi_{\mathbb{N}}$ of matching is not larger than the threshold $\pi^{thr}_{(dom^S,dom^T)}$, then the domains of the considered attribute a^S and a^T are analogously compared again, but the equalness relation, which decides on the coreference of the terms, is replaced by the low-level string comparison method proposed in [8]. This low-level comparison method estimates the possibility that two given terms are coreferent or not (see Section 2.3.3.2). Two terms are considered as coreferent if $\mu_{\tilde{p}}(F)$ of the resulting PTV is lower than $\mu_{\tilde{p}}(F)$ of the predefined threshold $P_V.\tilde{thr}$ (see Section 1.4.4).

Example

Let us consider the attribute a^S = "Address" from the source dataset in Table 3.1 and the attribute a^T = "City" from the target dataset in Table 3.2. Let us assume that these attributes have not been indicated as coreferent based on the analysis carried out in Step 1. After preprocessing the subset $dom'(a^S)$ of the attribute

domain, the resulting multiset W_{a^S} contains the terms: “Sint-Baafsplein” (multiplicity 2), “9000” (3), “Ghent” (4), “Hoegaarden” (1), “St-Denijs-Westrem” (1), “Gentstraat” (1), etc. Whereas, the multiset W_{a^T} of the subset $dom'(a^T)$ of the attribute domain consists of the terms: “Gent” (4), “Hoegaarden” (1) and “St-Denijs-Westrem” (1). Next, the intersection I of the multisets W_{a^S} and W_{a^T} is calculated which contains two terms, “Hoegaarden” and “St-Denijs-Westrem”, both with a multiplicity equal to 1 (the multiplicity of an element in the intersection of two multisets is the minimum of the multiplicities of that element in both multisets, see Section 1.4.2). Both returns are given an associated PTV (1,0). Thus, the fuzzy integer of this intersection is constructed from the multiset of PTVs $\{(1,0);(1,0)\}$ by Equation 1.30.

Figure 3.2 Fuzzy integers derived from the possibilistic truth values of the attribute matching (“Address”; “City”) based on the equality relation, the threshold $\pi^{thr}_{(dom^S,dom^T)}$ and the attribute matching (“Address”; “City”) based on the low-level string comparison method.

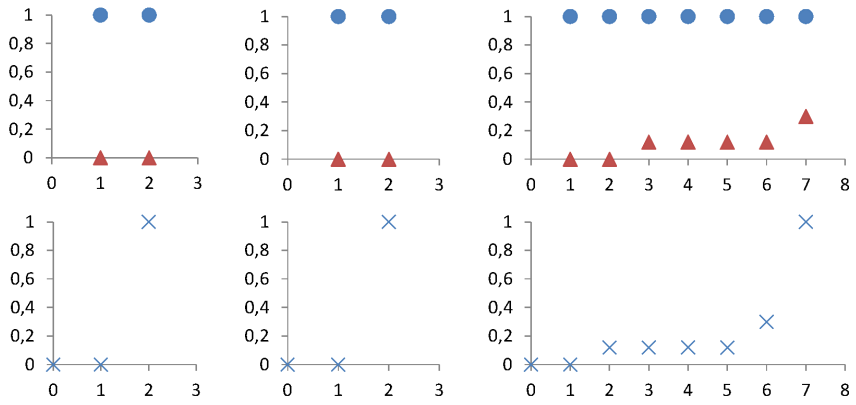


Figure 3.2 (the top left-most graph) shows the multiset of possibilistic truth values, where a *circle* denotes the possibility of T and a *triangle* denotes the possibility of F. The derived possibility distribution $\pi_{\mathbb{N}}$ (the fuzzy integer) is shown below the possibilistic truth values. The middle graph of Figure 3.2, in turn, shows the multiset of PTVs which are used to construct the threshold $\pi^{thr}_{(dom^S,dom^T)}$ which depends on the particular attribute domains and the predefined parameter $P_V.thrOverlap$ and is shown in the graph below the multiset of PTVs. Following the specification, the multiset of PTVs which is used to construct the threshold $\pi^{thr}_{(dom^S,dom^T)}$ consists of n PTVs equal to (1,0), where n is calculated by Equation 3.4 with $P_V.thrOverlap = 0.35$ and equals to $n = \lfloor \min(33,6) \times 0.35 \rfloor = 2$. The fuzzy integer $\pi_{\mathbb{N}}$ is not larger than $\pi^{thr}_{(dom^S,dom^T)}$ (w.r.t. Definition 6), because the 1-cuts of both fuzzy integers have the same supremum equal 2. So,

the domains of the considered attributes a^S and a^T are next compared using the low-level string comparison method with $\mu_{\tilde{p}}(F)$ = 0.5 as the predefined threshold $P_V.\tilde{thr}$. That comparison returns an intersection I, which consists of the following elements: ("Gentstraat","Gent") with an associated PTV (1,0.3) and multiplicity 1; ("Ghent","Gent"), (1,0.12), 4; ("St-Denijs-Westrem","St-Denijs-Westrem"), (1,0), 1; and ("Hoegaarden","Hoegaarden"), (1,0), 1. Figure 3.2 (the right-most top graph) shows the multiset of PTVs {(1,0); (1,0); (1,0.12); (1,0.12); (1,0.12); (1,0.12); (1,0.3)}, which are used by Equation 1.30 to construct a fuzzy integer $\pi_{\mathbb{N}}$ and is shown below the PTVs in Figure 3.2. Now, it turns out that, the fuzzy integer $\pi_{\mathbb{N}}$ is larger than the threshold $\pi^{thr}_{(dom^S,dom^T)}$ (w.r.t. Definition 6), because the 1-cut of the right-most fuzzy integer has a higher supremum, equal 7, than the middle fuzzy integer threshold, which has the supremum 2. This is the same fuzzy integer threshold as above because it depends on the same particular attribute domains - we consider the same attributes. Thus, the attributes "Address" and "City" are considered as being potentially coreferent and the established matching m between them is added to the set $M_V^{1:1}$ of matchings.

3.4.3 Step 3: Generalization

The last step of the vertical matching phase derives a one-to-many schema matching $M_V^{1:n}$ by using the method called getGeneralization based on one-to-one schema matching $M_V^{1:1}$ (line 28 in Algorithm 3.2). Afterwards, the vertical schema matching M_V is composed of the schema matching $M_V^{1:n}$ and $M_V^{1:1}$ (line 29 in Algorithm 3.2). The method getGeneralization is implemented by the Algorithm 3.3 which works as follows.

The input schema matching $M_V^{1:1}$, which is generated in steps 1 and 2 (Section 3.4.1 and 3.4.2, respectively), is the basis to generate a set $M'_{1:n}$ of one-to-many matchings by combining a number of one-to-one matchings which have the same attribute from A_S or A_T, i.e., a one-to-many matching has either the form: (line 1 in Algorithm 3.3):

$$(A, a_j^T), \text{ where } A \subseteq A_S \wedge |A| \geq 2 \wedge \forall a_i^S \in A \ \exists (a_i^S, a_j^T) \in M_V^{1:1} \tag{3.5}$$

or,

$$(a_i^S, A), \text{ where } A \subseteq A_T \wedge |A| \geq 2 \wedge \forall a_j^T \in A \ \exists (a_i^S, a_j^T) \in M_V^{1:1} \tag{3.6}$$

Afterwards, for each matching $m_{1:n} \in M'_{1:n}$, the *extended* domains are compared by the compareDom method (line 3 in Algorithm 3.3). This is done analogously as in the "Overlapping" step of Section 3.4.2. An extended domain is constructed by the method getDom and contains concatenated values of all attributes that are specified in the parameters, i.e., of all attributes forming the set A

in $m_{1:n}$ (cf. (3.5)-(3.6)). The values are concatenated one by one and separated with a white space into a new value which belongs to the extended domain.

Next, *alternative* matchings $M_{alt}^{1:1} \subseteq M_V^{1:1}$ are selected by the method getAlternatives (line 4 in Algorithm 3.3). An alternative matching $m_{1:1} \in M_{alt}^{1:1}$ for $m_{1:n}$ should have at least one attribute in common with the matching $m_{1:n} \in M'_{1:n}$, i.e., (a_k^S, a_l^T) is an alternative matching with respect to (A, a_j^T) if $a_k^S \in A$ or $a_l^T = a_j^T$, and similarly for (a_i^S, A). Finally, if the fuzzy integer $\pi_{\mathbb{N}}$ of the one-to-many matching $m_{1:n}$ is larger (w.r.t. Definition 6) than the fuzzy integer of any alternative matching $m_{1:1} \in M_{alt}^{1:1}$ (line 6 in Algorithm 3.3), then the matching $m_{1:n}$ is added to the schema matching $M_V^{1:n}$ (line 7 in Algorithm 3.3).

Algorithm 3.3 GENERALIZATIONALGORITHM

Require: Schema Matching $M_V^{1:1}$
Ensure: Schema Matching $M_V^{1:n}$
1: Schema Matching $M'_{1:n} \leftarrow$ getCombination($M_V^{1:1}$)
2: **for all** Matching $m_{1:n} \in M'_{1:n}$ **do**
3: $m_{1:n}.\pi_{\mathbb{N}} \leftarrow$ compareDom(getDom($m_{1:n}.A'_S$),getDom($m_{1:n}.A'_T$))
4: Schema Matching $M_{alt}^{1:1} \leftarrow$ getAlternatives($M_V^{1:1}$,$m_{1:n}$)
5: **for all** Matching $m_{1:1} \in M_{alt}^{1:1}$ **do**
6: **if** $m_{1:1}.\pi_{\mathbb{N}} \prec_{sup} m_{1:n}.\pi_{\mathbb{N}}$ **then**
7: $M_V^{1:n} \leftarrow M_V^{1:n} \cup m_{1:n}$
8: **break**
9: **end if**
10: **end for**
11: **end for**

Remark. *This generalization is specified for alphanumerical data, where numerical data are considered as character data. For numerical data more sophisticated concatenation method (such as aggregation or transformation function, e.g., a function which calculates average value) of values of all attributes that are specified in the matching parameters, i.e., of all attributes forming the set A in $m_{1:n}$ (cf.* (3.5)-(3.6)*), is required and it is out of the scope of this work.*

Example

Let us consider the one-to-many matching $m_{1:n} \in M'_{1:n}$ as a combination of the one-to-one matching $M_V^{1:1}$ in Table 3.5. Namely, $m_{1:n}$ establishes a matching of the attribute a^S = "Address" ($a^S \in A_S$) from the source dataset in Table 3.1 and the attributes A'_T = {"Street", "City", "ZipCode"} from the target dataset in Table 3.2 ($A'_T \subseteq A_T$). This matching is derived as a combination of the one-to-one (candidate, alternative) matchings 7, 8 and 9 of Table 3.5.

First, the extended domains are constructed by using the method getDom.

Table 3.5 Example of the vertical schema matching $M_V^{1:1}$.

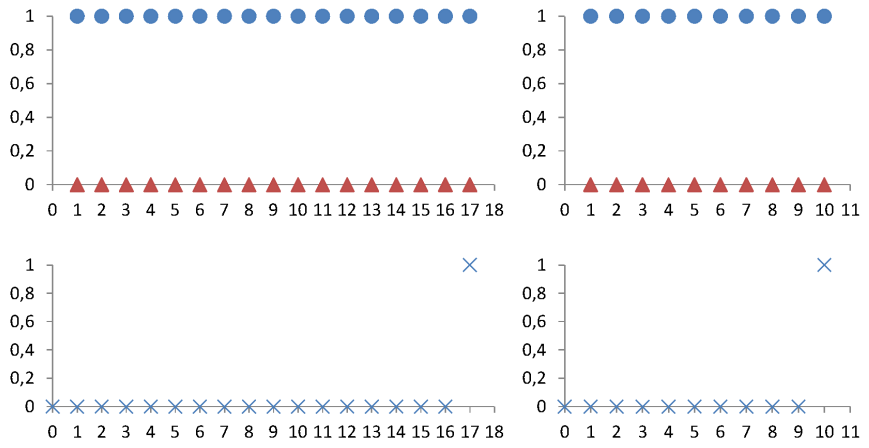
Matching m	**Source**	**Target**
1	Name	Type
2	Name	POI
3	Lon.	Geo2
4	Lat.	Geo1
5	Category	Type
6	Key	Id
7	Address	ZipCode
8	Address	Street
9	Address	City

These domains contain the values of all attributes forming the set A in a one-to-many matching; cf. (3.5)-(3.6). The extended domain $dom_{ext}(A'_S)$ = $dom(a^S)$ = dom(“Address”) contains the values: “Sint-Baafsplein, 9000 Ghent”, “Grotestraat 91, 7471 BL Goor”, “Jan Breydelstraat 35, 9000, Ghent”, etc. The extended domain $dom_{ext}(A'_T)$ = $dom_{ext}(\{$“Street”, “City”, “ZipCode”$\})$ contains the concatenated values: “Emile Braunplein Gent 9000 BE, “Sint-Baafsplein Gent 9000 BE”, “Jan Breydelstraat 35 Gent 9000 BE”, etc. Both extended domains $dom_{ext}(A'_S)$ and $dom_{ext}(A'_T)$ are compared by the compareDom method just as in the “Overlapping” step in Section 3.4.2. More specifically, after preprocessing the attribute domains, the multiset $W_{A'_S} = W_{a^S}$ contains the terms: “Sint-Baafsplein” (multiplicity 2), “9000” (3), “Ghent” (4), “Hoegaarden” (1), “St-Denijs-Westrem” (1), “Gentstraat” (1), etc. The multiset $W_{A'_T}$ contains the terms: “BE” (6), “Gent” (4), “9000” (4) “Hoegaarden” (1), “St-Denijs-Westrem” (1), “Sint-Baafsplein” (1), etc. Next, the intersection I of these multisets is determined which contains the terms: “Kleine” with associated PTV (1,0) and multiplicity 1; “Gentstraat”, (1,0), (1); “24”, (1,0), (1); “5”, (1,0), (1); “69”, (1,0), (1); “3320”, (1,0), (1); “9051”, (1,0), (1); “Schouwburgstraat”, (1,0), (1); “9000”, (1,0), (4); “Stoopkensstraat”, (1,0), (1); “Sint-Baafsplein”, (1,0), (1); “Jan”, (1,0), (1), “St-Denijs-Westrem”, (1,0), (1); “Hoegaarden”, (1,0), (1). All associated PTVs are (1,0) because all the compared terms are equal. Next, the fuzzy integer expressing the cardinality of the multiset of matching terms is constructed from the resulting multiset of PTVs multiplied by the term multiplicity by Equation 1.30.

Figure 3.3 shows the set of possibilistic truth values, where a *circle* denotes the possibility of T and *triangle* denotes the possibility of F. The derived possibility distribution $\pi_{\mathbb{N}}$ (the fuzzy integer) is shown below the possibilistic truth values.

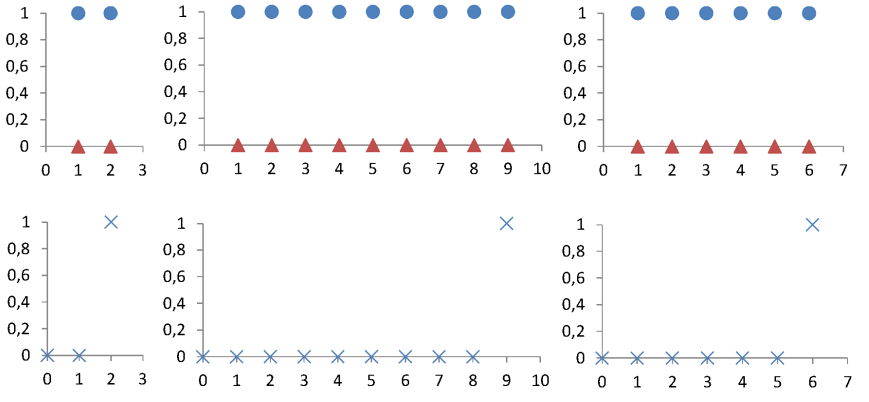
Next, *alternative* matchings $M_{alt}^{1:1} \subseteq M_V^{1:1}$ are selected by the method getAlternatives, i.e. one-to-one matchings that have at least one attribute in common with the constructed one-to-many matching $m_{1:n}$. Namely, (“Address”; “ZipCode”), (“Address”; “City”), and (“Address”; “Street”). Finally, it turns out that

Figure 3.3 Fuzzy integers derived from possibilistic truth values of the attribute matching (“Address”; “Street”, “City”, “ZipCode”).



the fuzzy integer $\pi_{\mathbb{N}}$ related to the one-to-many matching $m_{1:n}$ is larger than the fuzzy integer (w.r.t. Definition 6) related to each alternative matching in Figure 3.4, because the 1-cut of the fuzzy integer resulting from concatenation $m_{1:n}$ has a higher supremum (17 in Figure 3.3) than the fuzzy integers of the alternative matchings (2, 9, 6, respectively in Figure 3.4). So, the matching $m_{1:n}$ is added to the schema matching $M_V^{1:n}$.

Figure 3.4 Fuzzy integers derived from possibilistic truth values of the alternative attribute matchings for the attribute “Address”: (“Address”; “City”), (“Address”; “Street”) and (“Address”; “ZipCode”).



Finally, the the union of sets of matchings $M_V^{1:1}$ and $M_V^{1:n}$ forms the final set of candidate matchings M_V, which is the basis to detect coreferent tuples and to

establish the final matching between corresponding attributes in the next phase.

3.5 Phase II: Horizontal matching

In this phase, the candidate vertical schema matching M_V from the previous phase is used to establish the horizontal schema matching M by the method getHorizontalMatchings (line 2 in Algorithm 3.1). More specifically, in step 1 of this phase the vertical schema matching M_V is used to efficiently detect coreferent tuples across heterogeneous data sources (Section 3.5.1). This significantly reduces the number of comparisons and the complexity of the approach and in turn is the basis for generating the final schema matching in the second step of this phase (Section 3.5.2). The following subsections describe both steps of the horizontal matching phase.

3.5.1 Step 1: Coreferent tuples detection

The coreferent tuples detection Algorithm 3.4 for schema matching searches for the n-most coreferent tuple pairs D across tuples in the source dataset S and the target dataset T using the candidate vertical schema matching M_V as follows. First, each tuple from the source is compared with each tuple from the target. More precisely, the values of the corresponding attributes, which are matched by M_V, are compared by the method compareTuples (line 3 in Algorithm 3.4) which inter alia calculates the possibility that two given tuples are coreferent (expressed by a PTV denoted as $\tilde{d}$) and returns a pair of coreferent tuples d. The details of this comparison are presented in Algorithm 3.5 and described in the next Paragraph 3.5.1.1. Next, coreferent tuple pair d is added to the set D of coreferent tuple pairs (line 4 in Algorithm 3.4). Finally, the detected coreferent tuple pairs $d \in D$ are sorted by $\tilde{d}$ using Equation 1.17 and the $P_H.n$ most coreferent tuple pairs are the result of this algorithm (line 7 and 8 in Algorithm 3.4, respectively). Using a fixed threshold on the matching degree would be unreasonable because the (un)certainty of tuples coreference varies along with the number of corresponding attributes [19], i.e. if only a few attributes are truly coreferent then the certainty will be low. Thus, instead, our method ranks coreferent tuple pairs by their PTVs and gets the n-most coreferent tuple pairs. It has to be clear that the goal is not to detect all coreferent tuples. These coreferent tuple pairs serve as the basis to establish horizontal schema matching, what is discussed in the next Section 3.5.2.

3.5.1.1 Tuples comparison

A comparison of two tuples is conducted by Algorithm 3.5 and works as follows. First, the input tuples t^S and t^T from the source dataset and the target dataset, respectively, form a pair of candidate coreferent tuples $(d.t^S, d.t^T)$ (line 1 and 2 in Algorithm 3.5, respectively). Second, a comparison of attribute values for each

Algorithm 3.4 COREFERENTTUPLEDETECTIONALGORITHM

Require: Dataset S, Dataset T, Schema Matching M_V, Parameters P_H
Ensure: n-most Coreferent Tuple Pairs D
1: **for all** Tuple $t^S \in S$ **do**
2: **for all** Tuple $t^T \in T$ **do**
3: Coreferent tuple pair $d \leftarrow$ compareTuples(t^S,t^T,M_V,P_H)
4: $D \leftarrow D \cup d$
5: **end for**
6: **end for**
7: $D \leftarrow$ sort(D)
8: $D \leftarrow$ getMostCoreferent($P_H.n$,D)

matching $m \in M_V$ computed in the previous step works as follows. An initial matching $d.m$ is initialized as a copy of a matching m (line 4 in Algorithm 3.5). Next, for each attribute(s) $m.A'_S$ ($m.A'_T$) of a candidate matching $m \in M_V$ the value(s) v^S or v^T from tuples $d.t^S$ and $d.t^T$ are respectively extracted (lines 5 and 6 in Algorithm 3.5). In case of 1:n matching of attributes (see Section 3.4.3), the extracted values can be vectors of values $v^S[]$ or $v^T[]$ whose coordinates are concatenated into v^S or v^T before they are compared. Afterwards, the extracted values v^S and v^T are compared using a data type-specific method which estimates the possibility $d.\tilde{m}$ that two given values are coreferent (line 7 in Algorithm 3.5). More precisely, a numerical and an alphanumerical matchers are considered as follows.

Numerical matcher. Numerical values are compared by a method which is based on the difference ($diff$) of the considered values and a difference threshold ($P_H.thrDiff$). The difference threshold is a real number which defines the maximum allowed difference of values, and depends on the range of values of a particular attribute and the predefined parameter $P_H.thrNum$, which specifies the percentage of difference for average range of the considered attributes $m.A'_S$ and $m.A'_T$ (1:n attribute matching does not apply for numerical data, thus, $m.A'_S = a^S$ and $m.A'_T = a^T$). More specifically, $P_H.thrDiff$ is calculated by the following equation:

$$P_H.thrDiff = \left\lfloor \frac{\text{range}(dom'(a^S)) + \text{range}(dom'(a^T))}{2} \times P_H.thrNum \right\rfloor \tag{3.7}$$

where range is a difference between the maximum and minimum value of $dom'(a^S)$ or $dom'(a^T)$ from the source or the target. If the difference $diff$ between values is smaller than the difference threshold $P_H.thrDiff$, then the possibility that a proposition p stating that the two values are coreferent is true ($\mu_{\tilde{p}}(T)$) equals 1, and the possibility that p is false ($\mu_{\tilde{p}}(F)$) is a fraction of the difference $diff$ and

the difference threshold $P_H.thrDiff$; otherwise, a lack of coreference is declared, i.e. $\mu_{\tilde{p}}(T) = 0$ and $\mu_{\tilde{p}}(F) = 1$.

Alphanumerical matcher. Alphanumerical values are transformed into sets of substrings which are in most cases separate words. The string is split at the position of a white space, comma, dot or other special character. The usefulness of this approach follows from the fact that character-based methods are typically not well suited for longer strings. Next, the substrings are compared with one another by the low-level string comparison method [8] (see Section 2.3.3.2). This gives a comparison matrix of PTVs which is used to establish a mapping between substrings (see Section 2.3.3.3). The selected PTVs are aggregated by the Sugeno integral [37, 38, 94]. Aggregation results in a single PTV which reflects the possibility that two given values are coreferent (see Section 1.4.4). More specifically, the aggregation operator for the comparison of two values, v^S and v^T, is defined by the Sugeno integral for PTVs, where:

- $P = \{p_i\}$ is a set of propositions stating coreference of pairs of substrings,
- $\tilde{P}$ is the set of selected PTVs corresponding to the above-mentioned propositions representing the uncertainty about their truth values computed by the low-level string comparison method,
- the fuzzy measure γ^T is defined by:

$$\gamma^T(Q) = \sum_{j=1}^{k} w_j, \qquad Q \subseteq P, Q = \{p_1, \ldots, p_k\} \tag{3.8}$$

  where w_j is the weight of the jth pair (s_S^j,s_T^j) of substrings, computed by:

$$w_j = \frac{1}{|P|} \tag{3.9}$$

- the fuzzy measure γ^F is defined by

$$\gamma^F(Q) = \begin{cases} 1 & \text{if } Q = P \\ 0 & \text{otherwise} \end{cases} \tag{3.10}$$

  what is implied by condition (1.26) is that for each Q which is a subset of *P* (see Section 2.3.3.4).

Moreover, pairs of substrings with the selected PTVs are grouped into two sets. The first set contains substrings which have an associated PTV that is larger than $P_H.\tilde{thr}$ w.r.t. Equation 1.17, and are called coreferent tokens. The second set contains substrings which have an associated PTV that is not larger than $P_H.\tilde{thr}$ and are called non-coreferent tokens. These two sets are the basis for deriving the

type of matching. The type $type$ of matching $d.m$ (see Section 3.3.1) is specified based on the number of coreferent and non-coreferent tokens (substrings) of the values v^S and v^T by a method which is presented in the next Section 3.5.1.2 (line 8 in Algorithm 3.5). Next, the matching $d.m$ is added to the set $d.M_V$ of matchings for the coreferent tuples pair d.

Finally, the set $d.M_V$ contains matchings $d.m$ of coreferent attributes A'_S and A'_T for the particular tuples $d.t^S$ and $d.t^T$. Each matching $d.m \in d.M_V$ has an associated PTV ($d.\tilde{m}$), which expresses the possibility that attributes A'_S and A'_T are coreferent for the particular tuples $d.t^S$ and $d.t^T$ based on their values, and is the basis for further checking whether the particular tuples $d.t^S$ and $d.t^T$ are coreferent or not. Namely, these associated PTVs form a multiset $d.\tilde{M}_V$ and are aggregated by the Sugeno integral [37, 38, 94]. The aggregation returns a single PTV $\tilde{d}$ which reflects the possibility that two given tuples are coreferent (line 11 in Algorithm 3.5). More specifically, the aggregation operator for the comparison of two tuples, $d.t^S$ and $d.t^T$, is defined by the Sugeno integral for PTVs, where:

- $P = \{p_i\}$ is a set of propositions stating coreference of attributes A'_S and A'_T for the particular tuples $d.t^S$ and $d.t^T$, represented by $d.M_V = \{d.m_i\}$,
- $\tilde{P}$ is the set of PTVs corresponding to the above-mentioned propositions, represented by $d.\tilde{M}_V$,
- the fuzzy measure γ^T is defined by:

$$\gamma^T(Q) = \sum_{j=1}^{k} w_j, \qquad Q \subseteq P, Q = \{p_1, \ldots, p_k\} \tag{3.11}$$

  where w_j is the weight of the jth pair (A'_S,A'_T) of attributes for the particular tuples, computed by:

$$w_j = \forall d.\tilde{m} \in d.\tilde{M}_V : w_{d.\tilde{m}} = \frac{1}{|d.M_V|}. \tag{3.12}$$

  These weights are equal for each PTV and depend on the number of matchings.

- the fuzzy measure γ^F is defined by

$$\gamma^F(Q) = \begin{cases} 1 & \text{if } Q = P \\ 0 & \text{otherwise} \end{cases} \tag{3.13}$$

  what is implied by condition (1.26) is that for each Q which is a subset of P (see Section 2.3.3.4).

Algorithm 3.5 COMPARETUPLESALGORITHM

Require: Tuple t^S, Tuple t^T , Schema Matching M_V, Parameters P_H
Ensure: Coreferent tuple pair d
1: $d.t^S \leftarrow t^S$
2: $d.t^T \leftarrow t^T$
3: **for all** Matching $m \in M_V$ **do**
4: Matching $d.m \leftarrow m$
5: var $v^S \leftarrow d.t^S[m.A'_S]$
6: var $v^T \leftarrow d.t^T[m.A'_T]$
7: $d.\tilde{m} \leftarrow$ compare(v^S, v^T, $P_H.\tilde{thr}$, $P_H.thrNum$)
8: $d.m.type \leftarrow$ mType(v^S, v^T)
9: $d.M_V \leftarrow d.M_V \cup d.m$
10: **end for**
11: $\tilde{d} \leftarrow$ aggregate($d.\tilde{M}_V$)

3.5.1.2 Matching type of tuples

The matching types, which are defined in Section 3.3.1, depend on the factors $ratioS$ and $ratioT$. These factors represent the completeness of each matching $d.m \in d.M_V$ for the particular tuples t^S and t^T and are based on the number of coreferent tokens (substrings) of compared values $v^S \in t^S$ and $v^T \in t^T$. The factor $ratioS$ ($ratioT$) is the fraction of the number of coreferent tokens over the number of tokens of the value v^S (v^T, respectively). Thus, the following conditions have to be considered:

- if $ratioS = 0$ and $ratioT = 0$ then $m \in M$ is an "unknown" matching
- if $ratioS = 1$ and $ratioT = 1$ then $m \in M$ is a "full" matching
- if $ratioS = 1$ and $ratioT \neq 1$ then $m \in M$ is a "source is part of target" matching
- if $ratioS \neq 1$ and $ratioT = 1$ then $m \in M$ is a "target is part of source" matching
- if $ratioS \neq 1$ and $ratioT \neq 1$ then $m \in M$ is a "a common part" matching

Example

Let us consider coreferent tuple pairs detection for the following case. The input is the vertical matching M_V of Table 3.6, which is the basis for detecting coreferent tuple pairs across the source and target datasets, of Table 3.1 and Table 3.2, respectively. Each tuple from the source dataset is compared to each tuple in the target dataset which results in the set D of coreferent tuple pairs. For example, let

us consider that tuple t^S in row 2 in Table 3.1 is compared to tuple t^T in row 2 in Table 3.2. First, the values of the matched attributes (by M_V) are compared in sequence, e.g. the value "Saint Bavo" of the attribute "Name" in t^S is compared by the alphanumerical matcher to the following values in t^T w.r.t. the matchings m_1, m_2 and m_3 in Table 3.6: "Church" (of the attribute "Type"), "Sint-Bavokerk" (of the attribute "POI") and "Church Sint-Bavokerk" (of the concatenation of the attributes "Type" and "POI"). This comparison results in $d.M_V$ which contains all the attributes matchings $m \in M_V$, namely $d.m \in d.M_V$ (with associated PTVs $d.\tilde{m}$ which form a multiset $d.\tilde{M}_V$) for the particular tuples t^S and t^T, e.g. $d.\tilde{m}_1$ = (0,1), $d.\tilde{m}_2$ = (1,0.12), $d.\tilde{m}_3$ = (1,0.16), etc. Next, the matching type for each $d.m \in d.M_V$ for particular tuples is derived: $d.m_1.type$ is an "unknown" matching (no common tokens), $d.m_2.type$ is a "full" matching (all tokens are common), $d.m_3.type$ is a "source is part of target" matching (all tokens from the source are common, but not all from the target), etc. Next, the PTVs in $d.\tilde{M}_V$ are aggregated by the Sugeno integral with equal weights $w = 1/15$ (calculated by Equation 3.12). This results in a single PTV $\tilde{d}$ that equals (1,0.5) which reflects the possibility that the two given tuples t^S and t^T are coreferent.

Finally, the detected coreferent tuple pairs D are sorted by $\tilde{d}$, and the $P_H.n$ (in our case 3) most coreferent tuple pairs are returned. Namely, the pairs of tuples from rows 2, 4 and 5 of Tables 3.1 and 3.2 are returned as the 3 most coreferent tuple pairs and are used to establish the final schema matching, what is discussed in the next section.

Table 3.6 Example of the vertical schema matching M_V.

Matching m	**Source**	**Target**
m_1	Name	Type
m_2	Name	POI
m_3	Name	Type, POI
m_4	Name, Category	Type
m_5	Lon.	Geo2
m_6	Lat.	Geo1
m_7	Category	Type
m_8	Key	Id
m_9	Address	ZipCode
m_{10}	Address	Street
m_{11}	Address	City
m_{12}	Address	ZipCode, Street
m_{13}	Address	ZipCode, City
m_{14}	Address	Street, City
m_{15}	Address	ZipCode, Street, City

3.5.2 Step 2: Schema matching

The n most coreferent tuples pairs of D detected in the previous step and the vertical schema matching M_V established in the first phase of our approach are used to infer the final horizontal schema matching M. Our novel approach is implemented by Algorithm 3.6, which works as follows. First of all, for each matching $m \in M_V$, the cardinality $card_m$ of this matching is calculated (lines 2-6 in Algorithm 3.6). This cardinality is the number of coreferent tuple pairs whose values of attributes $m.A'_S$ and $m.A'_T$ are coreferent. Hereby coreference is considered if $\mu_{\tilde{p}}(F)$ of $d.\tilde{m}$ is lower than $\mu_{\tilde{p}}(F)$ of the predefined threshold $P_H.thr\tilde{D}up$ = (0.5,0.5) w.r.t. Equation 1.17.

Next, if a particular matching m returns most of the coreferent tuple pairs (line 7 in Algorithm 3.6), i.e. $m.card_m/|D|$ is greater than the predefined threshold $P_H.thrMajority$, then m is added to the final schema matching M (line 8 in Algorithm 3.6). Next, the propositions evaluated using PTVs of the matching m across all coreferent tuple pairs of D (i.e., $\forall d \in D : d.\tilde{m}$) are aggregated by the Sugeno integral. This results in a possibility degree $\tilde{m}$ (PTV), which expresses the uncertainty of that matching $m \in M$ (line 9 in Algorithm 3.6) [37, 38, 94] (see Section 1.4.4) and is used to resolve schema matching conflicts in the next phase which is described in Section 3.6. More specifically, the aggregation operator for matching $m \in M$ is defined by the Sugeno integral for PTVs, where:

- $P = \{p_i\}$ is a set of propositions stating coreference of attributes $m.A'_S$ and $m.A'_T$ across all coreferent tuple pairs of D, represented by $D[m] = \{d_i.m\}$,
- $\tilde{P}$ is the set of PTVs corresponding to the above-mentioned propositions, represented by $\tilde{D}[m] = \{d_i.\tilde{m}\}$,
- the fuzzy measure γ^T is defined by:

$$\gamma^T(Q) = \sum_{j=1}^{k} w_j, \qquad Q \subseteq P, Q = \{p_1, \ldots, p_k\} \tag{3.14}$$

  where w_j is the weight of the jth duplicate d_j, computed by:

$$w_j = \frac{1}{|D|}. \tag{3.15}$$

  These weights are equal for each PTV $d_j.\tilde{m}$ and depend on the number of coreferent tuple pairs.

- the fuzzy measure γ^F is defined by

$$\gamma^F(Q) = \begin{cases} 1 & \text{if } Q = P \\ 0 & \text{otherwise} \end{cases} \tag{3.16}$$

what is implied by condition (1.26) is that for each Q which is a subset of P (see Section 2.3.3.4).

Finally, the matching types of the matching $m \in M$ across all coreferent tuple pairs D are unified by the method unifyType in line 10 in Algorithm 3.6. This method returns for each $m \in M$ the most popular matching type across all coreferent tuple pairs D. In the case of indistinguishable matching types, i.e. if maximum frequency of matching type for matching $m \in M_V$ over all coreferent tuple pairs of D is not unique, the matching type with the predefined lowest coverage level is selected (see Section 3.3.1). For example, if full matching (with coverage level 1) and inclusion matching (with coverage level 0.5) types are specified for the same number of coreferent tuple pairs then inclusion matching type is selected. The unified matching types are used to resolve schema matching conflicts in the next phase in Section 3.6.

Algorithm 3.6 HORIZONTALMATCHINGALGORITHM

Require: Coreferent tuple pairs D, Schema Matching M_V, Parameters P_H
Ensure: Schema Matching M
1: **for all** Matching $m \in M_V$ **do**
2: **for all** Coreferent tuple pair $d \in D$ **do**
3: **if** $d.\tilde{m} > P_H.thr\tilde{D}up$ **then**
4: $card_m \leftarrow card_m + 1$
5: **end if**
6: **end for**
7: **if** $card_m/|D| > P_H.thrMajority$ **then**
8: $M \leftarrow M \cup m$
9: $\tilde{m} \leftarrow$ aggregate($\tilde{D}[m]$)
10: $m.type \leftarrow$ unifyType($D[m].type$)
11: **end if**
12: **end for**

Example

Let us consider the coreferent tuple pairs in D which were detected in the previous step. The set D of coreferent tuples pairs consists of 3 pairs, each composed of rows 2, 4, 5 from Tables 3.1 and 3.2. The matching cardinality $card_m$ is calculated for each matching $m \in M_V$ in Table 3.6. For example, the $card_m$ of the matching $m_{10} = \{$"Address"; "Street"$\}$ equals 2 because only the attribute values of two tuple pairs are coreferent for the threshold $P_H.thr\tilde{D}up$ equal to (0.5, 0.5). More specifically, the value "Stoopkensstraat 24, 3320 Hoegaarden" is similar to "Stoopkensstraat 24", and "Kleine Gentstraat 69, 9051 St-Denijs-Westrem" is similar to "Kleine Gentstraat 69". The certainty as to their similairty is expressed for both of

them by a PTV (1,0.33). The similarity of values "Sint-Baafsplein, 9000 Ghent" and "Sint-Baafsplein" is expressed by a PTV (1,0.5), but this does not exceed the threshold. In contrast, $card_m$ of the matching m_{15} = {"Address"; "Street", "City", "ZipCode"} is equal to 3, because the attribute values of all coreferent tuple pairs are coreferent. This confirms that the concatenation of attributes makes sense.

Next, if $card_m$ of m satisfies the majority condition in line 7 of Algorithm 3.6, then the matching m is added to M. The predefined threshold $P_H.thrMajority$ = 0.3 and $|D| = 3$, thus if $card_m$ of m is greater than 0.9, then m is considered as a matching in M. This means that M contains only matchings which are confirmed by at least one coreferent tuple pair. Matchings m_4, m_9 and m_{11} in Table 3.6 are not included in M because they do not satisfy this condition. Next, the PTVs for matching m across all coreferent tuple pairs D are aggregated by the Sugeno integral. For the matching m_{10}, the PTVs (1,0.33), (1,0.5), (1,0.33) are aggregated with equal weights $w = 1/|D| = 1/3$ to a single PTV equal to (1,0.33). This PTV reflects the possibility that the attributes "Address" and "Street" are coreferent. Finally, the matching types of the matching $m \in M$ across all coreferent tuple pairs D are unified by the method unifyType. For the matching m_{10}, the unified matching type is "t is part of s" (target is part of source) because the matching type of all considered coreferent tuple pairs is "t is part of s".

This gives us the schema matching M in Table 3.7 with conflicts, i.e. for some of the attributes more than one matching is established, e.g. m_{10}-m_{15} in Table 3.7. These conflicts are resolved in the next phase "Conflict resolution".

Table 3.7 Example of the schema matching M with conflicts.

Matching m	**Source**	**Target**	$card_m$	$type_m$	$\tilde{m}$
m_1	Name	Type	1	t is part of s	(1,0.67)
m_2	Name	POI	1	has a common part	(1,0.5)
m_3	Name	Type, POI	1	has a common part	(1,0.6)
m_5	Lon	Geo2	2	full matching	(1,0.33)
m_6	Lat	Geo1	2	full matching	(1,0.33)
m_7	Category	Type	1	full matching	(1,0.67)
m_8	Key	id	3	full matching	(1,0)
m_{10}	Address	Street	2	t is part of s	(1,0.33)
m_{12}	Address	ZipCode, Street	3	has a common part	(1,0.03)
m_{13}	Address	ZipCode, City	1	has a common part	(1,0.5)
m_{14}	Address	Street, City	3	t is part of s	(1,0.12)
m_{15}	Address	ZipCode, Street, City	3	s is part of t	(1,0.12)

3.6 Phase III: Conflict resolution

The goal of the last optional phase of our novel schema matching approach is to resolve *conflicts*. The schema matching M contains conflicts if there exists more

than one matching $m \in M$ for any attribute $a^S \in A_S$ or $a^T \in A_T$. In other words, the schema matching is not unique if there exists alternative matchings for a particular attribute a^S or a^T, which are called conflicting attributes. These conflicts are resolved by the method resolveConflicts in line 3 of Algorithm 3.1, which is based on the following a set of heuristic rules. The general motivation is that repeated matchings across many coreferent tuples, matchings donated with large PTVs (certain matchings), matchings which represent the same concepts (high coverage level), and one-to-many matchings are preferable.

The rule execution order depends on the user to fine-tune the method. The proposed order below is only one possible combination of these rules to show the application of them to establish correct unique schema matching. Moreover, the conflict resolution is an optional phase, thus, this phase can be omitted which may give alternative matchings for considered schema attributes.

3.6.1 Rule I

First, rule I is applied. It states that a matching which has larger cardinality, i.e., is repeated by the larger number of coreferent tuples, than alternative matchings is preferable and it works as follows.

Let $M_a^{alt} \subseteq M$ be the set of alternative matchings for the conflicting attribute $a \in A_S$ or $a \in A_T$, $card_m$ be the largest cardinality of the cardinalities of all $m_{alt} \in M_a^{alt}$, and the parameter $P_C.w$ is an integer number to relax the condition. Then each alternative matching $m_{alt} \in M_a^{alt}$ is removed if satisfies the following equation:

$$m_{alt}.card_m < card_m - P_C.w \tag{3.17}$$

For instance, let us consider the schema matching M in Table 3.7 which is not unique. The attributes "Name" and "Address" from the source, and "Type", "POI", "Street", "City" and "ZipCode" from the target are matched by more than one matching $m \in M$ and are hence conflicting attributes. According to rule I and $P_C.w$ = 0, only the alternative matchings m_{10} and m_{13} of the conflicting attribute "Address" from the source are removed, because their $card_m$ is equal to 2 and 1, respectively, and is smaller than the largest cardinality (3) of the other alternative matchings.

3.6.2 Rule II

Next, if M is still not unique, then rule II is applied. It states that matching which has more data in common (covers more the same concept) than alternative matchings is preferable and it works as follows.

Let $M_a^{alt} \subseteq M$ be a set of alternative matchings for the conflicting attribute $a \in A_S$ or $a \in A_T$, and let $type_m$ be the matching type with the highest coverage level of the matching types of all $m_{alt} \in M_a^{alt}$ (see definition of matching

types and their predefined coverage levels in Section 3.3.1). Then each alternative matching $m_{alt} \in M_a^{alt}$ which is not assigned the matching type $type_m$ is removed.

For instance, consider the schema matching M in Table 3.7 after the application of rule I. According to rule II, the alternative matchings m_1 and m_3 of the conflicting attribute "Type" from the target are removed because their matching type "target is part of source" and "has a common part" have a lower coverage level, (0.5) and (0.3), respectively, than the alternative matching m_7 which is a "full matching" with associated coverage level (1.0). Analogously, the alternative matching m_{12} is removed.

3.6.3 Rule III

Next, if the matching M is still not unique, then rule III is applied which is based on the (un)certainty of a matching. This means that a matching which is donated with the larger PTV than alternative matchings is preferable and it works as follows.

Let $M_a^{alt} \subseteq M$ be a set of alternative matchings for the conflicting attribute $a \in A_S$ or $a \in A_T$, and $\tilde{m}$ is the largest PTV of the PTVs of all $m_{alt} \in M_a^{alt}$. Then each alternative matching $m_{alt} \in M_a^{alt}$ which is not donated with the largest PTV $\tilde{m}$ is removed.

For instance, consider schema matching M in Table 3.7 after the application of rule II. According to rule III, no alternative matching is removed because the alternative matchings m_{14} and m_{15} are assigned a PTV $(1, 0.12)$.

3.6.4 Rule IV

Finally, if M is still not unique, then rule IV is applied which considers two cases.

Let $M_a^{alt} \subseteq M$ be a set of alternative matchings for the conflicting attribute $a \in A_S$ or $a \in A_T$, and $card_l^l$ ($card_s^l$) is the largest local cardinality (the number of matched attributes) (the smallest local cardinality, respectively) of all local cardinalities $card_m^l$ of all $m_{alt} \in M_a^{alt}$ (see Section 3.3.1). On the one hand, if each $m_{alt} \in M_a^{alt}$ is not a "full" matching, then each alternative matching $m_{alt} \in M_a^{alt}$ which is not assigned the largest local cardinality $card_l^l$ is removed. On the other hand, if each $m_{alt} \in M_a^{alt}$ is a full matching, then each alternative matching $m_{alt} \in M_a^{alt}$ which is not assigned the smallest local cardinality $card_s^l$ is removed.

The idea behind this resolution is simple. If alternative matchings are "full" matchings then it might be redundant to match more attributes, and vice versa. If alternative matchings are not "full" matchings then it might be desirable to match more attributes.

For instance, consider the schema matching M in Table 3.7 after the application of rule III. According to rule IV, the alternative matching m_{14} of the conflict-

ing attribute "Address" from the source is removed because its local cardinality $card_m^l$ is equal to 2 and is different from the largest local cardinality (3) of the alternative matching m_{15}, and both matchings are not "full" matchings.

The final schema matching M without conflicts is presented in Table 3.8.

Table 3.8 Example of the final schema matching M without conflicts.

Matching m	**Source**	**Target**	$card_m$	$type_m$	$\tilde{m}$
1	Name	POI	1	has a common part	(1,0.5)
2	Lon	Geo2	2	full matching	(1,0.33)
3	Lat	Geo1	2	full matching	(1,0.33)
4	Category	Type	1	full matching	(1,0.67)
5	Key	id	3	full matching	(1,0)
6	Address	Street, City, ZipCode	3	s is part of t	(1,0.12)

Moreover, if there are still some conflicts, it may mean that the alternative matchings are equivalent and the particular attribute(s) in the source have more than one corresponding attribute(s) in the target.

3.7 Evaluation and discussion

In this section we describe an experimental evaluation of our method which shows the influence of the parameters and the benefits of using content data (compared to schema information-only-based methods). Moreover, our technique is compared to DUMAS [19], an approach which uses duplicates and information retrieval techniques to establish a schema matching.

3.7.1 Datasets

To illustrate the proposed approach we consider different real-world datasets, respectively, containing information about 'compact discs', 'restaurants' and 'points of interest'.

Compact disc (CD) data are contained in two datasets which are defined in two schemas (also being used in Chapter 2[4]). As a reminder, the first schema is extracted from FreeDB[5] and consists of 8 leaf schema elements (here called attributes). The second schema is extracted from Discogs[6] and consists of 24 leaf schema elements, of which 6 have been identified manually as being coreferent with the FreeDB schema elements. The truly coreferent schema elements are presented in Table 3.9 and act as the ground truth for our experiments. The FreeDB

[4]The schemas are slightly different because the schema of extracted data has been changed since Dec 12, 2011

[5]FreeDB, http://www.freedb.org/

[6]Discogs, http://www.discogs.com/data/

dataset contains 124 tuples which are extracted from the CD dataset[7], while the Discogs dataset contains 132 tuples which are extracted from Discogs[8]. The number of coreferent tuples in these datasets is equal to 33, which are detected manually.

Table 3.9 True coreferent attribute (schema elements) paths of datasets.

Discogs vs FreeDB	R1.Fodor vs R1.Zagat	R2.Fodor vs R2.Zagat	RouteYou (R) vs Google (G)
/discs/disc/id	Fodor/id	Fodor/id	R/id
/cddb/disc/did	Zagat/id	Zagat/id	G/id
/discs/disc/artists/name	Fodor/name	Fodor/name	R/internalName
/cddb/disc/artist	Zagat/name	Zagat/name	G/name,G/vicinity
/discs/disc/title	Fodor/street	Fodor/street,Fodor/city	R/name
/cddb/disc/dtitle	Zagat/street	Zagat/street-city	G/name
/discs/disc/styles/style	Fodor/city	Fodor/telephone	R/lon
/cddb/disc/genre	Zagat/city	Zagat/telephone	G/lng
/discs/disc/genres/genre	Fodor/telephone	Fodor/type	R/lat
/cddb/disc/category	Zagat/telephone	Zagat/type	G/lat
/discs/disc/year	Fodor/type		R/category
/cddb/disc/year	Zagat/type		G/type

Restaurant data are represented by two famous datasets [9]. One dataset stems from the on-line guide 'Zagat', while the other dataset stems from the on-line guide 'Fodor'. Zagat contains 331 tuples and Fodor contains 533 tuples, where 112 coreferent tuples are counted (i.e. 112 restaurants occur in both lists). These datasets are defined in two pairs of schemas, called R1 and R2, respectively. Both schemas of the first pair R1 consist of 6 attributes, of which all 6 have been identified manually as being coreferent, i.e. each attribute in the Fodor schema corresponds to exactly one attribute in the Zagat schema, and vice versa. More specifically, 6 one-to-one matchings are established. The Fodor schema in the second schemas pair R2 is identical to the Fodor schema in the first schema pair R1. But the Zagat schema in the second schemas pair R2 consists of 5 attributes, of which the attribute "street-city" is a concatenation of the attributes "street" and "city". Thus, each of the 4 attributes in the Fodor schema corresponds to exactly one attribute in the Zagat schema, and the concatenation of the attributes "street" and "city" in the Fodor schema corresponds to the attribute "street-city" in the Zagat schema. So, four one-to-one matchings and one one-to-many matching are present. The truly coreferent schema elements are also presented in Table 3.9 and act as ground truth for our experiments.

Points of interest data are represented by two datasets which are defined in two schemas. The first dataset is made available by the Belgian company RouteYou[9], which is an on-line provider of cycling routes. In order to support their routing algorithms, RouteYou manages a database with POIs (see also the Introduction

[7] http://hpi.de/naumann/projects/repeatability/datasets/cd-datasets.html
[8] Discogs, http://www.discogs.com/data/
[9] http://www.routeyou.com

Chapter). This database is defined by a schema which consists of the attributes latitude, longitude, POI name, POI category, POI internal name (which is a copy of the POI name extended by location information) and the language in which the name and category are given. An important characteristic of the given POI database is that data is mostly contributed by independent users of the website. Hereby, a user can pinpoint a location on the map, type in the name of the POI he/she wants to add and associate one of the predefined POI categories to it. From the complete POI database, we inferred one dataset by selecting tuples in English and in a specific area: the center of Ghent. This resulted in the RouteYou dataset which consists of 136 tuples. The second dataset contains 945 tuples which were extracted from the Google Maps database and are related to the same specific area. The tuple extraction has been done using the Google Places API[6]. The resulting dataset is defined by a schema which consists of the attributes id, name, vicinity, lat, lng, googleId and type, of which 6 have been identified manually as being coreferent with the attributes of the RouteYou dataset.

Table 3.10 contains a summary of all datasets considered in the experiments. The number of attributes in the data varies between 5 and 24 (column 2 in Table 3.10), while the number of truly coreferent attributes varies between 5 and 6 (column 5 in Table 3.10). The number of detected coreferent tuple pairs varies between 33 and 112 (column 4 in Table 3.10), while the number of tuples varies between 124 and 945 (column 3 in Table 3.10).

Table 3.10 Real-world datasets.

Datasets	# attributes	# tuples	# dup	# coreferent attr.
S: CD.FreeDB	8	124	33	6
T: CD.Discogs	24	132		
S: R1.Fodor	6	533	112	6
T: R1.Zagat	6	331		
S: R2.Fodor	6	533	112	5
T: R2.Zagat	5	331		
S: POI.RouteYou	9	136	51	6
T: POI.Google	7	945		

3.7.2 Evaluation setting

To determine the quality of our approach, we compared its result against the manually derived results from Table 3.9. Based on the standard confusion matrix, we will consider the following three sets. The first set, denoted as B, contains the truly

[6]Google Places, http://developers.google.com/places/

coreferent objects which are discovered by our approach, i.e., so-called true positives. The second set, denoted as A, contains truly coreferent objects which are not identified, i.e., so-called false negatives. The last set, denoted as C, contains objects which are falsely identified as coreferent, i.e., so-called false positives.

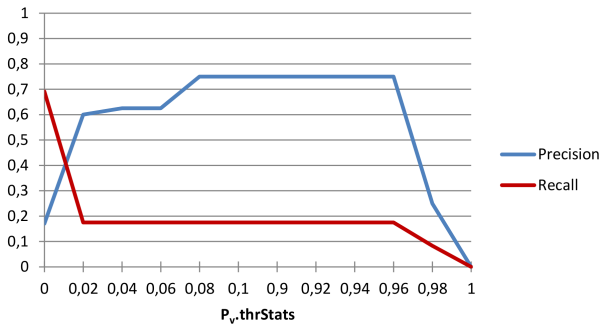
Precision is defined as the fraction of truly coreferent objects among all objects classified by a given algorithm as being coreferent:

$$\text{Precision} = \frac{|B|}{|B| + |C|}. \tag{3.18}$$

Recall is another important quality measure which in our case can be defined as the fraction of true positive objects among all coreferent objects present in a test dataset:

$$\text{Recall} = \frac{|B|}{|A| + |B|}. \tag{3.19}$$

Figure 3.5 Mean precision and recall over all datasets in function of $P_V.thrStats$ and $P_M.thrOverlap$ equal to 0.3.

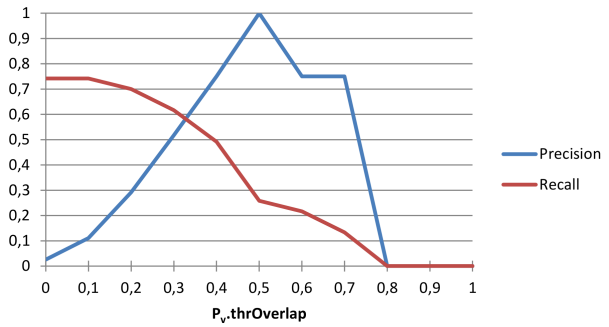


3.7.3 Experiment: Configuration of parameters of the vertical matcher

Goal. Our vertical matching algorithm 3.2 in Section 3.4 employs the parameters $P_V.thrStats$ and $P_V.thrOverlap$. $P_V.thrStats$ specifies the threshold above which statistical properties of attribute domains are considered as similar. $P_V.thrOverlap$ specifies the percentage of the domains overlap. This experiment evaluates the impact of these parameters on the precision and recall of the established vertical schema matching.

Procedure. For the parameter $P_V.thrStats$, a range from 0 to 1 with a step equal to 0.01 is considered. For the parameter $P_V.thrOverlap$, a range from 0 to

Figure 3.6 Mean precision and recall over all datasets in function of $P_V.thrOverlap$ and $P_M.thrStats$ equal to 0.9.



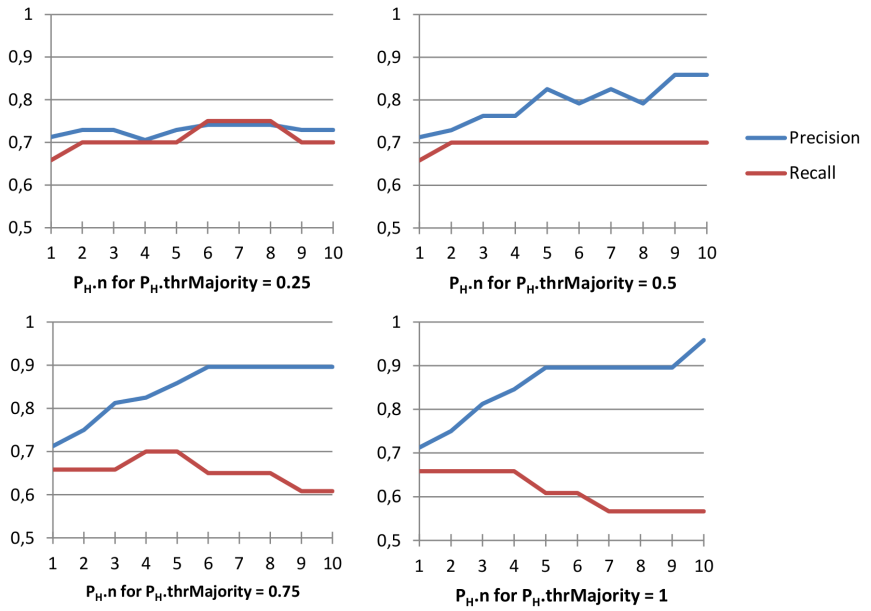
1 with a step equal to 0.1 is considered. Mean recall and precision for each value of $P_V.thrStats$ and $P_V.thrOverlap$ over all datasets are calculated. Statistical matcher is executed before the lexical matcher, thus, the overlap threshold does not have to be considered.

Result. Figure 3.5 shows the mean precision and recall over all datasets for the different values of the parameter $P_M.thrStats$ (uninterested results are omitted). For $P_M.thrStats$ values between 0.08 and 0.96 the statistical comparison of content data gives the highest precision of matching. In this case, precision is more important than recall because non matched truly coreferent attributes can be matched by the lexical matcher. Besides that, the criteria of the statistical matcher are strict (all properties have to be similar) because statistical information may mislead the matcher, i.e., there can exist domains which have similar properties but may describe non coreferent attributes. We choose $P_M.thrStats$ equal to 0.9 for the further evaluations.

Figure 3.6 shows the mean precision and recall obtained over all datasets for different values of the parameter $P_M.thrOverlap$ in the lexical matcher. For $P_M.thrOverlap$ values equal to 0.3 and 0.4 the lexical comparison of content data gives matchings with the highest precision and recall. The established matchings are the basis to detect coreferent tuple pairs, which in turn are used to derive the final schema matching, thus, the method has to derive as many as possible matchings at the expense of precision - the non coreferent matchings are eliminated by the horizontal matcher. Thus, $P_M.thrOvelap$ equal to 0.3 is selected for the further evaluations, because the smaller percentage of domain terms that overlap is easier to satisfy.

The combination of the matchings which are established by statistical and lexical matchers gives an average precision equal to 0.58 and recall equal to 0.79 over all datasets for $P_M.thrStats$ equal to 0.9 and $P_M.thrOvelap$ equal to 0.3.

Figure 3.7 Mean precision and recall over all datasets for different $P_H.thrMajority$ in function of $P_H.n$ before conflicts resolution.



3.7.4 Experiment: Configuration of parameters of the horizontal matcher

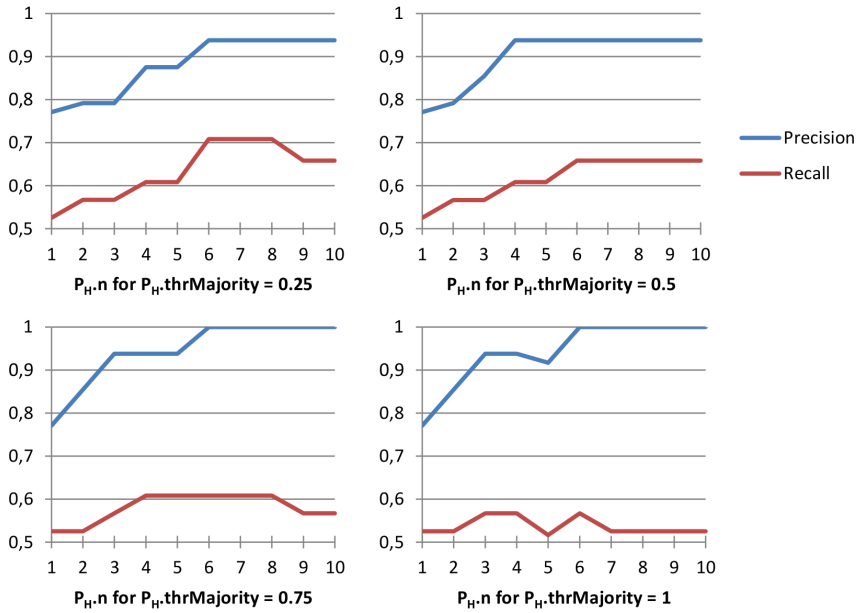
Goal. The goal of this experiment is to show the impact of the number $P_H.n$ of coreferent tuple pairs, which is the basis to establish the schema matching, and the parameter $P_H.thrMajority$ of our horizontal matching algorithm in Section 3.5 on the precision and recall of the established schema matching. $P_H.thrMajority$ is a threshold which specifies that a matching is considered as a correct matching by the horizontal matcher.

Procedure. For the parameter $P_H.n$, a range from 1 to 10 is considered. For the parameter $P_H.thrMajority$, values 0.25, 0.5, 0.75 and 1 are considered. $P_H.thrMajority$ equal to 0.25 (0.5, 0.75 or 1) means that if any vertical matching $m \in M_V$ between the particular attributes is repeated by a quarter (two quarters, three quarters or all, respectively) of detected coreferent tuple pairs then m is added to the set M of horizontal matchings. The mean precision and recall for each value of $P_H.thrMajority$ and $P_H.n$ over all datasets are calculated.

Result. Figure 3.7 shows the mean precision and recall over all datasets for different values of the parameters $P_H.thrMajority$ and $P_H.n$.

Setting $P_M.thrMajority$ to 0.5 or 0.75, and $P_H.n$ to 5 or 6 is sufficient to establish the schema matching with high precision and recall. More coreferent tuple pairs ($P_H.n$ greater than 6) do not increase the precision significantly or can even decrease the recall for a large value of $P_H.thrMajority$, because $P_H.thrMajority$ equal to 1 forces that a particular matching has to be confirmed by all the selected coreferent tuple pairs. However, this may be unrealistic and difficult to satisfy. Besides that, our horizontal matcher is based on the n most coreferent tuples pairs so using many coreferent tuple pairs may result in coreferent tuples pairs having assigned low certainty (because the values of some attributes may not be coreferent). Thus, it is recommended to use only a few coreferent tuple pairs but then those that are assigned the highest certainty and $P_M.thrMajority$ between 0.5 and 0.75.

Figure 3.8 Mean precision and recall over all datasets for different $P_H.thrMajority$ in function of $P_H.n$ after conflicts resolution.



3.7.5 Experiment: Conflict resolution

Goal. The goal of this experiment is to show the impact of the conflict resolution step of our approach (of Section 3.6) on the quality of the final schema matching.

Procedure. Like in the previous experiment, a range from 1 to 10 is considered for the parameter $P_H.n$; for the parameter $P_H.thrMajority$, values 0.25, 0.5, 0.75 and 1 are considered. The mean precision and recall for each value of $P_H.thrMajority$ and $P_H.n$ over all datasets are calculated after conflict resolution, and these are compared with the mean precision and recall before conflict resolution which have been calculated in the previous experiment and the results are shown in Figure 3.7.

Result. Figure 3.8 shows the mean precision and recall over all datasets after conflict resolution. Conflict resolution helps to significantly increase the mean precision from 0.8 to 0.92 at the expense of the mean recall which decreases from 0.67 do 0.6 over all values of $P_H.n$ and $P_H.thrMajority$ and over all datasets. However, for the recommended values of the parameters in Table 3.11, namely $P_H.n$ equal to 6 and $P_H.thrMajority$ equal to 0.75 (see previous experiments), the precision increases after conflict resolution from 0.89 to 1 at the expense of the recall which only decreases from 0.65 do 0.61 over all datasets.

3.7.6 Experiment: Results comparison

Goal. The goal of this experiment is to compare our novel content data-based schema matching approach with DUMAS [19] (see Section 3.2).

Procedure. For both approaches, respectively, precision and recall are calculated and compared over each pair of datasets. Our approach is configured with the parameters chosen based on the previous experiments and shown in Table 3.11. Our approach is evaluated on various datasets, thus, these parameters can be considered as universal so can work also for other datasets. The parameters of the available implementation of DUMAS[10] cannot be changed. We assume that the parameters are optimal and universal.

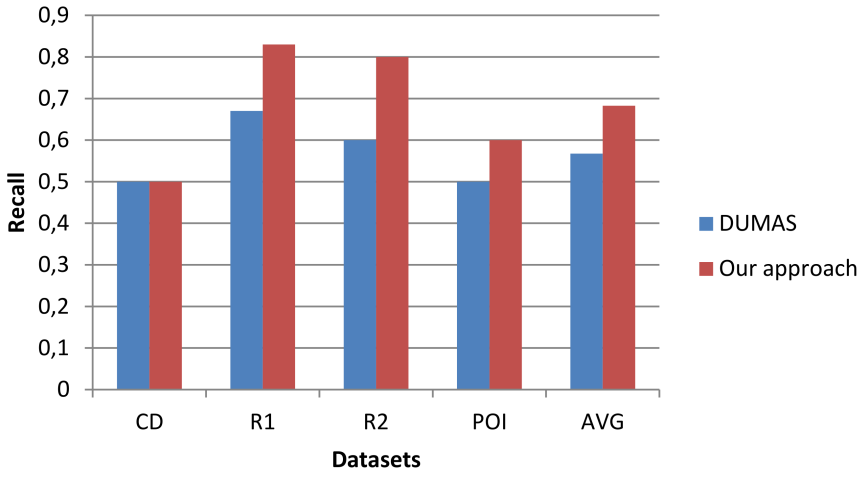
Table 3.11 Parameters set up of our approach.

Parameter	Value
$P_V.thrStats$	0.9
$P_V.thrOverlap$	0.3
$P_H.n$	6
$P_H.thrMajority$	0.75

[10]https://hpi.de/naumann/projects/repeatability/algorithms/dumas-duplicate-based-matching-of-schemas.html

Result. The precision equals 1 over all datasets for both approaches, thus, it is omitted. However, Figure 3.9 shows the recall over each pair of datasets separately for DUMAS and our approach, respectively. For each pair of datasets the recall of our approach is slightly higher than the recall for DUMAS. The average recall of our approach equals 0.68, while the average recall of DUMAS equals 0.57. This means that our technique correctly detects more matchings between corresponding schema elements than DUMAS.

Figure 3.9 Recall for DUMAS and our approach.



3.7.7 Benefits of using content data

Content data-based schema matching approaches are able to establish semantical matchings of corresponding schema elements in those situations where the schema information-only-based methods can be ineffective. For example, a schema matching method which is based only on element names, such as the Paths Matcher described in Chapter 2, is not able to create a matching if the names of coreferent attributes are synonyms. This means that content data add additional information about the particular attribute which can be used to better infer the semantics of the attribute and, as a consequence, create a matching between coreferent attributes. Moreover, attribute names may even mislead the schema matching methods which are only based on schema information. For example, let us consider the compact disc datasets which are used in our evaluation. The attribute “genre” in the Discogs dataset represents the general music category, such as “rock”, “pop”, etc., and it corresponds to the attribute “category” in the FreeDB dataset. However, the schema of FreeDB dataset also contains an attribute “genre” which can mislead the schema information-based matching method because it represents a more

specific music style, such as “hard rock”, and, it is corresponding to the attribute “style” in the schema of Discogs dataset. This and similar issues can be overcome using content data what we show in the above experiments. Thus, content data are a powerful and valuable source of information which can be used to considerably improve schema matching. In contrast, approaches which are based on schema information only can establish matchings of attributes which are not coreferent based on the content data, e.g., “id” attributes. The combination of these two schema matching methods can be valuable and effective but it is out of the scope of this work.

3.8 Conclusions

In this chapter, a content data based schema matching algorithm has been proposed as a way to construct proper matching between corresponding schema elements of heterogeneous datasets. The algorithm is especially useful in cases where finding the correspondences between the schema elements based on schema information only is difficult or impossible. Our novel technique employs possibilistic truth values, to express certainty of matchings, similarities etc., and fuzzy integers, to express cardinalities of sets of true propositions based on the certainty of their truth expressed using PTVs. This allows us to explicitly cope with the (un)certainty of semantical one-to-one and one-to-many schema matchings which are set up by our novel techniques in an automated fashion. As a consequence, solutions to the attribute granularity problem and the data coverage problem are proposed. The behaviour of the novel schema matching algorithm has been evaluated on several real life datasets, thus providing us with a valuable insight into the influence of the different parameters. Moreover, it has been shown what are the advantages of the proposed approach compared with a schema matching approach based on schema information only.

The novel techniques presented here constitute an answer to the second research question of this thesis which concerns the semantical matching of corresponding schema elements of heterogeneous datasets using content data. This means that our techniques are able to establish a matching of schema elements by not using element names. Indeed, element names can in some cases be confusing, e.g. when they are synonyms. Instead, in our techniques coreference in the content of the data sources is used to derive the meaning of the schema elements under consideration. However, checking the content data can also be insufficient because the same information can be represented in different ways. Some knowledge base is necessary to resolve this problem. Nevertheless, in many cases it is unfeasible to use a predefined knowledge base during coreference detection because such a knowledge base might not exist. Hence, constructing such a knowledge base in a dynamic way is an important aspect of improving the detection of coreferent ob-

jects. In the next chapter the dynamic construction of a knowledge base, based on the content of the data sources is studied.

4

Dynamical construction of a knowledge base: a partial order relation on the domains of attributes

The following publications have been based on the contents of this chapter:

- M. Szymczak, A. Bronselaer, S. Zadrożny, and G. De Tré, "Dynamical construction of binary relations in coreference detection," 31st Annual Meeting of North American Fuzzy Information Processing Society, IEEE, Proceedings. pp. 100-–106. Berkley, USA 2012,
- Bronselaer, A., Szymczak, M., Zadrożny, S. & De Tré, G. "Dynamical Order Construction in Data Fusion," Information Fusion. (Under review)

4.1 Introduction

In this chapter, dynamical construction of a partial order relation on the domain of an attribute. Such as a relation reflects a notion of generality and is meant as a knowledge base to support coreference detection. More precisely, the Dynamical Order Construction (DOC) algorithm is specified. The DOC algorithm is investigated closely in terms of different generality measures, which are used to form this relation, and its application for coreference detection. An experimental evaluation of our method confirms its effectiveness compared to the usage a fixed and prede-

fined taxonomy and provides some suggestions as to the choice of its parameters values.

4.1.1 Problem illustration

As a running example throughout this chapter, we consider a relational database of Points Of Interest (POIs) used already in the previous chapters. Table 4.1 shows a sample of these POIs. Provided that the tuples between horizontal lines shown in Table 4.1 have been detected as coreferent tuples (duplicates), the first step in the construction of an order relation is projecting these tuples over their *category* attribute. These values, namely "Tower", "Hist. Bld.", "Monument", "Restaurant", "City", "POI" and their taxonomical connection are the candidate elements and basis for the construction of the order relation. These elements can be ordered as follows. "POI" is the most general and contains three sub elements, "Monument", "Restaurant" and "City". "Monument" is specialized by "Hist. Bld." which in turn has sub element "Tower".

Table 4.1 Example of coreferent POIs.

name	**lon.**	**lat.**	**category**
Belfry	3.725098	51.053552	Tower
Belfry	3.724837	51.053555	POI
Belfrey	3.724911	51.053653	Monument
Korenlei 2	3.720472	51.055569	Hist. Bld.
Korenlei 2	3.720472	51.055568	Restaurant
Ghent	3.715449	51.025529	City
Gent	3.715449	51.025529	POI
Castle	3.721106	51.056984	Monument
Castle	3.721100	51.056981	POI

A number of problems occur that deserve attention:

- How to construct an order relation?
- How can a dynamically constructed order relation improve coreference detection?

4.1.2 Contributions

With respect to the problems given above and in Chapter 1 in case of semantical comparison where attributes values are compared using external knowledge (e.g., ontology, taxonomy), the following important contributions are made by this chapter. An algorithm is proposed for the construction of an order relation in an automated fashion over the domain of a selected attribute (called *category* attribute).

The input for this algorithm is a list of coreferent tuples (i.e., tuples that mutually describe the same entity). The order relation obtained can be used to improve detection of coreferent tuples, which is shown in this chapter, and also to fuse data which is investigated in Chapter 5. The correctness of the constructed order relation strongly depends on the quality of given coreferent tuples what is guaranteed by a multiplicity cut operator which filters misleading data. This approach has the advantages that there is no need for a priori taxonomical knowledge on the attribute domain and that the order relation automatically adapts to the values in the dataset.

4.1.3 Outline

The remainder of this chapter is structured as follows. Related work is described in Section 4.2. Next, in Section 4.3, it is explained how a semantical knowledge base in the form of an order relation $\leq_a$, can be constructed. To this aim, an algorithm for Dynamical Order Construction (DOC) is introduced. The important features of this algorithm are pointed out and the parameters are discussed. In Section 4.4 an experimental study of the proposed methods and techniques is reported and supported by careful statistical analysis. Moreover, the impact of an order relation on the coreference detection is evaluated. Finally, Section 4.5 summarizes the most important contributions of this chapter.

4.2 Related work

There is a large body of work on the automated construction of ontologies from *textual documents*. Weng et al. use formal concept analysis to assemble the different levels of ontological concepts [95]. Lee et al. adopt the use of a Chinese Part-Of-Speech tagger to extract concepts from Chinese text documents based on grammatical analysis [96]. Dahab et al. adopt semantical patterns for concept and relationship identification [97]. Semantical patterns are hereby defined as "a generic format for natural language expression". Liu et al. build an ontology from a set of keywords rather than from a textual document [98]. Therefore, the set of keywords is first enriched by using a general purpose knowledge base (Probase). Then, multi-branch clustering is used to obtain a taxonomy. The technique introduced in this chapter differs from these techniques in several ways. First, like Liu et al. in [98], we do not consider textual documents from which an ontology/taxonomy should be inferred. Instead, we consider a range of values for a specific attribute in a database. Second, unlike existing techniques, we do *not* rely on an external knowledge base. Instead, the observation that different values of the same attribute are used interchangeably to describe a property of an entity, forms the basis for knowledge inference.

4.3 Dynamical order relation construction

In this section, a method of Dynamical Order Construction (DOC) is proposed to (re)construct taxonomical knowledge for an attribute $a \in \mathcal{R}$. More specifically, an order relation $\leq_a$ is constructed based on the observation of coreferent data.

Remark. *It is noted that such an order relation $\leq_a$ is more general than what is usually understood as a taxonomy. Indeed, while a taxonomy is sometimes assumed to be equivalent to a* tree*, a (partial) order relation is equivalent to a* directed graph*. In order to maintain the notion of a tree, a structural constraint to enforce the "single-parent" requirement is necessary. However, such a priori constraints might imply more non-comparable values and thus more ties e.g. upon data fusion time. In order to minimize the number of non-comparable values, a priori assumptions on the order relation are avoided. It is pointed out that non-comparable values still might exist, because $\leq_a$ is a partial order relation. As such, non-comparable values will be processed by a tie-breaking mechanism.*

4.3.1 Generative multirelation

As already mentioned, the construction of $\leq_a$ relies on observed coreferent tuples, i.e., multiple tuples representing the same entity. We thus start by defining the concept of coreferent tuples more formally.

Definition 9 (Δ-partition)**.** *Let R be a relation with schema $\mathcal{R}$ and let $\sim$ be an equivalence relation on R where $t_1 \sim t_2$ means that t_1 and t_2 are coreferent tuples. A Δ-partition $\{\Delta_i\}_{i=1..m}$ (denoted $\mathcal{E}$) is a subset of non-singleton equivalence classes of $\sim$, i.e., comprises sets of tuples from R such that each set Δ_i contains at least two tuples and all of them are mutually coreferent, and such that two tuples from two different sets are not coreferent.*

A Δ-partition is usually generated by an Entity Resolution algorithm. In Table 4.1, the horizontal lines between tuples indicate a Δ-partition $\mathcal{E}$ consisting of four sets of coreferent tuples. The first set contains three coreferent tuples that describe the Belfry in Ghent, while the other three sets each contain two coreferent tuples that describe respectively a restaurant called Korenlei 2, the city of Ghent and castle. Let us now assume that there exists an attribute $a \in \mathcal{R}$ for which we want to construct an order relation $\leq_a$. We do so by first constructing a multirelation that models all observed coreferent tuple pairs of values for attribute a.

Definition 10 (Generative multirelation)**.** *Consider a relation R with schema $\mathcal{R}$ for which a Δ-partition $\mathcal{E} = \{\Delta_i\}_{i=1..m}$ is given. For $a \in \mathcal{R}$, the generative multirelation G_a is a binary multirelation over* $\mathrm{dom}(a)$ *such that for all* $(v_1, v_2) \in$

$\mathrm{dom}(a) \times \mathrm{dom}(a)$*:*

$$G_a(v_1, v_2) = G_a(v_2, v_1) = \\ = |\{(t_1, t_2) \mid t_1 \neq t_2 \wedge \exists_i t_1, t_2 \in \Delta_i \wedge t_1[a] = v_1 \wedge t_2[a] = v_2\}| \,. \quad (4.1)$$

The generative multirelation G_a is a multirelation for which the multiplicity of a couple of values (v_1, v_2) is equal to the number of coreferent tuple couples (t_1, t_2) such that $(t_1[a], t_2[a]) = (v_1, v_2)$. It can be written as a weighted graph with a subset of $\mathrm{dom}(a)$ being the set of nodes, a subset of $\mathrm{dom}(a) \times \mathrm{dom}(a)$ being the set of edges and the weights of an edge equal to $G_a(v_1, v_2)$.

Figure 4.1 Generative multirelation G_a for attribute a = "category" of the Δ-partition in Table 4.1.

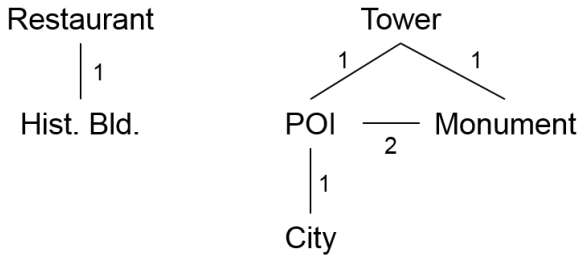


Figure 4.1 shows the generative multirelation G_a for attribute "category" of the Δ-partition in Table 4.1 as such a weighted graph. For an attribute a, the generative multirelation G_a provides an insight in the *interchangeability* of values from $\mathrm{dom}(a)$. More specifically, it can be inferred how many times a value v can be considered as "coreferent" with another value v'. Intuitively, if there exist many such values v', then v must represent a very general concept, because it can be linked (in the sense of duplication) to many other values. However, if there exist few values v', then v represents a rather specific concept because it is used solely in a specific context. As such, the generality of a value can be approximated by the number of neighbors it has in G_a. In addition, if a value has many neighbors, it must have a high frequency as well. Thus, the frequency of a value in the dataset/database might as well be an approximation of generality. In the next section we will propose three measures to approximate generality and evaluate them experimentally in Section 4.4. With such measures at hand, it is possible to infer an order relation $\leq_a$ from G_a, such that $\leq_a$ orders values from $\mathrm{dom}(a)$ based on their suspected *generality*. However, making such an inference brings along two problems that need handling:

- **Constraints.** The requirement to create a partial order $\leq_a$ implies that the minimal constraints for such a relation must be met. We recall that a partial

order relation is reflexive, antisymmetric and transitive. The transformation from the generative multirelation G_a to the order relation $\leq_a$ should take these constraints into account.

- **Robustness.** The input for construction of $\leq_a$ is a Δ-partition, which is in turn output of an Entity Resolution algorithm. Consequently, a Δ-partition may contain errors and the construction of $\leq_a$ should be robust to these errors.

Remark. *At this point, an assumption on the data can be stated. In the above explanation, the detection of interchangeability relies on* repetition *of values. The DOC method therefore relies on the assumption that the attribute, for which an order relation is constructed, has a low number of unique values.*

4.3.2 Transformation into an order relation

We will now introduce a general strategy for the inference of $\leq_a$ from G_a. This strategy uses the multiplicity cut operator to derive a regular, binary relation from G_a. As explained in Section 1.4.2, the k-cut of G_a is denoted $(G_a)_k$ and is a regular relation that maintains only those couples with multiplicity larger than or equal to k. Given the fact that the multiplicity of a couple is equal to the number of observations in the data, multiplicity filtering offers a sense of *robustness* to errors in the Δ-partition. This statement will be verified experimentally in Section 4.4.

After application of the k-cut, the next step is to transform $(G_a)_k$ into a partial order relation. Therefore, $(G_a)_k$ must be made *reflexive*, *antisymmetric* and *transitive*. While making a relation reflexive is trivial, making it antisymmetric and transitive is not. On the contrary, the way in which antisymmetry is defined, determines to a large extent the correctness of our order relation. Therefore, we explain in detail our approach to these transformations.

Let us assume that $(G_a)_k$ is reflexive or at least transformed into a reflexive relation. We note that, for any $(v_1, v_2) \in \mathrm{dom}(a)^2$, we have that $G_a(v_1, v_2) = G_a(v_2, v_1)$ (Definition 10). Therefore, $(G_a)_k$ is by definition symmetric. This means that, for any $(v_1, v_2) \in (G_a)_k$, we must either remove (v_1, v_2), (v_2, v_1) or both, in order to make the resulting relation antisymmetric. In order to decide which couple should be removed from the relation, recall that $\leq_a$ should provide a notion of *generality*. This means that $v_1 \leq_a v_2$ if v_2 is more general than v_1. As such, the decision on which couple to remove should be driven by an estimate of the generality of both values. Such a measure of generality is denoted as λ and we propose three different measures that are evaluated experimentally in Section 4.4.

A very simple notion of generality is obtained by inspecting multiplicity (i.e., frequency). The first measure of generality is therefore given by the number of

occurrences of each value within the Δ-partition. Formally:

$$\forall v \in \mathrm{dom}(a) : \lambda_1(v) = \left(\bigoplus_{\Delta \in \mathcal{E}} \Delta[\![a]\!] \right)(v). \tag{4.2}$$

where $\bigoplus$ denotes the union operation for multisets; cf. section 1.4.2.

This estimation relies on the assumption that a value of a is more general if it is often "confused" with other values, which is measured by the fact that it occurs in many equivalence classes Δ.

The second measure of generality is given by the number of unique values to which a given value v is related in G_a. We can write that:

$$\forall v \in \mathrm{dom}(a) : \lambda_2(v) = |\{v' \mid G_a(v, v') > 0\}|. \tag{4.3}$$

This second measure estimates the generality of a value v in terms of the number of unique values v' to which v is observed equivalent, i.e., v and v' occur as the projections over a of two tuples in the same equivalence class Δ_i. Note that the symmetric construction of G_a implies that we do not need to check whether $(v', v) \in G_a$.

A potential weakness of measures λ_1 and λ_2 is their dependency on the partition $\mathcal{E}$. More specifically, because λ_1 and λ_2 depend on $\mathcal{E}$, it can be suspected that the quality of the generated $\leq_a$ varies in terms of $\mathcal{E}$. In order to further investigate this, a third measure of generality is considered that does not depend on $\mathcal{E}$. This measure is established by adopting a similar notion as for λ_1, but $\mathcal{E}$ is generalized to the whole relation R. As such, we have that:

$$\forall v \in \mathrm{dom}(a) : \lambda_3(v) = (R[\![a]\!])(v). \tag{4.4}$$

where $R[\![a]\!]$ (in general $R[\![A]\!]$, A is a singleton $\{a\}$) is a *multi-projection* which is defined as follows.

Definition 11 (Multi-Projection)**.** *Let R be a relation with schema $\mathcal{R}$ and consider attributes $A \subseteq \mathcal{R}$. The multi-projection of R over A is a multiset of tuples $R[\![A]\!]$ such that:*

$$R[\![A]\!](t) = \begin{cases} 0 & \textbf{\textit{if}}\ t \notin R[A] \\ |\{t' : t' \in R \wedge t'[A] = t\}| & \textbf{\textit{otherwise}} \end{cases} \tag{4.5}$$

Definition 11 states that the multi-projection of a relation R contains the tuples from the regular projection possibly with multiplicity greater than 1. Consider the relation shown in Table 4.1 and let a be attribute "category", then $R[\![a]\!]$ is characterized by:

$R[\![a]\!]$(Tower)	= 1,	$R[\![a]\!]$(Hist. Bld.)	= 1,
$R[\![a]\!]$(POI)	= 3,	$R[\![a]\!]$(Restaurant)	= 1,
$R[\![a]\!]$(Monument)	= 2,	$R[\![a]\!]$(City)	= 1.

In this multiset, values "Monument" and "POI" occur respectively two and three times.

Table 4.2 shows the different measures of generality applied to attribute $a =$ "category" of the Δ-partition in Table 4.1. Note the difference for value "Tower", which occurs only once, but can be linked to two different values.

Table 4.2 Different generality measures λ applied to attribute $a =$ "category" of the Δ-partition in Table 4.1.

Value	λ_1	λ_2	λ_3
POI	3	3	3
Monument	2	2	2
City	1	1	1
Hist. Bld.	1	1	1
Restaurant	1	1	1
Tower	1	2	1

In this example, λ_1 and λ_3 provide the same result because the Δ-partition comprises the whole relation. However, it is important to note that the difference between λ_1 and λ_3 is subtle. Whereas λ_1 is based on the principle that a value v is general if it is probable to be "confused" with other values in the sense of duplication, λ_3 simply states that a value is general if it occurs a lot. The independence of $\mathcal{E}$ thus comes with the downside that measuring generality by means of λ_3 seems less justified. In addition, λ_1 relates its estimate of "confusion probability" to frequency of values, while λ_2 actually measures the number of values with which a given value v is confused. It is therefore to be expected that λ_2 will be the best measure of generality. In Section 4.4, the (dis)advantages of the proposed measures will be investigated experimentally in order to obtain a better evaluation of their usefulness.

Either of the proposed measures for generality can be used to ensure antisymmetry. After transformation of $(G_a)_k$ into an antisymmetric relation, the final task at hand is to ensure transitivity. This can be done by calculating the transitive closure of the relation. The transitive closure of a binary relation B is its minimal superrelation that is transitive and is denoted as B^+. In literature, this problem has been studied extensively [99–102] and we will not elaborate on these results. It suffices to say that we adopt Warren's algorithm within the scope of this work [101].

Algorithm 4.1 presents the pseudo-code for the DOC algorithm. The strategy to transform G_a into an order relation has the advantage that it attempts to infer only those orderings that are clearly observed in the data (i.e., the Δ-partition). Removal of couples from or addition of couples to $(G_a)_k$ is caused by structural constraints for obtaining an order relation. From that perspective, it can be assumed that $\leq_a$ is a reasonable approximation of a generality ordering. On the

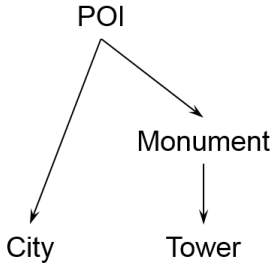
downside, it is to be expected that $\leq_a$ can not compare all values if k used for k-cut becomes large.

Algorithm 4.1 DYNAMICALORDERCONSTRUCTION

Require: G_a, k, λ
Ensure: An order relation $\leq_a$
1: *Ord* $\leftarrow (G_a)_k$
2: **for all** $v \in \mathrm{dom}(a)$ **do**
3: **add**(v, v)
4: **end for**
5: **for all** $(v, v') \in$ *Ord* $\mid v \neq v'$ **do**
6: **if** $\lambda(v) > \lambda(v')$ **then**
7: **remove**(v, v')
8: **else if** $\lambda(v) < \lambda(v')$ **then**
9: **remove**(v', v)
10: **else**
11: **remove**(v', v)
12: **remove**(v, v')
13: **end if**
14: **end for**
15: $\leq_a \leftarrow$ Ord$^+$
16: **return** $\leq_a$

Figure 4.2 shows the result of applying the DOC algorithm to the generative multirelation from Figure 4.1 with $\lambda = \lambda_1$ and $k = 1$. The order relation is shown as directed graph with the understanding that edges depart from the more general values and end in the more specific values. For the sake of clarity we are using here the Hasse diagram of the relation. Note that for "Restaurant" and "Hist. Bld.", the $\lambda_1(.)$ values are equal, implying that they are removed from the order relation.

Figure 4.2 Order relation $\leq_a$ derived from G_a from Figure 4.1 for $k = 1$ and λ_1.



4.4 Evaluation and discussion

In this section, an experimental evaluation of the proposed techniques is presented. The accuracy, stability and impact of the order relation on the coreferent tuples detection are evaluated. Moreover, improvement of the ground truth taxonomy is proposed.

4.4.1 Datasets

Throughout the several experiments, seven datasets are used. The first dataset is the famous 'restaurant' dataset [9], which contains two lists of restaurants. One list of restaurants stems from the on line guide 'Zagat', while the other list stems from the on line guide 'Fodor'. The union of these two lists counts 864 tuples, where 112 coreferent tuples are counted (i.e. 112 restaurants occur in both lists). Within the context of our experiments, the attribute of interest is the type of the restaurant, which specifies the cuisine that can be found in the restaurant.

Six other datasets are real-life datasets made available by the Belgian company RouteYou[1], which is an on line provider of cycling routes[2]. As a reminder, in order to support their routing algorithms, RouteYou manages a database with POIs (see also the Introduction Chapter). Data stored about a POI comprises latitude, longitude, POI name, POI category and the language in which name and type are given. An important characteristic of the given POI-database, is that data is mostly contributed by independent users of the website. Thus, many coreferent POIs are inputted, because there is no verification at input time whether the new POI is already in the database. From the complete POI database, we infer six datasets by partitioning according to the 'language' attribute. Within the context of our experiments, the attribute of interest is the category of POI, which provides a rudimentary classification of the POI.

The details of the used datasets are shown in Table 4.3. Table 4.3 shows the name and the number of tuples in the dataset. In addition, the number of found coreferent tuple clusters is reported. The coreferent tuples detection algorithm described in [103] is used. Finally, Table 4.3 also reports the number of sets of coreferent tuples that are non-trivial, that is for which $\Delta[\![a]\!]$ contains at least two *distinct* values.

In Table 4.4, the distribution of the sizes of Δ are shown for optimally relaxed comparison operator. As mentioned already, the majority of the cases involves only two coreferent tuples. There is a small number Δ with three coreferent tuples, and cases with more than three coreferent tuples are very rare.

Because the attribute of interest in each of the datasets is the 'category' attribute, some more details on this attribute are made available. In Table 4.5, the

[1] http://www.routeyou.com

[2] A part of this dataset is also used in Chapter 3

Table 4.3 Dataset details.

Dataset	**# tuples**	**Δ found**	**Δ non-trivial**
R-NL	495651	3063	1724
R-EN	436616	1790	1349
R-FR	428163	2130	1463
R-DE	401176	1495	1324
R-ES	398451	1441	1312
R-IT	397137	1438	1311
Restaurant	864	112	71

Table 4.4 Distribution of $|\Delta|$ for all datasets.

Dataset	2	3	4	5	6	7
R-NL	2916	130	16	0	0	1
R-EN	1775	13	2	0	0	0
R-FR	1867	258	3	1	0	1
R-DE	1487	7	1	0	0	0
R-ES	1437	4	0	0	0	0
R-IT	1434	4	0	0	0	0
Restaurant	112	0	0	0	0	0

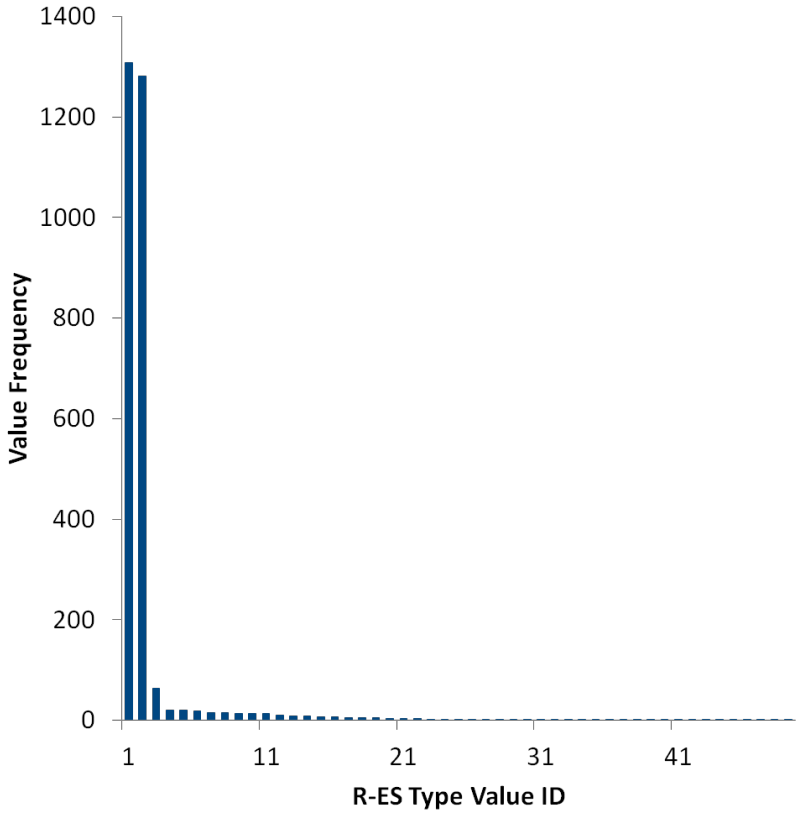
number of unique values for attribute 'category' is given per dataset. Two observations are important. First, in Section 4.3, it is mentioned that the DOC method assumes a low number of unique values for the attributes on which it operates. It can be verified that this assumption is met. Secondly, it can be seen that the number of unique values is varying over the datasets.

Table 4.5 Unique value count for attribute 'category'.

Dataset	# Unique Values for 'category'
R-NL	169
R-EN	104
R-FR	81
R-DE	57
R-ES	49
R-IT	47
Restaurant	37

In each of the datasets considered here, the 'category' value has a typical "long tail" distribution, meaning that few values occur many times, while most values occur very few. Figure 4.3 shows such a typical distribution for the 'category' attribute in the R-ES (RouteYou-ES) dataset. Plots for other datasets are omitted here as it is confirmed that they are similar.

Figure 4.3 Typical distribution of values for attribute 'category' (dataset R-ES).



4.4.2 Comparison with taxonomical ground truth

Goal. In this experiment, the accuracy of the DOC method is evaluated by comparing its result with a ground truth taxonomy. The influence of parameters k and λ is investigated, i.e., k specifies the k-cut of the generative multirelation G_a and is a regular relation that maintains only those couples with multiplicity larger than or equal to k, λ is a generality measure.

Procedure. For each of the seven datasets, the DOC method is applied on the 'category' attribute. The DOC method (Algorithm 4.1) has three input parameters. The generative multirelation G_a, a measure for generality λ and a cutoff threshold k. The first parameter, i.e., a relation G_a is constructed for each dataset as described in Section 4.3, based on the found coreferent tuples. The parameters λ and k are ranged over a set of alternatives in order to investigate their influence. For

λ, the three measures proposed in Section 4.3 are considered. For cutoff threshold k, a range going from 1 to 20 is considered. The order relation generated by the DOC method is compared to ground truth in the form of a reference taxonomy. For the six RouteYou datasets, such a reference taxonomical structure for POI categories was provided by a company expert. For the Restaurant dataset, a reference taxonomy was downloaded from the Zagat website[3][4].

Evaluation metrics. In the context of this experiment, two order relations $\leq_a$ (generated by DOC) and $\leq_{ref}$ (ground truth taxonomy) are compared. On the one hand, a comparison of the order relations as a whole is performed. For that purpose, the order relations are considered as sets of pairs and are compared as such with the well-known Tversky index [104]. For two sets X and Y, the assymetrical Tversky index [104] is given by:

$$\mathrm{Tve}(X,Y) = \frac{|X \cap Y|}{|X \cap Y| + \alpha \cdot |X \ominus Y| + \beta \cdot |Y \ominus X|} \tag{4.6}$$

where α and β are positive real-valued parameters of the index, $\ominus$ is the set difference. Because the DOC method does not aim at constructing an order relation over the complete domain of an attribute a, but rather focuses on those values that appear as coreferent tuples, we want to measure the extent of inclusion of $X = \leq_a$ in $Y = \leq_{ref}$. Such an inclusion measure is obtained by setting $\alpha = 1$ and $\beta = 0$. Indeed, choosing these values for α and β simplifies the Tversky index to $|\leq_a \cap \leq_{ref}|/|\leq_a|$.

This measure is minimal if $\leq_a$ and $\leq_{ref}$ are disjoint and is maximal if $\leq_a \subseteq \leq_{ref}$. In the following, we refer to this index as the Tversky Inclusion (TI) index. If $\leq_a$ is considered a knowledge base from which assertions on pairs of values $(x, y) \in \mathrm{dom}(a)^2$ can be made, then the TI index provides a *global* evaluation of $\leq_a$. However, because the DOC method produces $\leq_a$ with the intention of fusing coreferent values in a Δ-partition $\mathcal{E}$ (see next Chapter 5), it is also interesting to provide a more *local* evaluation. Hereby, "local" means to only evaluate those assertions that are relevant to the task of fusing values in $\mathcal{E}$. Let us consider a Δ-partition $\mathcal{E}$ as introduced in Section 4.3. Let us begin with noting that an assertion is *trivial* if it resembles $x \leq_a x$, due to the fact that $\leq_a$ is reflexive. In our evaluation, we consider only non-trivial assertions. For $(x, y) \in \mathrm{dom}(a)^2$ such that $x \leq_a y$, three cases can be considered. In the first case, the assertion is confirmed by the ground truth, meaning that $x \leq_{ref} y$. In this case, the assertion by $\leq_a$ is considered **correct**. In the second case, the assertion is falsified by the ground truth, meaning that $y \leq_{ref} x$. In this case, the assertion by $\leq_a$ is considered **wrong** or **erroneous**. In the third case, the ground truth makes no assertion

[3] http://www.zagat.com

[4] This data was removed on line from the Zagat website after the redesign on 29th of July, 2013.

about x and y. In this case, the assertion by $\leq_a$ is **non confirmed**. For non confirmed assertions, it is unknown whether they are correct or not. Therefore, we treat them as a separate case. Casting these cases into Boolean logic sentences, we have that an assertion is *correct* if $(x \leq_a y) \wedge (x \leq_{ref} y)$, an assertion is *erroneous* if $(x \leq_a y) \wedge (y \leq_{ref} x)$ and finally, an assertion is *non confirmed* if $(x \leq_a y) \wedge \neg (x \leq_{ref} y \vee y \leq_{ref} x)$. This leads us to the following three local evaluation metrics:

$$\mathrm{COR}\left(\mathcal{E}, \leq_a, \leq_{ref}\right) = \frac{\sum_{\Delta \in \mathcal{E}} \#\text{corr. assertions}}{\sum_{\Delta \in \mathcal{E}} \#\text{assertions}}$$

$$\mathrm{ERR}\left(\mathcal{E}, \leq_a, \leq_{ref}\right) = \frac{\sum_{\Delta \in \mathcal{E}} \#\text{err. assertions}}{\sum_{\Delta \in \mathcal{E}} \#\text{assertions}}$$

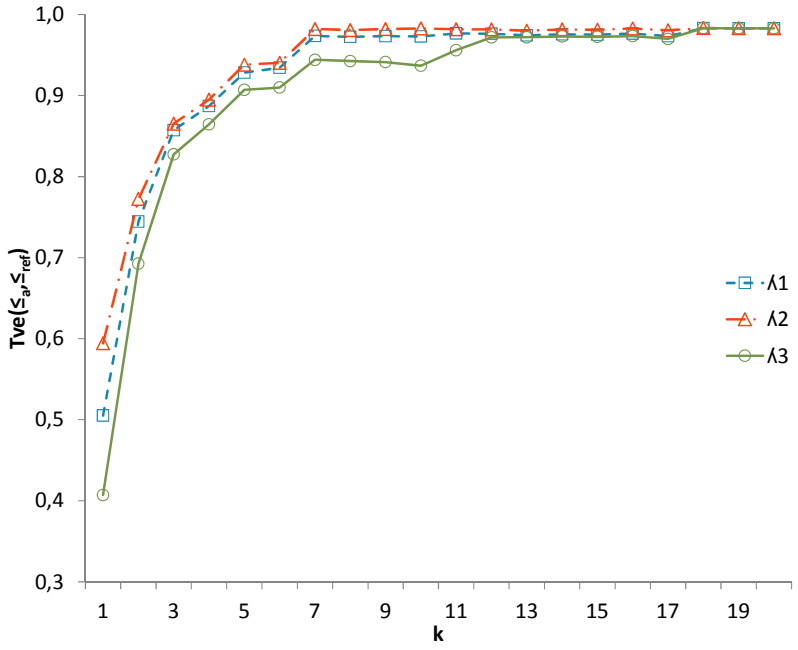
$$\mathrm{NCF}\left(\mathcal{E}, \leq_a, \leq_{ref}\right) = \frac{\sum_{\Delta \in \mathcal{E}} \#\text{non conf. assertions}}{\sum_{\Delta \in \mathcal{E}} \#\text{assertions}}.$$

It can be verified logically that an assertion is either correct, erroneous or non confirmed. Therefore, we have that the sum of COR, ERR and NCF is always 1.

Results and discussion. Figure 4.4 shows the mean TI indices over all seven datasets, for different measures of generality λ and in function of the cut-off threshold k. As explained in Section 4.3, parameter k serves as a way to filter erroneous information (i.e., errors made by Entity Resolution, errors in the data...). More specifically, k indicates the minimal number of times a connection between values needs to be observed, before an assertion on those values is done. Figure 4.4 illustrates that a larger value for k implies a higher inclusion of $\leq_a$ in $\leq_{ref}$ (the ground truth). As such, higher values of k indeed filter out non confirmed or erroneous information. Of course, by increasing k, the total number of assertions decreases, meaning that higher k leads to a *smaller*, but *more correct* relation. The mean number of assertions over all seven datasets, for different measures of generality λ and in function of k is shown in Figure 4.5. From this figure, it can be learned that the number of assertions decreases rapidly if k grows. Interestingly, for smaller k, the number of assertions is *minimal* if generality is measure by λ_2, which is also the measure for which most assertions are *correct*. From Figures 4.4 and 4.5 it can be concluded that k should be chosen sufficiently low to assure enough assertions, but choosing $k = 1$ might introduce some false assertions. In practice, if the Entity Resolution algorithm is sufficiently trustworthy, $k = 1$ or $k = 2$ is a good choice.

The difference between the measures of generality (Section 4.3) is analyzed by means of statistical testing. For each value of k, it is tested whether there is a significant difference between the λ. Because the sequence of TI indices over the seven datasets is *not* distributed normally, non-parametrical tests for comparison of related samples are used. More specifically, a Friedman test [105] and a Kendall's W test [106] are performed to analyze the differences between the three

Figure 4.4 Mean TI indices $\mathrm{Tve}(\leq_a, \leq_{ref})$ over all datasets for different λ in function of k.

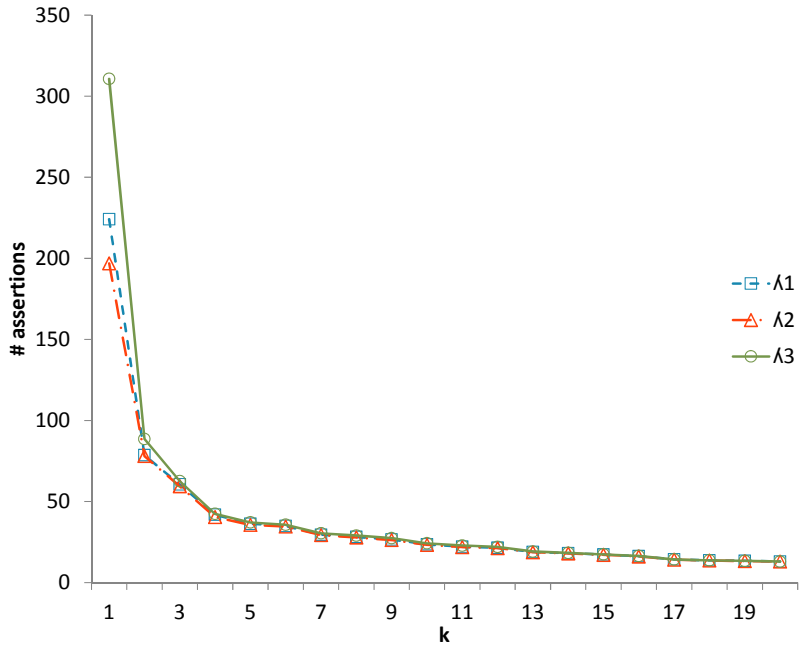


λ. Both tests are applied on *weighted* TI indices with the number of non-trivial cases (Table 4.3, rightmost column), to account for the differences in dataset sizes.

The results of the statistical analysis on the TI indices are shown in Table 4.6, where the mean ranks (MR) for each λ are shown, paired with Kendall's W and the statistical significance of the hypothesis that there is a difference between the three measures. From these results, we learn that the mean rank of the TI index for λ_2, is always better than the mean ranks of the TI indices for λ_1 and λ_3. For $k < 18$, there is a significant difference (p-value 0.05) between the three measures. Next, Wilcoxon tests [107] are performed in which the differences between two measures are analyzed. From this additional test, it is concluded that for $k \leq 17$, the TI index for λ_2 is significantly greater than the TI index for λ_1, which is in turn greater than the TI index for λ_3.

An even more detailed insight in the accuracy of the DOC method is obtained by quality assessment of assertions. Therefore, the local evaluation metrics COR, ERR and NCF are calculated for each of the seven datasets. The mean $\mathrm{COR}\,(\mathcal{E}, \leq_a, \leq_{ref})$ over all datasets in function of k and λ are shown in Figure 4.6. It can be seen that the ratio of correct assertions increases in function of

Figure 4.5 Mean number of assertions over all datasets for different λ in function of k.



increasing k, which confirms the conclusions drawn from analyzing the TI indices. In order to assess the differences between the different measures of generality, the results are again analyzed by means of Friedman and Kendall's W testing. The null hypothesis in each of these tests postulates a significant difference among λ_1, λ_2 and λ_3. The results of the tests are shown in Table 4.7. It can be seen that there is a difference between the different λ (p-value 0.05) for $k \leq 17$. For $k = 1$, Kendall's W equals 1, implying that λ_2 is strictly better than λ_1 and λ_3 on all datasets. Additional Wilcoxon tests are performed to compare the three λ pairwise. As a result, the differences between λ_1 and λ_2 on the one hand, and λ_1 and λ_3 on the other hand, are significant for $k \leq 17$. In summary, the statistical analysis of COR confirms the analysis of the TI indices.

The mean NCF $(\mathcal{E}, \leq_a, \leq_{ref})$ over all datasets in function of k and λ are shown in Figure 4.7. The results of statistical analysis of NCF are omitted here as they completely confirm the previous analyses. The combination of the results from Figures 4.6 and 4.7 provide us with some interesting and important insights. First, the sum of NCF and COR is approximately 1 for all λ and for all k. As a result, the ratio of erroneous assertions must be approximately 0. Although the plot for ERR is omitted here, it is confirmed that the maximal measure for ERR

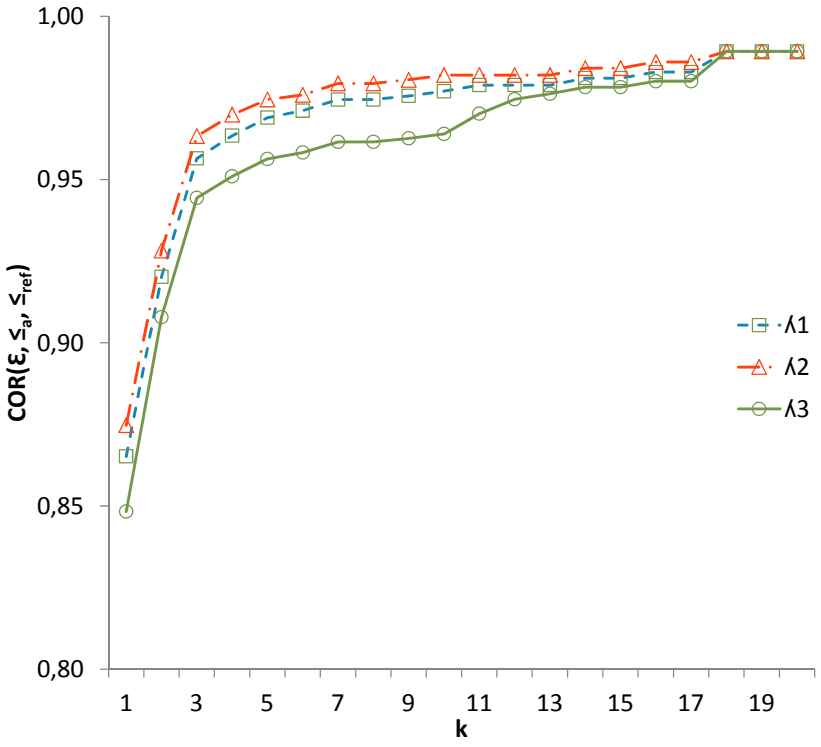
Table 4.6 Friedman mean rank (MR) and Kendall's W on TI indices for different λ in function of k.

k	MR λ_1	MR λ_2	MR λ_3	Sig.	W
1	1.99	3	1.01	0.000	0.992
2	1.84	2.99	1.17	0.000	0.859
3	2.31	2.68	1.01	0.000	0.923
4	2.22	2.68	1.09	0.000	0.842
5	2.12	2.68	1.19	0.000	0.759
6	2.32	2.58	1.09	0.000	0.850
7	2.32	2.58	1.09	0.000	0.850
8	2.32	2.58	1.09	0.000	0.850
9	2.32	2.58	1.09	0.000	0.850
10	2.32	2.58	1.09	0.000	0.850
11	2.17	2.43	1.40	0.000	0.555
12	2.02	2.27	1.71	0.000	0.280
13	2.02	2.27	1.71	0.000	0.280
14	2.02	2.27	1.71	0.000	0.280
15	2.02	2.27	1.71	0.000	0.280
16	2.02	2.27	1.71	0.000	0.280
17	2.02	2.27	1.71	0.000	0.280
18	2	2	2	-	-
19	2	2	2	-	-
20	2	2	2	-	-

is 0.0065 over all λ and all k. This result shows that the DOC method, regardless of k and λ, delivers an order relation from which very few erroneous assertions with respect to coreferent tuples detection and fusion of coreferent values in $\mathcal{E}$ are implied.

Secondly, the results provide a first experimental indication that the DOC method is preferential over a predefined and static taxonomy, e.g., in the context of fusion. The support for this claim lies in the following explanation. The assertions evaluated by COR and NCF in Figures 4.6 and 4.7, are assertions required during the fusion of coreferent tuples. In other words, an assertion on two values x and y is only evaluated if at some point, x and y need to be sorted during fusion of values of attribute a. From this point of view, the results show that for $k = 1$ and λ_2, more than 10% of the assertions made by $\leq_a$ are not made by the ground truth $\leq_{ref}$ and are at the same time not falsified by $\leq_{ref}$. Assertions that are not made, imply a higher risk of tie breaking. Because of the importance of this claim, it will be further investigated in Chapter 5, where $\leq_a$ is evaluated on its usability in the task of fusion.

Figure 4.6 Mean correct assertions $\mathrm{COR}(\mathcal{E}, \leq_a, \leq_{ref})$ over all datasets for different λ in function of k.



Conclusion. This experiment showed that parameter k can be used to control the quality of $\leq_a$ in terms of correct and non confirmed assertions. However, k should be chosen sufficiently low to guarantee that enough assertions are made. There appears to be a consistent trend in the accuracy of the different measures of generality, yielding λ_2 as the best measure. The statistical significance of the differences between the various λ fades with increasing k. It is shown that the assertions made by a DOC generated $\leq_a$, are either correct or non confirmed with respect to a ground truth taxonomical structure, but hardly ever falsified.

4.4.3 Stability under varying $\mathcal{E}$

Goal. In this experiment, the stability of DOC is evaluated under varying $\mathcal{E}$.

Procedure. For each of the seven datasets, $\mathcal{E}$ is randomly sampled without replacement in order to create subsets of the original partition. More specifically, clusters are randomly left out of the original partition in order to obtain a reduced

Table 4.7 Friedman mean rank (MR) and Kendall's W on $\mathrm{COR}\,(\mathcal{E}, \leq_a, \leq_{ref})$ for different λ in function of k.

k	MR λ_1	MR λ_2	MR λ_3	Sig.	W
1	2	3	1	0.000	1
2	2	2.99	1.01	0.000	0.992
3	2.22	2.68	1.09	0.000	0.842
4	2.22	2.68	1.09	0.000	0.842
5	2.22	2.68	1.09	0.000	0.842
6	2.32	2.58	1.09	0.000	0.850
7	2.32	2.58	1.09	0.000	0.850
8	2.32	2.58	1.09	0.000	0.850
9	2.32	2.58	1.09	0.000	0.850
10	2.32	2.58	1.09	0.000	0.850
11	2.17	2.43	1.40	0.000	0.555
12	2.02	2.27	1.71	0.000	0.280
13	2.02	2.27	1.71	0.000	0.280
14	2.02	2.27	1.71	0.000	0.280
15	2.02	2.27	1.71	0.000	0.280
16	2.02	2.27	1.71	0.000	0.280
17	2.02	2.27	1.71	0.000	0.280
18	2.02	2	2	-	-
19	2.02	2	2	-	-
20	2.02	2	2	-	-

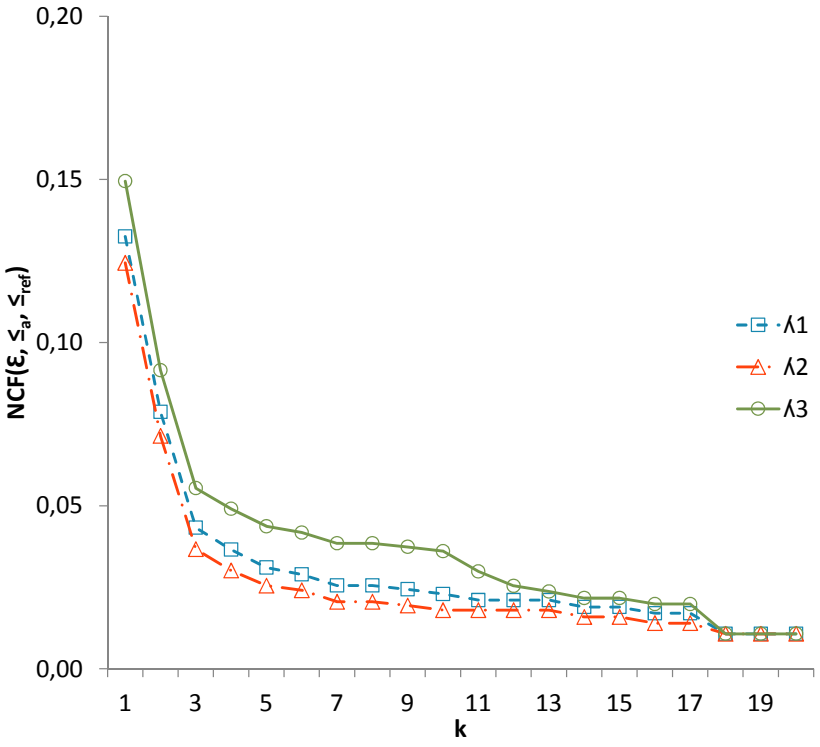
partition with approximately the same distribution of values as the original one. Sample sizes range from 10% up 100% of the original partition, in steps of 10%. For each sample, the DOC method is applied and evaluated. For each sample size, the procedure is repeated 100 times in order to obtain mean evaluations. Evaluation is done w.r.t. the same ground truth as in Section 4.4.2. In this experiment, k is assigned a fixed value: cases $k = 1$ and $k = 5$ are inspected.

Evaluation metrics. Local evaluation metrics NCF, COR and ERR (Section 4.4.2) are used to evaluate the obtained order relations.

Results and discussion. The experiment differentiates the seven datasets into two categories, which are discussed below. The first category includes four datasets: R-NL, R-EN, R-FR and R-DE. On each of these datasets, the stability experiment resulted in similar observations. Therefore, the discussion is limited to dataset R-NL. For this dataset, the mean correct assertions over all 100 samples in function of the sample size, are shown in Figure 4.8 ($k = 1$) and Figure 4.9 ($k = 5$). The whiskers indicate the standard deviation of COR over the samples.

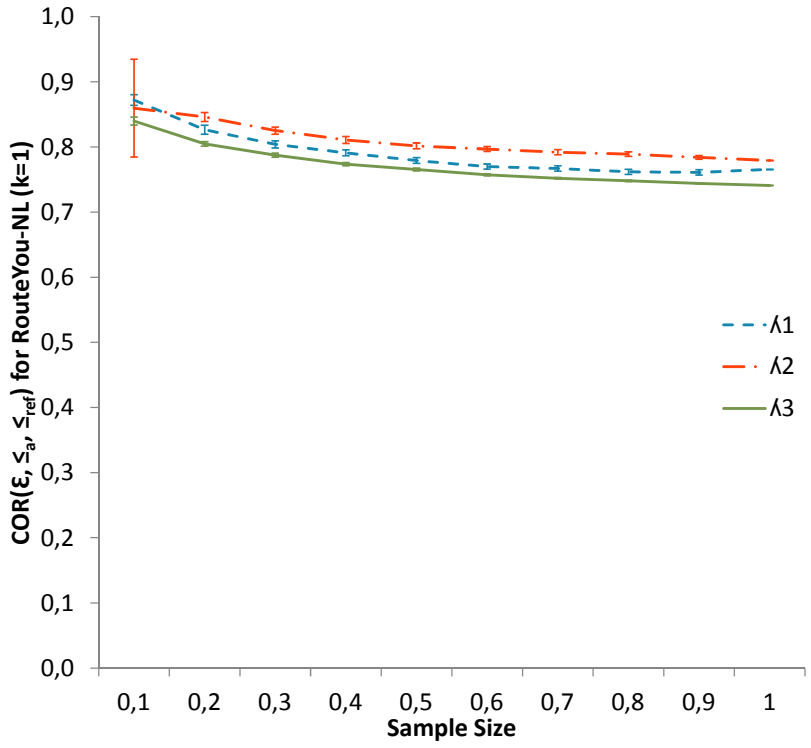
It can be observed from Figures 4.8 and 4.9, that for both $k = 1$ and $k = 5$,

Figure 4.7 Mean non-confirmed assertions $\mathrm{NCF}(\leq_a, \leq_{ref})$ over all datasets for different λ in function of k.



the variance in correct assertions is high for small samples. This is especially the case for λ_2. However, the mean percentage of correct assertions tends to be higher for smaller samples, than for bigger samples. This can be explained by the typical skew distribution of values (Figure 4.3). Due to this distribution, the more difficult cases corresponding to low frequent values, are likely to be filtered out. As such, a smaller sample size yields a slightly higher percentage of correct assertions. Despite this large variance, the four datasets under consideration are characterized by the fact that the mean $\mathrm{ERR}(\mathcal{E}, \leq_a, \leq_{ref}) \approx 0$, with standard deviation close to zero. This is observed for all λ and for all sample sizes. The implication thereof is that the non-stability of the DOC method does not result in erroneous assertions for these datasets. The opposite is observed for datasets R-ES, R-IT and restaurant, in particular for λ_1, as exemplified in Figure 4.10 for R-ES ($k = 1$). The cause of this large amount of erroneous assertions can be understood by recalling the histogram for R-ES 'category' values in Figure 4.3. It can be observed in this histogram that two values together account for 90% of the data. This implies that many assertions

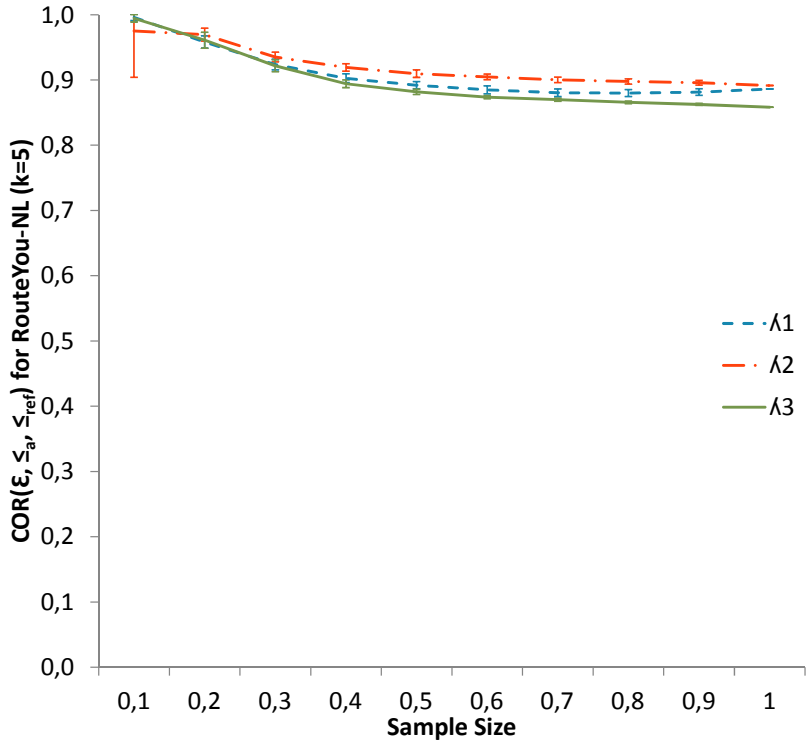
Figure 4.8 Mean correct assertions $\mathrm{COR}(\mathcal{E}, \leq_a, \leq_{ref})$ over 100 samples for different λ in function of sample size where $k = 1$.



will involve these two values. However, it can also be seen that their frequencies are close to each other, implying that their frequency rank can switch easily in a sample. Because this frequency rank is precisely the basis of λ_1, there is a high probability that assertions about these two values are erroneous in the context of a sample. In this sense, λ_3 is a more stable alternative for λ_1, because of the usage of frequency ranking on the *whole dataset*.

Conclusion. It is investigated to what extent DOC is stable if the number of clusters reduces, while maintaining the distribution of coreferent tuples. It is shown that λ_1 and λ_2 become unstable for very small partitions (i.e., large variance). For λ_2, the ratio of erroneous assertions remains low under unstable conditions. For λ_1, this is not the case and a significant amount of erroneous assertions is observed. Finally, λ_3 is found to be a stable measure of generality.

Figure 4.9 Mean correct assertions $\mathrm{COR}(\mathcal{E}, \leq_a, \leq_{ref})$ over 100 samples for different λ in function of sample size where $k = 5$.

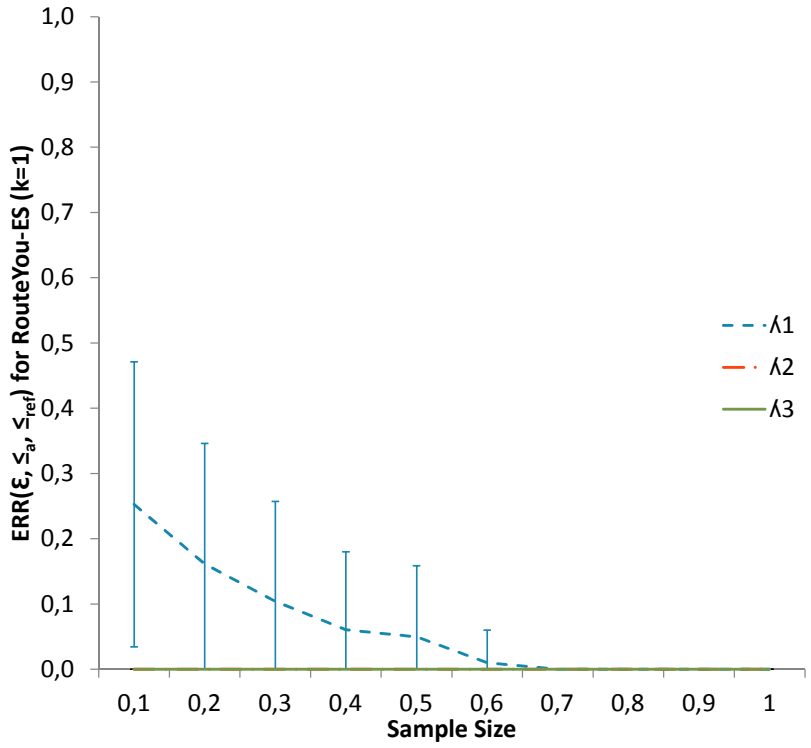


4.4.4 Impact on the coreferent tuples detection

Goal. In the last experiment, an impact of an order relation $\leq_a$ generated by the DOC method on the detection of coreferent tuples is evaluated. The influence of the parameter k is investigated.

Procedure. First of all, for each of the seven datasets, the DOC method is applied to the 'category' attribute which gives the order relation $\leq_a$. Next, the given Δ-partition $\mathcal{E}$ is an input for this experiment (as a reminder, coreference of tuples in Δ-partition is based on values of all attributes except the 'category' attribute). In this experiment, for each Δ_i in Δ-partition $\mathcal{E}$, tuples in Δ_i are considered as coreferent tuples if and only if their values of the 'category' attribute are considered as coreferent. Thus, values of the 'category' attribute are compared by two different coreference measures in order to investigate their influence on the coreferent tuples detection. Two coreference measures are investigated. For the first measure, denoted as $\approx$, values of the 'category' attribute are compared without any addi-

Figure 4.10 Mean erroneous assertions $\mathrm{ERR}(\mathcal{E}, \leq_a, \leq_{ref})$ over 100 samples for different λ in function of sample size where $k = 1$.



tional knowledge and only similarities between lexical forms of these values are considered. They are compared by edit (Levenshtein) distance method [12] which is defined as the minimal number of edit operations which are needed to transform one string into another. The Levenshtein distance based approach employs predefined threshold (equal 0.5) on the normalized number of edit operations needed, to decide if the coreference occurs. The second measure claims that two values are coreferent if there exists a relation between them in the order relation $\leq_a$.

Evaluation metrics. The quality in this experiment is evaluated using two standard measures of recall and precision, and the ground truth is the reference order relation $\leq_{ref}$. The precision is a fraction of detected real coreferent tuples among all detected tuples, the recall is a number of detected real coreferent tuples divided by the number of all real coreferent tuples in Δ-partition $\mathcal{E}$. The real coreferent tuples are tuples of which the 'category' attribute values exist in the reference order

relation $\leq_{ref}$. Thus, the following two evaluation metrics are used:

$$\mathrm{PRE}\left(\mathcal{E}, \bowtie, \leq_{ref}\right) = \frac{\sum_{\Delta \in \mathcal{E}} \#\text{detected real coreferent tuples}}{\sum_{\Delta \in \mathcal{E}} \#\text{all detected tuples}}$$

$$\mathrm{REC}\left(\mathcal{E}, \bowtie, \leq_{ref}\right) = \frac{\sum_{\Delta \in \mathcal{E}} \#\text{detected real coreferent tuples}}{\sum_{\Delta \in \mathcal{E}} \#\text{all real coreferent tuples}}$$

where $\bowtie$ is the $\approx$ coreference measure or the $\leq_a$-based measure respectively.

Results and discussion. Figure 4.11 and 4.12 show mean precision and recall of the coreferent tuples detection over all seven datasets, in function of the cut-off threshold k. Like in the previous experiments, higher values of k indeed filter out incorrect information. Of course, by increasing k, the total number of detected coreferent tuples decreases, meaning that higher k leads to *less*, but *more correct* coreferent tuples. From Figure 4.11, it can be learned that the first and the second coreference measure gives almost the same precision. However, recall which is presented in Figure 4.12 is definitely much more better for $\leq_a$-based coreference measure than for the $\approx$ coreference measure. It can be concluded that the order relation $\leq_a$ has strong advantages over string based comparison method in the coreferent tuples detection of tuples with categorical attribute.

Figure 4.11 Mean precision $\mathrm{PRE}\left(\mathcal{E}, \leq_a, \leq_{ref}\right)$ and precision syntax $\mathrm{PRE}\left(\mathcal{E}, \approx, \leq_{ref}\right)$ over all datasets in function of k.

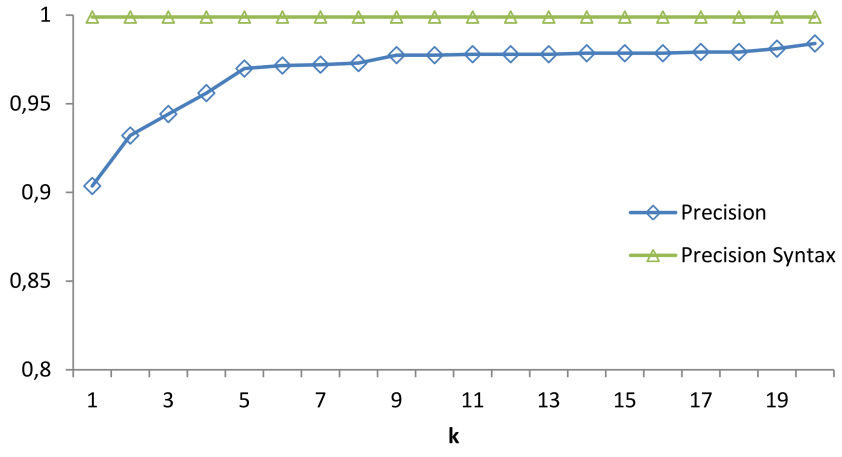
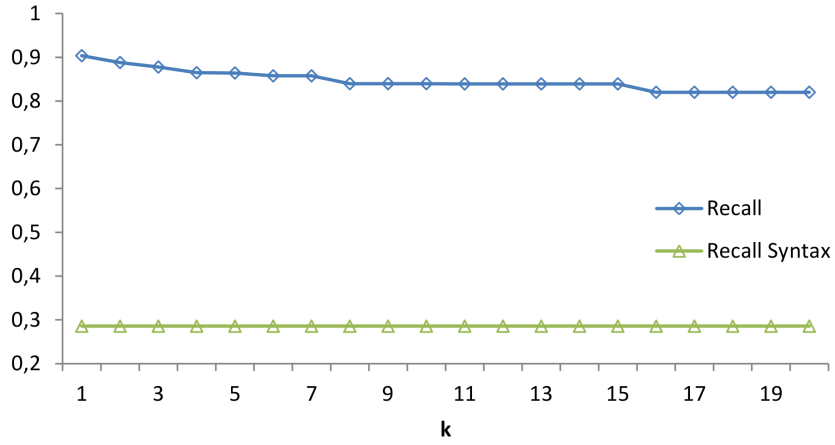


Figure 4.12 Mean recall $\mathrm{REC}(\mathcal{E}, \leq_a, \leq_{ref})$ and recall syntax $\mathrm{REC}(\mathcal{E}, \approx, \leq_{ref})$ over all datasets in function of k.



4.4.5 Concluding remarks

It has been shown that parameter k can be used to steer the quality of the generated order relation. For low k, a "larger" order relation is produced, usually containing a certain amount of *non-confirmed* assertions. For higher k, those non-confirmed assertions are filtered out. Three measures of generality λ have been tested. It is found that λ_2 provides the order relation with highest quality, followed by λ_1 and λ_3 in that order. If the partition size becomes very small, λ_1 and λ_2 can shown unstable behavior, while λ_3 is more resilient to small partition sizes. In addition, λ_3 provides more assertions.

4.5 Conclusions

In this chapter, the novel Dynamical Order Construction (DOC) algorithm has been proposed as a way to construct a proper order relation, which is either unknown or difficult to obtain, in an automated fashion, upon observing the coreferent data. Novel generality measures to specify the order, which are based on the frequency of values, are closely investigated. The behaviour of the DOC algorithm has been evaluated widely on several real life datasets, gaining us with valuable insight in the influence of the different parameters of our approach. Moreover, is was shown how the proposed approach has benefits with respect to a fixed taxonomy and string based comparison method in the coreferent tuples detection of tuples with categorical attribute.

On the one hand, the DOC algorithm constitute an answer to the first part of third research question which concerns dynamically constructed knowledge base

and its impact on the detection of coreferent tuples. On the other hand, the next chapter responds to the second part of third research question. Namely, the impact of the DOC algorithm on the data fusion in homogeneous and heterogeneous data collection is studied. More specifically, the novel *balanced* fusion function for attributes of which the values can be sorted by means of an order relation that reflects a notion of generality is proposed.

5

Dynamically constructed order relation in data fusion

The following publications have been based on the contents of this chapter:

- A. Bronselaer, M. Szymczak, S. Zadrożny, and G. De Tré, "Dynamical order construction in data fusion," Information Fusion. (Under review)

5.1 Introduction

In this chapter, we investigate fusion functions for attributes of which the values can be sorted by means of an order relation that reflects a notion of generality. It is shown that providing such an order relation a priori, let alone keeping it up-to-date, is a costly operation. Therefore, the Dynamical Order Construction (DOC) algorithm, which is proposed in the previous chapter, is used in the context of data fusion. Such order relations can be immediately deployed in a framework of selectional fusion functions, which are fusion functions that adopt the sort-and-select principle. These fusion functions are investigated closely in terms of their selection strategies and tie breaking mechanisms. An experimental evaluation of our method shows the influence of the parameters and the benefits with respect to using a fixed and predefined taxonomy.

5.1.1 Problem illustration

In the past decades, the handling of coreferent data has gained a lot of attention in the literature [108]. Usually, a solution to the problem of coreferent data comprises two distinct steps: (i) detection and (ii) fusion. The first step deals with the question "Are two descriptions referring to the same entity?" and has been the topic of much research throughout the past decades [2]. The second step deals with the question "How can coreferent descriptions be combined?". Hereby, there are several possibilities in how the result of such a combination must look like [109]: it could be a single description or a set of descriptions that contains as little redundant information as possible. In addition, there are several ways in how the result of fusion is used: it could be used to replace coreferent information in the database (possible using data lineage to link back to original value) or it could be stored in a table/view where it can be retrieved later on, for example to support coreferent free query results. Within the scope of this chapter, a contribution is made in solving the fusion step by proposing a novel fusion function that combines multiple descriptions into a single description using the dynamically generated order relation which has been proposed in the previous chapter.

As a running example throughout this chapter, we consider again a relational database of Points Of Interest (POIs). Provided that the tuples between horizontal lines shown in Table 4.1 have been detected as coreferent tuples, the next step in data cleansing is to combine them into one tuple that best represents the information about the referred location. As we will formalize in the following and we mention in the Introduction Chapter 1, this is usually done by projecting the tuples over their attributes and processing each attribute separately. For the POI name, a selection function which chooses the most frequent name or the longest name can be considered. For longitude and latitude, a possible solution is to apply median function. However, for the POI category, finding a representative value is less trivial, what we mention in the Introduction Chapter 1, because POI category variations are caused by *subjectivity*. In this setting, the taxonomical connection between the values of POI category can be used as a basis for fusing them. In order to deploy such a strategy, a number of challenges occur that deserve attention:

- How do we cope with the fact that, in many cases, the taxonomical structure that connects the values of an attribute is not known to the fusion function?
- How do we induce a fusion function from a generated taxonomical structure?
- Should the fusion be biased towards more specific or more general information?
- How to proceed with the cases where there is no explicit information on general-specific relation between a pair of values?

Within the remainder of this chapter, these questions shall be answered by developing a framework for fusion functions in which order relations can be generated automatically by the DOC algorithm that is proposed in Chapter 4.

5.1.2 Contributions

With respect to the problems given above, the following important contributions are made by this chapter. An application of the DOC algorithm (see Chapter 4) is proposed to support fusion of coreferent values for an attribute of which values may need semantical comparison and can be sorted by means of an order relation that reflects a notion of generality. Our approach has the advantages that there is no need for a priori taxonomical knowledge on the attribute domain and that the order relation automatically adapts to the values in the dataset. Moreover, in the context of data fusion a new strategy for selection is proposed called *balanced* selection and, in that context, tie breaking is studied. The behavior and (dis)advantages of our methods are experimentally investigated and validated on real life datasets.

5.1.3 Outline

The remainder of this chapter is organized as follows. In Section 5.2, an extensive overview of work related to the topic of this chapter is provided. Next, in Section 5.3, a framework of fusion functions is introduced in order to formalize the problem that is studied here. In Section 5.4, the usage of the DOC method, which has been presented in Chapter 4, in the context of fusion functions is investigated and selection strategies are evaluated. Hereby, a discussion on the role of tie breaking is given. In Section 5.5, an experimental study of the proposed methods and techniques is reported. Finally, Section 5.6 summarizes the most important contributions of this chapter.

5.2 Related work

In the past decades, the problem of fusing coreferent data in (relational) databases has been studied by many authors. In the first place, a number of authors consider the usage of relational operators in order to remove redundant data. Legaria et al. [110, 111] use the “union”-operator followed by removal of subsumed tuples. Yan et al. [112] propose the “match join”-operator which is a combination of union and join. Greco et al. [113] have investigated a similar approach. Bleiholder et al. [109] have proposed the “fuse by”-clause as an extension of the SQL syntax to support redundancy removal operations. Further development of that approach is reported in [114–116].

Apart from the usage of relational operators, some authors have been investigating stand-alone integration systems. Bilke et al. [117] and Naumann et al. [13]

have proposed an integrated system (HumMer) that allows the semi automated integration of heterogeneous data sources. It uses three steps of data integration: schema matching, duplicate detection and data fusion. In their work, they mention several functions for resolving inconsistencies that can be interpreted in terms of fusion functions discussed later in this chapter. Motro et al. [118] have approached the problem of data fusion as a multi-dimensional optimization problem. In their framework called *FusionPlex*, Motro et al. propose to use a utility function that is a linear combination of six metadata dimensions in order to tackle the problem of data fusion. Whereas the approach by Motro et al. assumes the data to be stored in a relational database, other approaches weaken this assumption and also consider semi-structured data by providing a data transformation layer (wrappers) [119].

Some authors have investigated the use of taxonomies and/or ontologies in the scope of data integration. In [120], a classification of possible semantical conflicts in (heterogeneous) databases is presented. Lu et al. [28] have investigated the automated construction of arithmetic-based conversion functions for numerical data. In their work, Lu et al. adopt correlation analysis for conflict detection and linear regression for conflict resolution. The resulting conversion functions are shown to allow for a mapping between different monetary rating systems. The initial framework of conversion functions is further developed in [121], where context awareness is taken into account in order to enhance the conversion functions. Ram et al. [122] have developed SCROL: a standard ontology to facilitate semantic translations between heterogeneous databases. Such an ontology allows to detect both data level conflicts (e.g., representation, precision...) and schema level conflicts (naming conflicts, entity identifier conflicts...). A similar approach has been investigated by Liu et al. [123]. In [124], the usage of ontologies is studied to support conflict resolution in query languages for heterogeneous databases. Bleiholder et al. [109] consider the selection of the most general and the most specific value as two of their conflict resolution strategies. They mention explicitly the usage of an ontology to infer a ranking of values to be fused. Dong et al. [14, 125] and Berti Equille et al. [126] have investigated the impact of source dependence on data fusion. Their work focusses on "choosing a proper one" among multiple sources that need to be integrated. For further readings on the usage of ontologies in data integration, the reader is referred to the overview paper by Wache et al. [127].

To the best of our knowledge, none of the above approaches relies on a knowledge base (a taxonomy, an ontology or some other structure) that is constructed dynamically without human intervention. There are some authors focussing on the automated construction of ontologies but it is not applied for data fusion directly and they differ from the technique presented in this work in several ways as described in Chapter 4.

5.3 A Primer on fusion functions

In this section, a basic framework of fusion functions is described, thereby relying on an existing definition of fusion functions. Within this framework, a formal distinction between *selectional* and *compositional* fusion functions is made. It is shown how compositional fusion functions are composed from atomic fusion functions. Within the framework developed in this section, a formal description of a novel selectional fusion function will be defined in 5.4.

5.3.1 The basic framework

In the remainder of this chapter, a relational database $D = \{R_1, ..., R_n\}$ with a schema $\mathcal{D} = \{\mathcal{R}_1, ..., \mathcal{R}_n\}$ is considered. Let us now assume that there exists a relation R with schema $\mathcal{R}$ for which coreferent tuples have been found (see formal definition in the previous chapter in Section 4.3.1). Coreferent tuples are hereby understood as two or more tuples that are copies or describe the same real world entity in a different manner, where differences can originate from typographical errors, lack of standardization, missing data... Obviously, the relation of coreference among the tuples is an equivalence relation. In order to remove these coreferent tuples from R, a fusion function is considered. The definition from [128] is adopted here.

Definition 12 (Fusion Function [128])**.** *Let $\mathcal{R}$ be a schema. A fusion function* F *for $\mathcal{R}$ is defined by a function* $\mathrm{F} : \mathcal{M}(\mathrm{dom}(\mathcal{R})) \rightarrow \mathrm{dom}(\mathcal{R})$ *where $\mathcal{M}(U)$ denotes the space of all multisets defined on U.*

For a schema $\mathcal{R}$, a fusion function takes a *multiset* of tuples and maps that multiset onto one tuple. Of course, very often the multisets of tuples in question will be in fact regular sets but the generalization provided by the concept of multiset is at work when we consider fusion at the level of attribute values sets, as discussed later on. Moreover, such a more general approach makes it also possible to consider identical coreferent tuples originating from merging several databases. In [128], it is noted that an important class of fusion functions is that of *preservative* fusion functions. These are defined as follows.

Definition 13 (Preservation [128])**.** *Let $\mathcal{R}$ be a schema. A fusion function* F *for $\mathcal{R}$ is preservative if and only if* $\mathrm{F}(\Delta) \in \Delta$.

Informally, a preservative fusion function is bounded by the constraint that the result must be chosen from the input multiset. In practice, two important classes of fusion functions can be distinguished: *selectional* functions and *compositional* functions.

Selectional Fusion Functions. A selectional fusion function is a preservative[1] fusion function that relies on the assumption that there exists a partial *order* over $\mathrm{dom}\,(\mathcal{R})$, say $\leq_{\mathcal{R}}$, which allows to rank tuples in Δ. The value of a selectional fusion function F is given by selecting a tuple on a fixed position after sorting. Because $\leq_{\mathcal{R}}$ is a partial order, there may be multiple sorted sequences that satisfy $\leq_{\mathcal{R}}$. Therefore, tie breaking will be discussed in Section 5.4. The two most common selection strategies are to take the minimal or the maximal tuple in Δ according to the order $\leq_{\mathcal{R}}$.

Definition 14 ($\min$-$\leq_{\mathcal{R}}$ Fusion Function)**.** *Let $\mathcal{R}$ be a schema and $\leq_{\mathcal{R}}$ an order over* $\mathrm{dom}(\mathcal{R})$*. A* $\min$-$\leq_{\mathcal{R}}$ *fusion function* $\mathrm{F}_{\min}^{\leq_{\mathcal{R}}}$ *for $\mathcal{R}$ is defined by:*

$$\mathrm{F}\,(\Delta) = \min_{t \in \Delta} t. \tag{5.1}$$

Definition 15 ($\max$-$\leq_{\mathcal{R}}$ Fusion Function)**.** *Let $\mathcal{R}$ be a schema and $\leq_{\mathcal{R}}$ an order over* $\mathrm{dom}(\mathcal{R})$*. A* $\max$-$\leq_{\mathcal{R}}$ *fusion function* $\mathrm{F}_{\max}^{\leq_{\mathcal{R}}}$ *for $\mathcal{R}$ is defined by:*

$$\mathrm{F}\,(\Delta) = \max_{t \in \Delta} t. \tag{5.2}$$

Many well known fusion functions that are found in the literature can be cast into this simple framework of selectional fusion functions by choosing a suitable order $\leq_{\mathcal{R}}$. Some examples of such functions are *most/least recent* [13], *subsumption* [110] and *maximal utility* [129]. As a side comment, note that $\mathrm{F}_{\max}^{\leq_{\mathcal{R}}} = \mathrm{F}_{\min}^{\leq_{\mathcal{R}}^{-1}}$ where $\leq_{\mathcal{R}}^{-1}$ denotes the inverse relation of $\leq_{\mathcal{R}}$.

For instance, let us consider again 3 tuples in Table 4.1 which represent Belfry of Ghent. The most (least) recent function will select the newest (oldest) tuples depending on, e.g., the modification date. Whereas, the second tuple can be chosen by the subsumption function because this method removes redundant information and "POI" can be considered as a generalization of others categories. Finally, utility-based techniques combine different information and are based on knowledge about the *performance* of the source data, including features such as recentness, availability, cost and accuracy.

Compositional Fusion Functions. A compositional fusion function is based on the principle of divide-and-conquer by fusing the values of each attribute separately. The fused values for all the attributes are then composed into a final tuple. A particularity is that fusion of attributes is based on the *multi-projection* (cf., Def. 11) rather than the classical relational projection. The multi-projection of an attribute will be processed by means of an atomic fusion function.

[1] Mathematically seen, the class of preservative fusion functions is broader than the class of selectional fusion functions.

Definition 16 (Atomic Fusion Function)**.**
Let $\mathcal{R}$ be a schema. For $a \in \mathcal{R}$, an atomic fusion function F_a is defined as a fusion function over one-attribute relational schema, i.e. $\mathcal{R} = \{a\}$.

A fusion function F_a is called *atomic* because it operates on the lowest level of the relational model (i.e., an attribute). In general, a distinction can be made between (a) preservative and (b) non-preservative atomic fusion functions. Two examples of the first type are *coalesce* fusion (random selection of a non-NULL value) [129] and *majority* fusion [130] characterized by:

$$\mathrm{F}_a\left(\Delta[\![a]\!]\right) = \underset{v \in \mathrm{dom}(a)}{\arg\max}\left(\Delta[\![a]\!]\right)(v). \tag{5.3}$$

Examples of the latter case are average operators for numerical values and concatenation operators for character strings. With the concepts of 'multi-projection' and 'atomic fusion function' at hand, a compositional fusion function can be defined as follows.

Definition 17 (Compositional Fusion Function)**.**
Let $\mathcal{R}$ be a schema and let F_a be an atomic fusion function for $a \in \mathcal{R}$. A compositional fusion function F for $\mathcal{R}$ is defined by:

$$\mathrm{F}(\Delta) = \left(\mathrm{F}_{a_1}\left(\Delta[\![a_1]\!]\right), ..., \mathrm{F}_{a_k}\left(\Delta[\![a_k]\!]\right)\right). \tag{5.4}$$

It can be seen that a compositional fusion function is not bound to be preservative, as there is no guarantee that the composed tuple occurs in Δ.

5.3.2 Construction of atomic fusion functions

In the remainder of the chapter, we consider a compositional fusion function F for a schema $\mathcal{R}$. This means that, for each attribute $a \in \mathcal{R}$, an *atomic* fusion function F_a must be specified. In the remainder of this chapter, we restrict ourselves to atomic fusion functions that are *selectional* (Section 5.3.1). This choice is motivated by the properties possessed by such functions:

- Some attributes are represented using nominal or ordinal scales [131] allowing for defining a rather limited number of meaningful operations. In that case, an aggregation of input values becomes virtually infeasible and choosing one of the input values seems the only valid strategy for fusion.
- Selectional fusion functions possess some nice mathematical properties such as idempotence and self-identity [132] in the absence of ties.
- Using a selectional fusion function increases the a priori confidence that the fused value is *valid* (i.e., it is observed in the database before fusion).

Moreover, if values in Δ stem from different sources, the confidence in the outcome can be linked to the confidence in the source of its origin.

- Selectional fusion functions cover a wide range of fusion functions that are studied in the literature. For example, in their overview paper on data fusion [133], Bleiholder and Naumann discuss different conflict resolution functions for attribute values. It can be verified that most of these functions fit the definition of a selectional fusion function as given here.

Despite the above mentioned advantages, the usability of a selectional fusion function completely relies on the assumption that a *relevant* order relation $\leq_a$ is a priori available. Obviously, the popularity of selectional fusion functions partially stems from the fact that this assumption holds in many cases. As an example, for date/time information, the natural ordering is usually a good and relevant order relation. As another example, for (short) textual information, a reasonable (although simple) heuristic could be that the information contained by a string is proportional to the length of the string or the number of tokens in the string. In that case, a relevant order relation is string length or token count. As yet another example, the lexicographical order relation for short text labels might be a suitable candidate.

However, the case may be that a relevant ordering for the purpose of data fusion is not available. As explained in Introduction Chapter 1, in some cases, the relevant order should be inferred from a taxonomical structure that represents a generalization/specialization relation. It often occurs that this taxonomy is unavailable at the time of data fusion. This is caused by a variety of reasons:

- The simplest reason is that the taxonomy is simply not (explicitly) known. In some cases, maintenance of a taxonomy can be an expensive operation for the database administrator, so that no taxonomical structures are available for utilization in data fusion.

- If there is a taxonomy at hand, it might be difficult to relate entries in this taxonomy to values in the database. More specifically, if the available taxonomy is not used for inserting values in the database, there is a high probability that values stored in the database differ from the ones used in the taxonomy.

- In some cases, data fusion operations deal with information that stems from different sources. As an example, the famous "Restaurant" dataset [9] (Section 5.5) features two lists of restaurants, taken from two on-line guides: Fodor and Zagat. In both cases, the type of the restaurant is specified. However, only 25% of the type values occur in both lists. This indicates that both restaurant guides use a different taxonomy to classify restaurants. In such a case, fusion requires some kind of translation between both sources

of values (for further studies on the translation in data fusion, namely value mapping, the reader is referred also to Chapter 6 where an order relation is used to established a semantical mapping between attributes values).

As a solution to these problems a method (namely the DOC algorithm) is proposed in Section 4.3 to construct an order relation $\leq_a$ in a dynamical manner. More specifically, if we are confronted with a data fusion problem in which one of the attributes requires fusion based on a taxonomy then such a taxonomy is constructed *automatically*, based on the observed tuple coreferency. The advantages of this method are clear: (i) the taxonomy is always known, (ii) contains solely values that are observed in the data and (iii) dynamically translates values that could stem from different taxonomies. In what follows, this method of automated taxonomy construction shall be used in the context of selectional fusion functions.

5.4 Atomic DOC-driven fusion functions

In this section, the DOC method (see Section 4.3) is used to construct atomic fusion functions. We hereby restrict ourselves to fusion functions that are *preservative* (Definition 13). More specifically, a selectional fusion function will be considered, meaning that, next to the generated order relation, a suitable selection strategy is needed.

First, some heuristics are provided that can aid in the recognition of attributes for which DOC-driven fusion is suitable. Next, some selection strategies are discussed that bypass simple minimal or maximal selection. Finally, the problem of tie breaking upon selection is discussed.

5.4.1 Recognition of suitable attributes

With the DOC method established, we are able to construct an order relation $\leq_a$ to backbone an atomic fusion function F_a for an attribute a. In order to do so, it should be clear that the DOC method can not be applied to every attribute. In fact, the usage of a DOC-driven fusion function requires a specific type of attributes. The fully automated recognition of such attributes is near to impossible. However, some heuristics can be used to recommend or discourage the usage of a DOC-driven function to a human supervisor. Below, a non-exhaustive list of such heuristics is presented.

- **Subjectivity.** Perhaps the most important feature that an attribute must possess, is that its values provide an answer to a subjective question. An example thereof is the "category" attribute for POIs (Table 4.1). Indeed, when several persons are asked to provide the category of a POI, they might provide a different value because of a different opinion or view to the matter.

The same is true for the category of a restaurant, the genre of a movie... Despite of the fact that it is a strong indicator, subjectivity is however difficult to measure.

- **Data structure.** Useful information can be derived by inspecting the data structure of an attribute. Typically, numerical and time/date data structures are not suitable for DOC-driven fusion. Enumeration data or textual data constrained by *check* constraints[2] are usually more suitable candidates.

- **Data statistics.** Statistics of the data can provide useful information during inspection of the attribute. For example, if an attribute has a very high percentage of unique values, the DOC method will not perform well, because it explicitly relies on the differences in the frequency of values to construct an order relation. In addition, attributes on which the DOC method performs well, typically have values with a skew frequency distribution.

- **General purpose ontologies.** Regardless of which structure is best suited for the actual fusion, a general purpose ontology like YAGO ([134]) can be used to (partially) detect some hierarchical connections between values of an attribute a. Such connections are a strong indication that the DOC method is applicable.

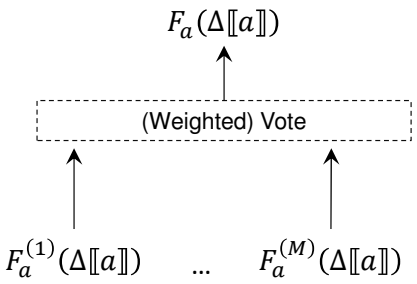
5.4.2 Selection strategies

Once we have established that an attribute a is suitable for DOC-driven fusion, a selection strategy is needed in order to obtain a selectional fusion function for a. Recall from Section 5.3 that a selectional fusion function adopts a sort-and-select strategy to perform fusion. As explained in Section 5.3, selection is often done by taking minima or maxima of the sorted sequence. Although these selection strategies might be very simple, they can prove to be useful in an *ensemble* of fusion functions. Such an ensemble for an attribute a considers several atomic fusion functions F_a and adopts a voting procedure to take the final decision.

The principle of an ensemble is depicted in Figure 5.1. Hereby, the atomic fusion function F_a consults M *ground functions* $\mathrm{F}_a^{(i)}$ and computes a result by performing a (weighted) vote on the results of them, where weights can express preferences for particular ground functions. Assuming that the ground functions are selectional, the idea of an ensemble is that simple selection strategies can be used by the ground functions if M is sufficiently high and the ground functions are chosen in a sensible way. A DOC-driven fusion function can then be chosen as one of the ground functions, combined with other selectional functions that rely on

[2] A check constraint is a type of integrity constraint (e.g., in SQL) which specifies a requirement that must be met by each row in a database table

Figure 5.1 Ensemble fusion function for attribute a.



source preferences [14, 125] or other order relations (e.g. based on ontologies). In the remainder of the chapter, ensemble functions will not be investigated deeper.

In the case where we wish to construct a fusion function that solely relies on an order relation generated by the DOC-method, a more advanced selection strategy than simple minimal or maximal selection is desired. This is mainly caused by the fact that selection is problem dependent. To clarify this, Figure 5.2 shows three examples in which an order relation $\leq_a$ is represented as a directed graph. The nodes of this graph correspond to values from $\mathrm{dom}(a)$. Edges depart from the more general value and points towards the more specific value. For reasons of clarity, the Hasse diagram is shown. In all three examples, the nodes that are marked in bold indicate the values that need to be fused. In the leftmost panel of

Figure 5.2 Selection strategy is problem-dependent.

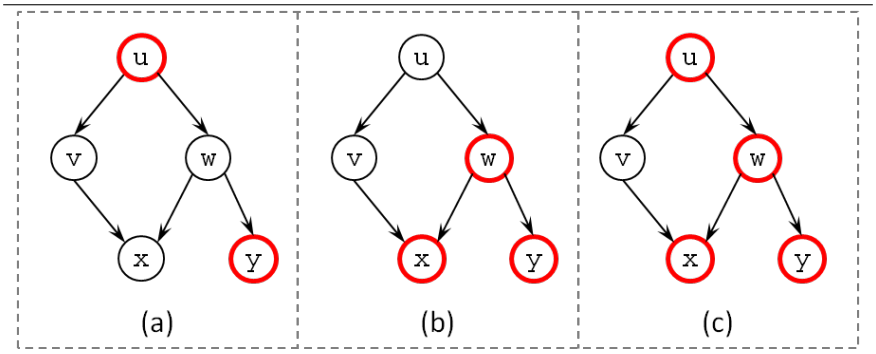


Figure 5.2, two values u and y need to be fused for which we know that $y \leq_a u$. In this case, two coreferent values are observed, where it can be inferred from $\leq_a$ that one is more general than the other. Minimization of information loss leads us to the decision that the more specific value should be preferred here. In the center panel

of Figure 5.2, three values w, x and y need to be fused for which we know that $x \leq_a w$ and $y \leq_a w$. In this case, evidence is given that the concept in question is a specialization of w, but it is unclear which one. In addition, choosing either x or y, implies a certain information loss. Therefore, it might be better to select the more general value in this second example, leading to the usage of $\max\text{-}\leq_a$ fusion function. Finally, in the rightmost panel, a mixture of both previous examples is observed. Choosing u implies an unnecessary loss of information, as there is no ambiguity about the specialization of u. As a result, w should be the preferred output over u. However, there is ambiguity about which specialization of w should be chosen. Therefore, in this third example, w should be selected as output of the fusion function. Clearly, to obtain this behaviour, a more sophisticated selection criterion is required than the ones introduced in Section 5.3.

In order to establish a selection criterion, a generalization of the strategies explained in the examples of Figure 5.2 is used. In each of the examples, the most specific value that was comparable to all other values, was chosen as result of the fusion function. This criterion is formalized as follows.

Definition 18 (Totality)**.** *Let $\leq_a$ be a partial order relation over $\mathrm{dom}(a)$ and assume $S \subseteq \mathrm{dom}(a)$. A value $v \in \mathrm{dom}(a)$ is called* total *w.r.t. S (denoted $S \models_{\leq_a} v$) if and only if:*

$$\forall v' \in S : v \leq_a v' \vee v' \leq_a v. \tag{5.5}$$

In words, a value v is total w.r.t. a set, if it can be compared with each value in that set. With the concept of totality at hand, the strategy of the balanced selection can be defined as below.

Definition 19 (Balanced selection)**.** *A selectional fusion function F_a is balanced if it returns the minimal value $v \in \mathrm{dom}(a)$ w.r.t. $\leq_a$, that is total w.r.t. $\Delta[\![a]\!]$. Formally:*

$$\mathrm{F}_a\left(\Delta[\![a]\!]\right) = \min\{v : v \in \Delta[\![a]\!] \wedge \Delta[\![a]\!] \models_{\leq_a} v\}. \tag{5.6}$$

Such a fusion function is denoted as $\mathrm{F}_{\mathrm{B}}^{\leq_a}$.

It can be verified that the balanced fusion function yields the desired outcomes for each of the examples shown in Figure 5.2. If we use balanced selection to fuse the "category" attribute for clusters shown in Table 4.1 and using $\leq_a$ as shown in Figure 4.2, we obtain the results as shown in the upper part of Table 5.1. For these examples, there is no difference between balanced and minimal selection. For values "Hist. Bld." and "Restaurant", no decision can be made because these values are incomparable. The lower part of Table 5.1 shows an example for which there is a difference between minimal and balanced selection. Note that for this example, the outcome for minimal selection is not entirely determined, because values "Monument" and "City" are incomparable under $\leq_a$. Therefore, both values are

Table 5.1 Fusion results for $\leq_a$ from Figure 4.2 and coreferent tuples from Table 4.1 with $a =$"type".

$\Delta[\![a]\!]$	$\mathrm{F}_{\mathrm{B}}^{\leq_a}(\Delta[\![a]\!])$	$\mathrm{F}_{\mathrm{min}}^{\leq_a}(\Delta[\![a]\!])$
{Tower, POI, Monument }	Tower	Tower
{Hist. Bld., Restaurant }	Hist. Bld./ Restaurant	Hist. Bld./ Restaurant
{City, POI }	City	City
{Monument, POI }	Monument	Monument
{City, POI, Monument }	POI	Monument/City

possible outcomes in case of minimal selection. In case, of the balanced selection, value "POI" is the only total value and is returned as result.

We can now show some interesting properties of the balanced selection.

Property 1. *If all values in $\Delta[\![a]\!]$ are total with respect to $\Delta[\![a]\!]$, then:*

$$\mathrm{F}_{\mathrm{B}}^{\leq_a}(\Delta[\![a]\!]) = \mathrm{F}_{\mathrm{min}}^{\leq_a}(\Delta[\![a]\!])\,. \tag{5.7}$$

Proof. The proof follows immediately from the observation that, if all values in $\Delta[\![a]\!]$ are total, the balanced selection selects the minimum from $\Delta[\![a]\!]$. □

Property 1 shows that minimal selection is a special case of the balanced selection. This is illustrated in the leftmost panel of Figure 5.2, where all values that need to be fused are total and as such, the smallest one (i.e., y) is chosen. Property 1 supports the following rule of thumb behind the balanced selection: "If the most specific value is unambiguously defined, then that is the result of the fusion function.". Yet another way of putting it is that the balanced selection maintains a balance between yielding specific information and information that is not falsified by the knowledge at hand.

Property 2 (Uniqueness)**.** *If $\Delta[\![a]\!]$ contains one or more total values with respect to $\Delta[\![a]\!]$ then $\mathrm{F}_{\mathrm{B}}^{\leq_a}(\Delta[\![a]\!])$ is unique.*

Proof. The proof follows immediately from the observation that fusion selects the minimum among all *total* values. This minimum is uniquely determined, because total values can be mutually compared by definition and $\leq_a$ in Definition 18 is assumed to be a partial order and thus antisymmetric. □

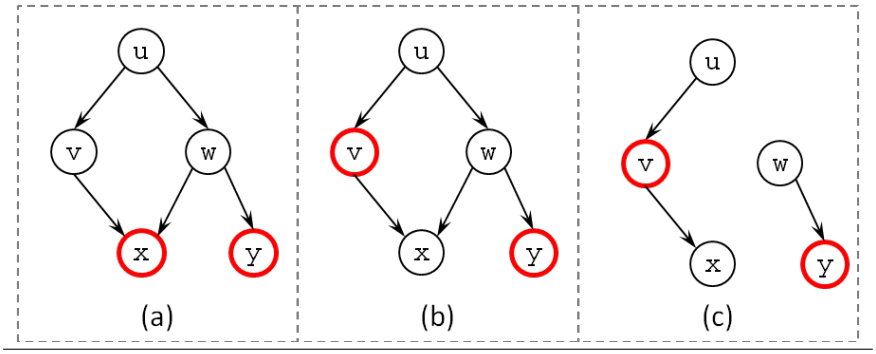
Property 2 shows an important advantage of the balanced selection, namely that it is uniquely determined. On the other hand, it reveals also an important disadvantage, namely that it might be undefined. Indeed, it is possible that none of the values that need to be fused is total. In that case, the set of total values in $\Delta[\![a]\!]$ is *empty* and the output of our fusion function remains unspecified. The following section is devoted entirely to an in-depth study of this problem.

5.4.3 Tie breaking

The problem with the proposed above balanced selection algorithm is the possible lack of total values. In other words, the algorithm does not yield a result when for each value in consideration, there exists another value to which it can not be compared. Such incomparability is an intrinsic aspect of a partial order relation (if not, it would be a total order relation). The problem where a partial order relation hinders the accuracy of a function is widely known as *tie breaking* and occurs in many disciplines (e.g., top-k queries [135, 136], social choice theory [137] ...). Within the scope of data fusion, the problem of tie breaking is encountered with many other selectional fusion functions. For example, when selecting the most recent information or the shortest string, ties can very well be met. In these cases, ties are usually handled by making a random choice between the candidate solutions. In the current setting, the problem of tie breaking is investigated in more depth.

Let us begin with providing some examples. In Figure 5.3, three cases are shown in which none of the values is total. As a result, application of the balanced selection strategy yields no result. In the leftmost panel, a case is shown in which two values are incomparable, but it is known that they are both a direct specialization of the same value (i.e., w). In the center panel, a similar case is shown, but with the difference that one of the values is not a direct specialization of the common ancestor. In the rightmost panel, two values are shown that have no common ancestor. A possible solution in any of these cases, is to resolve the incomparability between two values (i.e., breaking the tie), in order to create at least one total value.

Figure 5.3 Three examples of ties.



Before discussing possible solutions to the problem of tie breaking, let us first discuss the interesting question: “Should ties always be broken?” This question relates in a sense to the origin of ties. Let us clarify this with an example. In Table 4.1, there are two coreferent tuples describing the restaurant “Korenlei 2”. For each of these tuples, a different category is given. Suppose our order relation can

make no comparison between "Hist. Bld." and "Restaurant". Then the balanced selection will be undetermined. If the tie situation between these two values needs to be resolved, a relevant question is whether there exist reasons to choose one of these two values. Indeed, in a city like Ghent, it is common that restaurants are located in historical buildings, making both category values equally reasonable and simultaneously relevant. However, the data structure implies that each POI should have at most one category, forcing us to make a choice. In addition, the definition of fusion functions as given here (Section 5.3) also dictates that a single value should serve as the output of fusion. The current example illustrates that this assumption might not always hold.

Having this said, let us assume for the remainder of this section that a relevant tie breaking decision does exist. In the discussion of tie breaking, two aspects are of interest. On the one hand, given that two values x and y are incomparable under $\leq_a$, choosing one of them in the framework of Definition 19 is equivalent to assuming $x \leq_a y$ or $y \leq_a x$ as additional input information. On the other hand, if we consider that $\leq_a$ is a (part of a) knowledge base, caution must be taken when modifying this knowledge base. Breaking too many ties might render $\leq_a$ less useful if tie breaking relies on strategies that are not sufficient trustworthy. Both of these aspects are discussed below.

Let us first discuss the aspect of decision rules. Suppose two values x and y have been identified that are incomparable under $\leq_a$ and it is required to break this tie. This means that either $x \leq_a y$ or $y \leq_a x$ must be assumed. The choice usually relies on heuristics. Four heuristics, three of them proposed in literature, are listed below:

- **Majority [130].** A first decision rule selects the value with the highest multiplicity in $\Delta[\![a]\!]$. Although this rule provides s a good heuristic, it is equivalent to a random choice if $|\Delta[\![a]\!]| = 2$. Unfortunately, for real datasets such a case may prevail. For example, for the datasets used in Section 5.5, this is the case in 97% of the identified coreferent tuple clusters.

- **Source Preference [119, 125].** A second decision rule takes into account a preference of sources and selects the value that stems from the must reliable source.

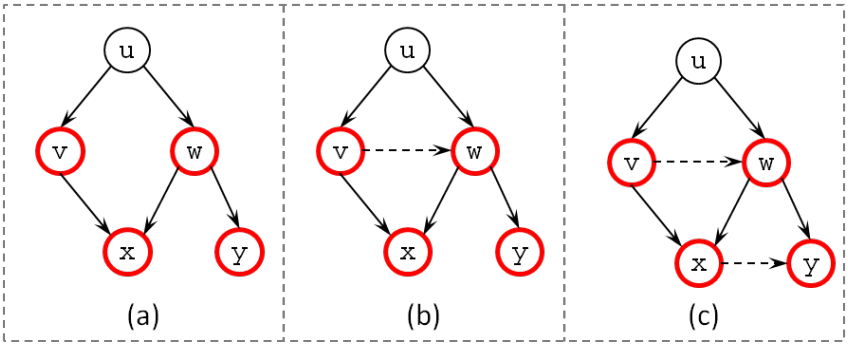
- **Random choice [129].** A third decision rule is to break the tie by making a random choice.

- **Structural Inference.** Finally, a proposal done in this chapter, it to use the structure of $\leq_a$ to break a tie. In that case, when two values are incomparable, the tie is broken by deriving some statistics for both values from $\leq_a$. An example thereof is to look at other values to which they can be compared. In the center panel of Figure 5.3, it can be observed that v and y are both more

specific than u. However, y is also more specific than w, which is in turn more specific than u. If it can be assumed that the specificity is reflected in the number of *intermediate* values, then y can be considered more specific than v. Another heuristic could be to count the number of values that are more specific than each of the tied values. The more values that have higher specificity than v according to $\leq_a$, the more general v becomes. Structural inference has the advantage that more ties can be broken than other strategies like majority, but the usefulness of structural heuristics depends completely on the assumption that specificity can be derived from the structure somehow. We shall evaluate tie breaking by structural inference in Section 5.5.

The second aspect that needs discussion is the extent to which ties are broken. In order to explain this aspect, let us consider the example shown in the leftmost panel of Figure 5.4.

Figure 5.4 Influence of tie breaking on balanced selection (dotted line indicates broken tie).



In this example, four alternatives are given that need to be fused. It can be verified that none of these values is total w.r.t. the set of alternatives. As a result, the balanced selection is undetermined and tie breaking is required to find a solution. However, the important question that now occurs is: "Which ties should be broken?". Suppose we break the tie between v and w and suppose that the decision rule asserts $w \leq_a v$, then balanced selection will result in w (Figure 5.4, center panel). Suppose that the decision rule asserts (in addition to the first assertion) that $y \leq_a x$, then balanced selection will result in y (Figure 5.4, rightmost panel).

On the one hand, it can be reasoned that balanced selection aims at selecting information that is *as specific as possible*. In that reasoning, tie breaking should focus at breaking ties between the most specific values. In the example shown in Figure 5.4, this means that balanced selection should never select v nor w, because they are less specific than x and y. On the other hand, it can also be reasoned that balanced selection accepts only information that is *not falsified*. In that reasoning,

the decision rule should be paired with a measure of confidence, and this confidence should indicate a preference on which ties to break first. Such a measure of confidence can be an implicit part of the decision rule (e.g. the preference of a source, the multiplicity in the case of majority voting...), but can also be an independent mechanism. The example in Figure 5.4 illustrates that it is desirable to maintain a balance between specificity and confidence upon tie breaking. It should be taken into account that the requirement of a measure of confidence may introduce some difficulties. Either it could be not so reliable or it could be cumbersome to provide a measure of confidence explicitly. In that case, a better alternative is to perform tie breaking only on the most specific values.

5.5 Evaluation and discussion

In this section, an experimental evaluation of the proposed techniques is presented. The time complexity of the DOC method in the context of data fusion are evaluated and the balanced selection strategy is compared with the minimal selection strategy.

5.5.1 Datasets

Throughout the experiments, seven datasets are used which are detailed described in Section Evaluation in Chapter 4. As a reminder, the first dataset is the famous 'restaurant' dataset [9], which contains two lists of restaurants. One list of restaurants stems from the on line guide 'Zagat', while the other list stems from the on line guide 'Fodor'. Six other datasets are real-life datasets made available by the Belgian company RouteYou[3], which is an on line provider of cycling routes. Within the context of our experiments, the attribute of interest is the category of the restaurant, which specifies the cuisine that can be found in the restaurant, or the category of POI, which describes a function of the location.

5.5.2 Selection strategies

Goal. In this experiment, the DOC method is evaluated in terms of the behavior of the fusion functions that rely on $\leq_a$. Therefore, two selection strategies are evaluated: $\min$ selection (i.e. most specific value) and balanced selection (i.e. the most specific value that can be compared with all other values). These fusion functions are compared with two alternatives: majority voting and selectional fusion based on ground truth $\leq_{ref}$ (the same which is applied for experiments in Chapter 4). By taking into account this last function, confirmation is sought for the benefit of DOC over a fixed taxonomy.

[3] http://www.routeyou.com

Procedure. For each of the seven datasets, the partition $\mathcal{E}$ of coreferent tuples is considered. For each set of coreferent tuples Δ, a fusion function is applied to fuse the 'category' attribute and it is verified if this result is *uniquely determined*, that is: the solution *exists* (applicable for balanced selection) and is *unique*. As such, the percentage of unresolved fusion operations indicate the percentage of cases in which tie breaking is necessary. We evaluate selectional fusion supported by DOC for two selection strategies: minimal and balanced. Note that we consider only non-trivial cases, i.e. cases where there is only one possible value for category are excluded. We compare the results of our approach with (a) a fusion function that adopts the ground truth taxonomy for sorting and uses balanced selection and (b) a fusion function that performs a majority vote.

Evaluation metrics. For all fusion functions, the ratio of coreferent tuple clusters for which fusion is unresolved, is reported. A set of coreferent tuples Δ has unresolved fusion for attribute a, if $\mathrm{F}(\Delta[\![a]\!])$ has either no solution, or multiple solutions. In addition, for the fusion functions relying on $\leq_a$ or $\leq_{ref}$, we also report the ratio of unresolved fusion operations for which there is at least one value $v \in \Delta[\![a]\!]$ that is *missing* in $\leq$. With this second metric, we measure the extent to which an unresolved case is due to the fact that the knowledge base $\leq$ has insufficient information to make a decision. In case of $\leq_a$ generated by DOC, this can be caused by increasing k. In case of the ground truth $\leq_{ref}$, this can be caused by the fact that the ground truth taxonomy contains (to some extent) other values than those in the dataset.

Results and discussion. First, some summarized results are provided. Table 5.2 shows a comparison between the baseline methods and a balanced fusion function supported by DOC with the measure for generality λ_2 and the cutoff threshold $k = 1$ (see Chapter 4). This simple comparison yields some interesting observations. First, it can be seen that majority voting yields a very high number of unresolved fusion operations. This comes as no surprise if we recall that 97% of the cases are characterized by $|\Delta[\![a]\!]| = 2$ (Table 4.4). Because we consider only non-trivial cases (i.e., not all alternatives are equal), most cases come down to two alternatives with multiplicity 1, which yields a tie if we use majority voting. Second, there are three datasets (R-NL, R-FR and Rest) for which using the taxonomy yields a high number of unresolved fusion operations, whereas the DOC supported fusion always yields a very low number of unresolved fusion operations.

Third, the knowledge base generated by DOC always contains *all* values that are observed during fusion if $k = 1$. This is a direct result from the fact that DOC builds $\leq_a$ by observing the values that needs to be fused. We can see that for dataset 'restaurant', the number of unresolved fusion operations that are due to some value(s) that are missing in $\leq_{ref}$, is relatively high. This is caused by the

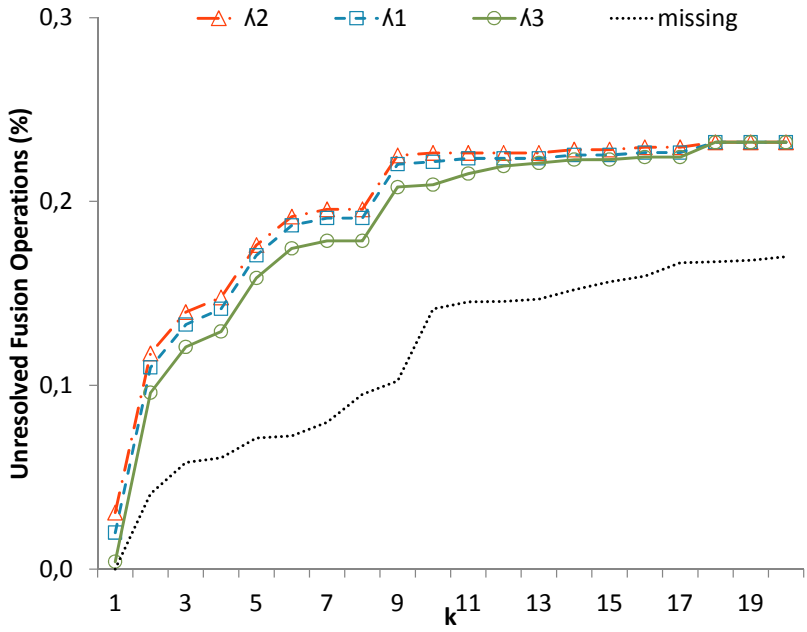
Table 5.2 Comparison of % unresolved fusion operations. For $\leq_{ref}$ and $\mathrm{F}_{\mathrm{B}}^{\leq_a}$, % of cases with at least one missing value are shown between brackets.

Dataset	$\leq_{ref}$	majority	$\mathrm{F}_{\mathrm{B}}^{\leq_a}$ (λ_2, $k=1$)
R-NL	0.253 (0.007)	0.952	0.047 (0)
R-EN	0.070 (0.004)	0.997	0.024 (0)
R-FR	0.138 (0.009)	0.996	0.037 (0)
R-DE	0.057 (0.003)	0.998	0.020 (0)
R-ES	0.049 (0.003)	0.998	0.021 (0)
R-IT	0.048 (0.003)	0.998	0.020 (0)
Rest	0.380 (0.183)	0.986	0.042 (0)

fact that the dataset contains restaurants from two guides (Zagat and Fodor), while the ground truth was downloaded from the Zagat website. Let us now study the differences between (a) different selection strategies and (b) different measures of generality. Figures 5.5 and 5.6 show the mean percentage of unresolved fusion operations over all datasets for respectively minimal and balanced selection. Results are shown for different λ and in function of k. The black dotted line indicates the percentage of fusion operations in which at least one value does not occur in $\leq_a$. Inspection of the results in Figures 5.5 and 5.6 yields the following observations. First, there appears to be a difference between different λ, which is an aspect that is further investigated statistically below. Second, the difference between minimal and balanced selection appears to be extremely small, which can be explained by the fact that a majority of the cases involve only two alternatives, rendering the balanced and minimal selectional strategy virtually equivalent. The impact of the selection strategy is also further investigated statistically in the following. Third, as k increases, the mean percentage of unresolved fusion operations increases strongly. This is caused partially by the fact that a higher k implies filtering out values from the generative multirelation (Section 4.3) as is shown by the increasing mean percentage of cases in which values are missing in $\leq_a$ (black dotted line). However, the percentage of cases with missing values only explains a part of the unresolved fusion operations. Increasing k also affects the ability to compare values that need to be fused, despite the fact that these values are all present in $\leq_a$. For these cases, proper tie breaking can resolve the fusion operation. This is investigated further below.

In order to gain a more in-depth insight in the difference between the different measures of generality, the same statistical analysis as performed in Section 4.4.2 is applied. That is, a Friedman test [105] and a Kendall's W test [106] are performed on weighted percentages of unresolved fusion operations, to verify whether there is a significant difference between these measurements for different λ. The null hypothesis is here that there is indeed a difference between different λ. Table 5.3 shows the results of the tests in the case of minimal selection. It can be

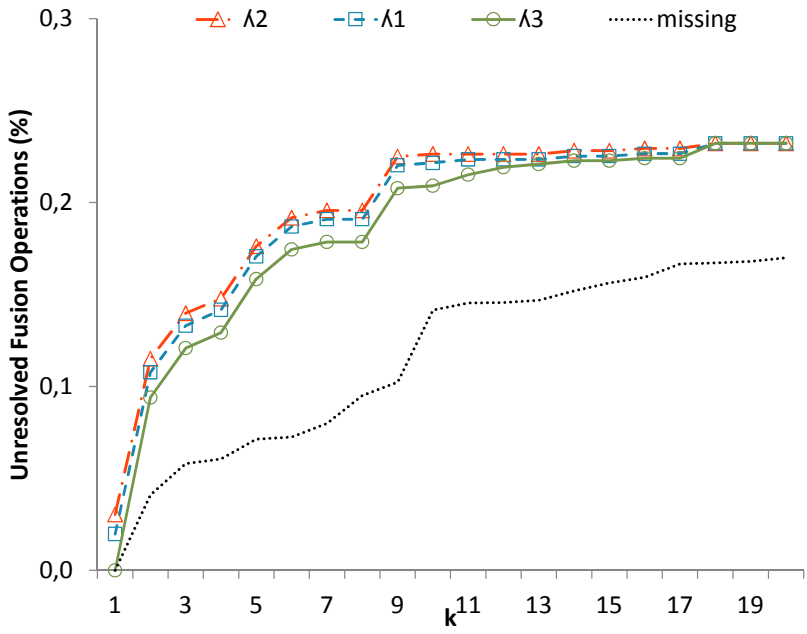
Figure 5.5 Mean percentage of unresolved fusion operations over all datasets for $F^{\leq_a}_{\min}$ for different λ in function of k.



learned from these tests that there is a significant difference between the tested measures of generality for $k \leq 17$. As from $k > 17$, there is no difference between the different measures. Interestingly, it is measure λ_3 that provides the lowest percentage of unresolved fusion operations, followed by λ_1 and λ_2 in that order. As done before, a Wilcoxon test for comparison of two related samples [107] is used. As a result, it is found that the percentage of unresolved fusion operations is significantly smaller for λ_3 for $k \leq 17$. If we pair these results with the results from Section 4.4.2, it can be concluded that λ_3 yields *more* assertions (there are less cases in which the fusion is not resolved), but tends to yield a lower ratio of *correct* assertions. The same statistical analysis is performed in the case of balanced selection. Those results are omitted here because they completely confirm the results that are found in the case of minimal selection.

At this point, it is found that there indeed are differences between the measures of generality. However, because we are evaluating $\leq_a$ in terms of its usability to support a fusion function, it is also interesting to investigate to what extent the selection strategy has an influence on the percentage of unresolved fusion operations. Because there are only two selection strategies, the Friedman test and Kendall's W test are skipped and we immediately perform a Wilcoxon test for two related samples. The null hypothesis is that there is a significant difference between se-

Figure 5.6 Mean percentage of unresolved fusion operations over all datasets for $F_B^{\leq_a}$ for different λ in function of k.



lection strategies. The Wilcoxon test shows that there is a significant difference between the selection strategies for all tested λ if $k \leq 2$. Under those conditions, the balanced selection results in *less* unresolved fusion operations than minimal selection. However, for $k \geq 3$, no significant difference is measured between both selection strategies. This rather small difference between selection strategies can be explained by noting that almost all fusion operations are characterized by $|\Delta[\![a]\!]| = 2$. In that case, the minimal and balanced selection strategy behave identically with respect to unresolved fusion operations.

A statistical comparison is done between $\leq_{ref}$ (reference taxonomy) and $\leq_a$ (DOC). For each value of k, a Wilcoxon test is used to compare the percentages of unresolved fusion operations obtained by using $\leq_{ref}$ with those obtained for $\leq_a$ under fixed selection and λ. Table 5.4 shows a summary of the results for those tests. Hereby, a table entry containing "$\leq_a$" indicates a significant lower percentage of unresolved fusion operations for the DOC method, while a table entry containing "$\leq_{ref}$" indicates a significant lower percentage of unresolved fusion operations for the reference taxonomy.

As can be seen from these results, no statistical difference is measured between both selection strategies. For λ_1 and λ_2, the DOC method results in significantly less unresolved fusion operations if $k \leq 3$. For λ_3, this is the case if $k \leq 11$. In

Table 5.3 Friedman and Kendall's W results on % unresolved fusion operations for minimal selection.

k	MR λ_1	MR λ_2	MR λ_3	Sig.	W
1	2	3	1	0.000	0.998
2	2	2.99	1.01	0.000	0.992
3	2.22	2.68	1.09	0.000	0.842
4	2.22	2.68	1.09	0.000	0.842
5	2.22	2.68	1.09	0.000	0.842
6	2.32	2.58	1.09	0.000	0.850
7	2.32	2.58	1.09	0.000	0.850
8	2.32	2.58	1.09	0.000	0.850
9	2.32	2.58	1.09	0.000	0.850
10	2.32	2.58	1.09	0.000	0.850
11	2.17	2.43	1.41	0.000	0.555
12	2.02	2.27	1.71	0.000	0.280
13	2.02	2.27	1.71	0.000	0.280
14	2.02	2.27	1.71	0.000	0.280
15	2.02	2.27	1.71	0.000	0.280
16	2.02	2.27	1.71	0.000	0.280
17	2.02	2.27	1.71	0.000	0.280
18	2	2	2	-	-
19	2	2	2	-	-
20	2	2	2	-	-

other cases, using a reference taxonomy yields less unresolved fusion operations. These results provide more evidence for the statement that was made at the end of Section 4.4.2, where it was claimed that using DOC might be preferred over using a fixed taxonomy. Indeed, it is seen that, for low values of k, the DOC method provides an order relation $\leq_a$ that yields significantly less unresolved fusion operations than with $\leq_{ref}$. In addition, in Section 4.4.2, it is shown that these additional assertions are in most cases not falsified by the ground truth. It is seen also that if the measure of generality is λ_3, the effect is also true for larger values of k.

Finally, the occurrence of ties is further investigated. First, it should be noted that, because DOC creates an order relation specifically based on the clusters that need to be fused, there is a very low percentage of ties that needs to be broken if $k = 1$ (Figures 5.5 and 5.6). This is again a motivation to choose k sufficiently low. However, even for $k = 2$, there is already a significant increase in the number of ties. Therefore, the breaking of ties is investigated deeper here. Because we have observed that only small differences exist between balanced and minimal selection, we restrict ourselves to minimal selection in this discussion. For all the mentioned datasets, we have evaluated to what extent ties can be broken. There-

Table 5.4 Summary of Wilcoxon's comparisons between % of unresolved fusion operations for $\leq_a$ and $\leq_{ref}$.

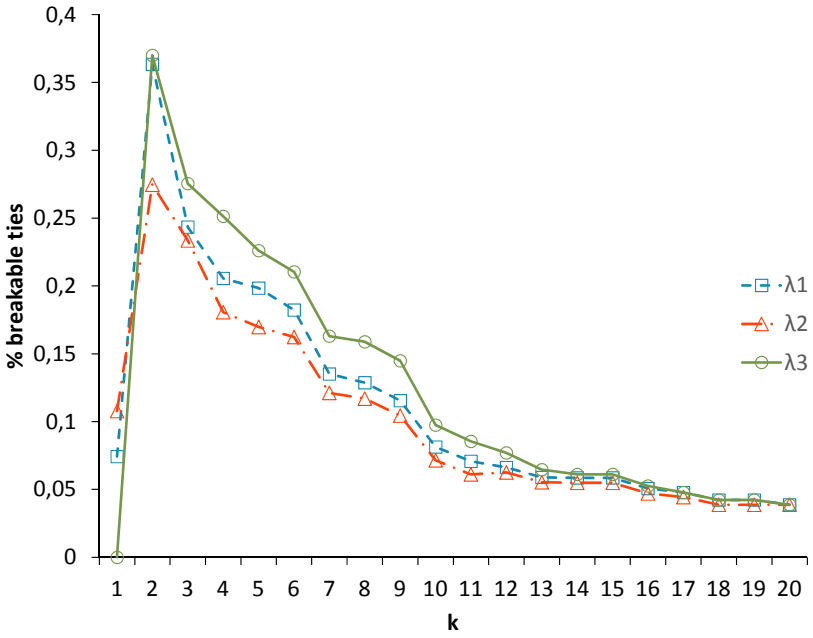
	Minimal			**Balanced**		
k	λ_1	λ_2	λ_3	λ_1	λ_2	λ_3
1	$\leq_a$	$\leq_a$	$\leq_a$	$\leq_a$	$\leq_a$	$\leq_a$
2	$\leq_a$	$\leq_a$	$\leq_a$	$\leq_a$	$\leq_a$	$\leq_a$
3	$\leq_a$	$\leq_a$	$\leq_a$	$\leq_a$	$\leq_a$	$\leq_a$
4	$\leq_{ref}$	$\leq_{ref}$	$\leq_a$	$\leq_{ref}$	$\leq_{ref}$	$\leq_a$
5	$\leq_{ref}$	$\leq_{ref}$	$\leq_a$	$\leq_{ref}$	$\leq_{ref}$	$\leq_a$
6	$\leq_{ref}$	$\leq_{ref}$	$\leq_a$	$\leq_{ref}$	$\leq_{ref}$	$\leq_a$
7	$\leq_{ref}$	$\leq_{ref}$	$\leq_a$	$\leq_{ref}$	$\leq_{ref}$	$\leq_a$
8	$\leq_{ref}$	$\leq_{ref}$	$\leq_a$	$\leq_{ref}$	$\leq_{ref}$	$\leq_a$
9	$\leq_{ref}$	$\leq_{ref}$	$\leq_a$	$\leq_{ref}$	$\leq_{ref}$	$\leq_a$
10	$\leq_{ref}$	$\leq_{ref}$	$\leq_a$	$\leq_{ref}$	$\leq_{ref}$	$\leq_a$
11	$\leq_{ref}$	$\leq_{ref}$	$\leq_a$	$\leq_{ref}$	$\leq_{ref}$	$\leq_a$
12	$\leq_{ref}$	$\leq_{ref}$	$\leq_{ref}$	$\leq_{ref}$	$\leq_{ref}$	$\leq_{ref}$
13	$\leq_{ref}$	$\leq_{ref}$	$\leq_{ref}$	$\leq_{ref}$	$\leq_{ref}$	$\leq_{ref}$
14	$\leq_{ref}$	$\leq_{ref}$	$\leq_{ref}$	$\leq_{ref}$	$\leq_{ref}$	$\leq_{ref}$
15	$\leq_{ref}$	$\leq_{ref}$	$\leq_{ref}$	$\leq_{ref}$	$\leq_{ref}$	$\leq_{ref}$
16	$\leq_{ref}$	$\leq_{ref}$	$\leq_{ref}$	$\leq_{ref}$	$\leq_{ref}$	$\leq_{ref}$
17	$\leq_{ref}$	$\leq_{ref}$	$\leq_{ref}$	$\leq_{ref}$	$\leq_{ref}$	$\leq_{ref}$
18	$\leq_{ref}$	$\leq_{ref}$	$\leq_{ref}$	$\leq_{ref}$	$\leq_{ref}$	$\leq_{ref}$
19	$\leq_{ref}$	$\leq_{ref}$	$\leq_{ref}$	$\leq_{ref}$	$\leq_{ref}$	$\leq_{ref}$
20	$\leq_{ref}$	$\leq_{ref}$	$\leq_{ref}$	$\leq_{ref}$	$\leq_{ref}$	$\leq_{ref}$

fore, for each tie that was not caused by a missing value (i.e. all values that cause the tie are present in $\leq_a$), it was verified whether or not it could be broken. The tie breaking mechanism used was structural inference with the heuristic of counting the number of values in $\leq_a$ that are more specific than the tied values. Majority voting was found to be useless due to the fact that most clusters contain only two values and source preference was not an option as sources were not mentioned in the RouteYou datasets.

Figure 5.7 shows the mean percentage of breakable ties over all datasets for minimal selection and different λ in function of k. It can be seen that, first, for increasing k a higher percentage of ties can be broken. This is due to the fact that, on the one hand, there is a huge increase in the number of ties (Figure 5.5). On the other hand, $\leq_a$ is still sufficiently large to break many of these ties. However, as for $k \geq 3$, the size of $\leq_a$ quickly drops and the number of ties grows less quickly, leading to a decreasing percentage of ties that can be broken.

Conclusion. The DOC method is evaluated in terms of the number of fusion operations it can not resolve uniquely. In the first place, it is found that the usage

Figure 5.7 Mean percentage of breakable ties over all datasets for minimal selection and different λ in function of k.



of a proper order relation outperforms majority fusion greatly in this respect. Still, as k increases, more and more ties occur and a proper tie breaking mechanism is required. Selectional inference can be used to some extent as such a mechanism. Secondly, there are significant differences between the three λ. It is shown that λ_3 yields the least amount of unresolved cases, followed by λ_1 and finally λ_2. Thirdly, few differences are found between the minimal and balanced selection. Finally, if k is kept low, the order relation provided by DOC method outperforms a fixed taxonomy with respect to the amount of unresolved fusion operations.

5.5.3 Execution time

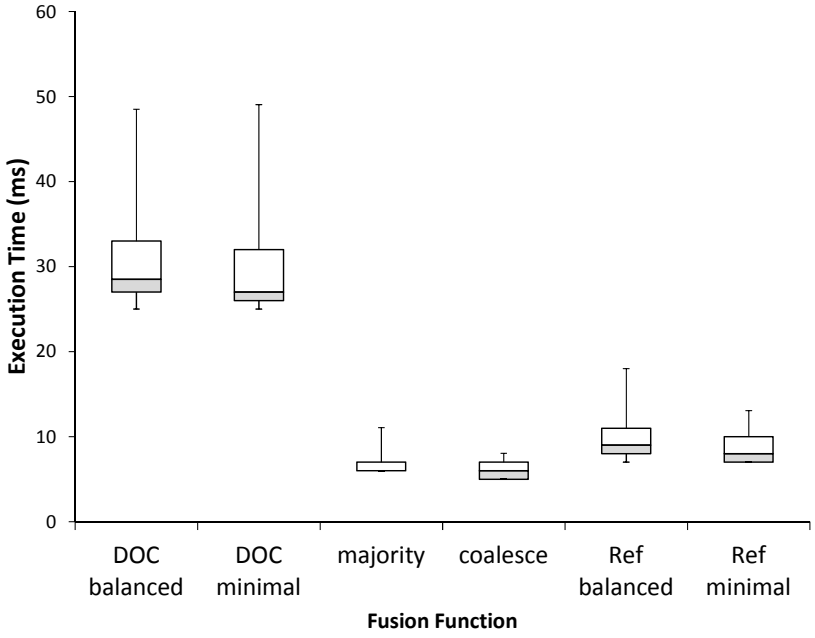
Goal. In the last experiment, it is investigated to what extent the usage of the DOC method introduces a computational overhead with respect to standard fusion functions.

Procedure. The execution time necessary to perform all fusion operations on all seven datasets is measured and summed. Hereby, it is assumed that $k = 1$ and $\lambda = \lambda_2$. In case of the DOC method, the time taken to construct $\leq_a$ is also measured

and added to the total execution time. This procedure is repeated for 100 times, so that mean execution times between different fusion function can be compared with a one-way ANOVA test. The fusion functions taken into account are majority fusion, coalesce fusion (i.e., random non-NULL value), two fusion functions with a fixed reference taxonomy using resp. minimal and balanced selection and two fusion functions based on DOC using resp. minimal and balanced selection.

Evaluation metrics. The mean execution time in milliseconds is used to compare fusion functions.

Figure 5.8 Box plots for execution times (in ms) for six fusion functions.



Results and discussion. It is confirmed by the Levene statistic that the variances on the sequences are homogeneous, which is a required constraint to check before using an ANOVA test. The ANOVA test shows that there is a strong significant difference between the six fusion functions. The differences are further investigated with a Bonferroni post-hoc test [138] and a box plot is shown in Figure 5.8, where the whiskers indicate the 95% confidence interval. The post-hoc test shows that there is a mutual difference in execution time between all tested fusion functions, except between majority fusion and coalesce fusion. The longest mean execution

time is measured for the balanced fusion function in combination with DOC, followed by the fusion function with minimal selection in combination with DOC. The majority fusion function and coalesce fusion function have the lowest execution time. In between, the fusion functions with reference taxonomy can be found. Two important conclusions are drawn from these results. First, a fusion function adopting the DOC method is an order of magnitude slower than trivial fusion functions. The gain in accuracy thus comes with the cost of a decreasing execution time. Second, surprisingly, an albeit small but significant difference was found between the two tested selection strategies for the fusion function adopting DOC. Indeed, the difference between fusion functions with reference taxonomy and with DOC-based order relations shows that the dynamical construction of the order relation by the DOC method contributes the most computational cost.

Conclusion. It is found that fusion functions adopting the DOC method for construction of an order relation, are significantly slower than simple baseline fusion functions. It is found that the balanced selection strategy implies a small additional computational cost, but the largest computational cost is due to the DOC method.

5.5.4 Concluding remarks

It has been shown that parameter k can be used to steer the quality of the generated order relation in the context of data fusion. A higher k implies an order relation that yields more unresolved fusion operations upon deploying it in a selectional fusion function. It is found, in the context of data fusion, that measure of generality λ_3 provides more assertions, which translates into less unresolved fusion operations. When deploying the order relation generated by the DOC algorithm in a fusion function, few differences are measured between the minimal and the balanced selection strategies (only for very low k). The usage of the balanced selection comes with a small additional computational cost compared to minimal selection. It is found that the usage of the DOC method implies a significant higher execution time compared to baseline functions such as majority fusion and coalesce fusion, as well as fusion functions that use a fixed taxonomy.

5.6 Conclusions

In this chapter, selectional fusion functions have been studied in the special case where a proper order relation is either unknown or difficult to obtain. The Dynamical Order Construction (DOC) algorithm has been used to construct such an order relation in an automated fashion, upon observing the data that need to be fused. In addition, a novel selection strategy called balanced selection has been introduced.

Attention is hereby given to the problem of tie breaking. The behaviour of the selection strategy has been evaluated widely on several real life datasets, gaining us with valuable insight in the influence of the different parameters of our approach. Moreover, it has been shown how the proposed approach has benefits compared to a fixed taxonomy.

Thus, this chapter constitute an answer to the third research question which concerns the impact of the dynamically constructed knowledge base on the data fusion in homogeneous and heterogeneous data collections. However, in case of heterogeneous data collections, an additional transformation (mapping) of values in one data source into values in another data source can be desired to improve the data fusion process. Therefore, novel techniques to establish semantical mappings between values, which can be sorted by means of an order relation that reflects a notion of generality, are proposed in the next chapter. However, the established mappings are unambiguous, i.e. a value in the one source can be mapped to more than one value in other source, thus the novel selection techniques are also presented for this mapping. It should be clear, that the selection techniques proposed in this chapter differ from the selection methods presented in the next chapter as follows. For the selection strategies used in data fusion presented in this chapter the desired value is the most specific value that is comparable to all of the other values. This is in contrast to the work presented in the next chapter, where the desired mapping is a specific mapping for a particular object if such a mapping exist or an equivalent mapping otherwise.

6

Semantical mapping of attribute values in data fusion

The following publications have been based on the contents of this chapter:

- Szymczak, M., Bronselaer, A., Zadrożny, S., & De Tré, G. "Selection of Semantical Mapping of Attribute Values for Data Integration," IEEE Intelligent Systems'2014,, Advances in Intelligent Systems and Computing Volume 322, 2015, pp 581-592

- Szymczak, M., Bronselaer, A., Zadrożny, S. & De Tré, G. "Semantical Mapping of Attribute Values for Data Integration," IEEE 2014 Conference on Norbert Wiener in the 21st Century, NAFIPS Annual Meeting, Proceedings. Boston, USA 2014.

6.1 Introduction

In this chapter, we present a novel approach for a specific part of the object mapping problem. Namely, we study automatic value mapping methods for attributes whose values may need semantical comparison and can be sorted by means of an order relation that reflects a notion of specialization-generalization hierarchy. We also propose a novel algorithm that finds a specific mapping for an attribute value of a particular object based on the onomastic information. An experimental evaluation of our method shows the benefits of using semantical mappers and partial

Table 6.1 Examples of tuples extracted from dataset S.

Id	Name	Lon.	Lat.	Category
1	Selfstorage-Achel	5.47067	51.27678	storage
2	Campirama NV	3.25189	50.85282	campground
3	Cafe-Restaurant De Ster	4.05087	51.28177	cafe
4	Het Kouterhof	3.66512	51.03433	lodging
5	Borluut Bed Breakfast	3.65799	51.01888	lodging
6	Carlton Hotel	3.71395	51.03628	lodging
7	Snooz Inn	3.73304	51.05880	lodging

order relations as compared to a lexical mapper, and helps to understand the role of the parameters.

6.1.1 Problem illustration

Let us consider two datasets as a running example in this chapter. They contain tuples which provide geographical annotations for a map and pinpoint locations of specific interest which are called points of interest (POIs). As a reminder, each tuple is characterized by at least four attributes: "name", "longitude", "latitude" and "category". The attribute "name" identifies a specific POI, the "longitude" and "latitude" give the geographic coordinates of the place, and the "category" specifies the type or function of the location. The first dataset is shown in Table 6.1. Its tuples are extracted from a Google database[1], called the source S, with a known partial order relation $\leq_{a_c}^{S}$ on the domain of the "category" attribute. Figure 6.1 presents a part of this relation. The most general concept is represented in the root of the tree and its descendant nodes correspond to more specific concepts. For instance, the concept "establishment" in Figure 6.1 is the most general concept among the values of the attribute "category" of dataset S and has children corresponding to more specific structures (e.g., "lodging", etc.). The second dataset contains tuples extracted from the RouteYou dataset[2], called the target T, also with a known partial order relation $\leq_{a_c}^{T}$ on the domain of the "category" attribute. Table 6.2 shows a few tuples extracted from the target dataset, while a part of the order relation $\leq_{a_c}^{T}$ is presented in Figure 6.2. For instance, the concept "POI" in Figure 6.2 is the most general concept among the values of attribute "category" of dataset T and has children corresponding to more specific concepts (e.g., "Support", "Accommodation", "Shopping location", etc.).

[1] Google, http://maps.google.com
[2] RouteYou, http://www.routeyou.com/

Table 6.2 Example of tuples extracted from dataset T.

Id	Name	Lon.	Lat.	Category
1	Pakhuis Stokholm	4.66589	51.81835	Warehouse
2	Camping De Iembarg	7.11147	52.96790	Camp Site
3	Cafe Theatre	3.72201	51.04983	Restaurant
4	Het Kouterhof	3.66514	51.03437	Hotel
5	Borluut Bed Breakfast	3.65797	51.01893	Guest room
6	Hotel City Inn	9.36967	52.32978	Hotel
7	Santellone Resort	10.1663	45.55010	Hotel

Let us consider a data integration scenario in which tuples from a dataset S have to be merged with tuples from a dataset T. Like we mentioned in the Introduction, values of attributes such as "name", "longitude" and "latitude" might be merged without any additional processing. However, importing values of attributes such as "category", representing information on the type of point of interest, is less trivial as they may often refer to the same concepts presented in a different way in both datasets. For instance, the concept "accommodation" is represented by the category "lodging" in the dataset S and by the categories "Accommodation", "Hotel", "Guest Room" etc. in the dataset T. Therefore, for successful data integration, it may be crucial to create mappings of values of the category attributes in the datasets S and T. Following intuition, the desired mapping is a mapping between concepts at the same level of abstraction, e.g., for the above example that mapping should be established between "lodging" and "Accommodation". However, for a particular tuple in the source dataset, e.g. "Carlton Hotel" with the category "lodging" in Table 6.1, there may exist a mapping which better expresses the category of this particular tuple, e.g. "Hotel". Thus, for successful data integration, it is crucial not only to establish mappings but also to select proper mappings of values of the category attributes in the datasets S and T. Such mappings help to maintain consistency and decrease the number of duplicates in the integrated dataset, which has an extreme influence on data quality. This in turn decreases the cost of database maintenance.

In our approach, on the one hand, *explicit* mappings are created by using predefined mappers that are based on category *descriptions* which are compared using information retrieval techniques. The category description is a textual description of each category and is generated from the values of other attributes (such as the "name" attribute) or extracted from an external source (e.g., World Wide Web). Moreover, the certainty of each mapping is expressed by a possibilistic truth value (PTV) and hence is based on fuzzy set and possibility theories [31, 35]. On the

Figure 6.1 A part of the partial order relation $\leq_{a_c}^{S}$ for the category attribute from the dataset S.

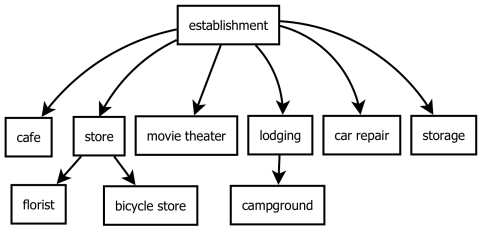


Figure 6.2 A part of the partial order relation $\leq_{a_c}^{T}$ for the category attribute from the dataset T.

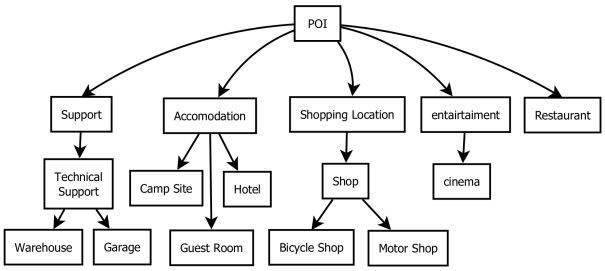
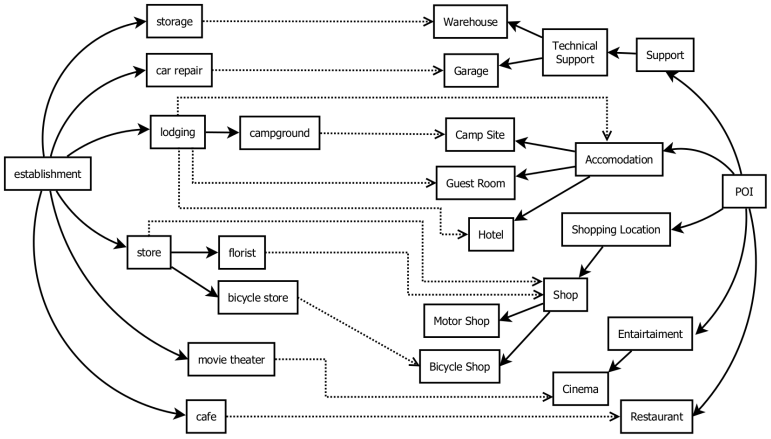


Figure 6.3 Example of mappings between categories of dataset A and dataset B.



other hand, the partial order relations and the explicit mappings are used to infer *implicit* mappings.

As a consequence, one-to-many mappings are established, which means that one category from the source dataset is mapped to one or more categories from the

target dataset. The examples of mappings between categories forming hierarchies shown in Figure 6.1 and Figure 6.2 are presented in Figure 6.3. The dotted arrows indicate the mappings of categories from the source dataset to categories of the target dataset, e.g., the mapping of "lodging" to "Hotel". We make the following three assumptions. First of all, the schema of each considered dataset contains such an attribute that there exists a partial order relation defined on its domain. Next, such an order relation is known in advance. Finally, we assume that the schema matching between input datasets is established. Many problems have to be addressed while devising such a mapping algorithm. The most important among them are the following:

- How to create a mapping between categories from heterogeneous sources?
- How can a partial order relation be used to create a mapping between categories?
- Should the mapping be biased towards more specific or more general information?
- How to select the proper mapping for each category?
- How to use the proper mapping for each tuple?

6.1.2 Contributions

With respect to the problems given above, the following research objectives are dealt with in this chapter. Namely, the novel automatic mapping and selection algorithms are proposed for attribute domains which are endowed with partial order relations that reflect a notion of specialization-generalization hierarchy. The mapping algorithm creates one-to-many candidate mappings between semantically related values. To this aim, an extensible set of mappers are investigated that are based on the constructed textual descriptions of considered values and employs information retrieval techniques for further processing. The selection algorithm reduces the mapping set to a set of one-to-one mappings, if possible, on the same level of abstraction. It is based on the concept of majority (the frequency of mappings) and the balanced selection (the most specific category from the candidate categories provided that the selected category is comparable to all other candidate categories) over the candidate mappings set. In addition, we also propose an algorithm to find a mapping specific for a particular object based on the onomastic information. In each of these algorithms, the given specialization-generalization hierarchy plays an important role which is also studied in this chapter. The established semantical mappings help to maintain consistency of integrated data from heterogeneous sources.

6.1.3 Outline

The remainder of this chapter is organized as follows. In Section 6.2, an overview of work related to the topic of this chapter is provided. Next, in Section 6.3 a framework of value mapping is introduced and the algorithm is briefly described. Section 6.4, Section 6.5 and Section 6.6 contain the detailed description of applied mapping functions (mappers) and mapping selection heuristics. Next, Section 6.7 presents the results of computational experiments. Finally, we conclude in Section 6.8.

6.2 Related work

The problem of object mapping has been studied in different contexts such as record linkage [3], duplicate detection [22, 139, 140], data integration [13] and knowledge base construction [25]. Most of the previous works assume that values of corresponding attributes are drawn from the same domain or at least that they bear some textual similarity that can be measured using a kind of the distance (e.g. edit distance). Some approaches are based on statistical information processing [24, 26, 28]. For instance, Kang et al. in [24] exploit a statistical model which captures the co-occurrence of values of all attributes characterizing a dataset. Next, constructed models are aligned assuming various matchings between the values of a given attribute in both datasets. The alignment with the minimum distance between the aligned models is returned as the mapping. In [28] a strategy is presented that uses statistical techniques to detect overlapping subsets of data present in disparate sources, through which rules for data conversion may be extracted. In [27] domain independent string transformations are proposed to syntactically compare shared tuple attributes. The established mappings depend on the mapping rules, which are determined by a mapping learner and supervised by the user. In contrast, in [23], mappings are based on non-overlapping correlated attributes using a combination of profilers that contain the specific knowledge about what constitutes a typical concept.

There is also a lot of work about semantic relatedness measures which differ in the source of background knowledge. Thesaurus-based approaches are limited in the vocabulary for which they can provide relatedness measures and they are constructed manually, e.g., WordNet [57] and Roget [141], while corpus-based techniques are based on the statistical analysis of large untagged document collections; e.g., LSA [59] relies on the tendency for related words to appear in similar contexts. The last group contains approaches using Wikipedia. WikiRelate [142] applies techniques that are used by WordNet. Namely, path-length measure is employed, which takes into account the depth at which considered concepts are found [143], what gives results similar in terms of accuracy to thesaurus-based

approaches [58]. Instead of comparing vectors of term weights to evaluate the similarity between queries and documents, ESA [144] compares weighted vectors of the Wikipedia articles related to each term. On the other hand, WLM [58] is based on Wikipedia's hyperlink structure to define relatedness and is theoretically cheaper and more accurate than ESA because textual content can be largely ignored and is more closely tied to the manually defined semantics of the resource.

Moreover, there have been many studies on POIs. For instance, Choi et al. in [145] investigated the problem of POI categorization based on the onomastic and local contextual information using a training set. Namely, each POI is described by a text, which is processed to infer the category of POI in contrast to our method, where we propose mappings for categories of POIs from different databases.

An object mapping problem can also be investigated in an ontology alignment (OA) context. An ontology alignment system finds correspondence between concepts in two ontologies being aligned [146]. Various matchers in OA systems produce similarity measures between concepts and rely on internal information (such as labels, synonyms, instances and relationships) or external knowledge such as a reference ontology. Systems like OLA [147], Imapper [148], SAMBO [149], SAMBOdtf [150], ASMOV [151, 152], UFOme [153] or the method that is presented in [154] are based on a reference ontology(ies) as knowledge source(s) to map concepts from aligned ontologies via concepts from that reference ontology. These approaches extend string-based techniques by looking up synonyms for considered concepts in lexicons, such as WordNet [57], to improve the results of the OA process. The mediating matcher with semantic similarity MMSS [155] differs from current OA systems using a reference ontology in that it permits a wide variety of semantic measures to be used within different reference ontologies to find additional mappings between the considered ontologies.

In contrast to the above work, our method presented in this chapter suggests semantical mappings for attributes values based on the data itself, extracted descriptions from the World Wide Web and a given partial order relation on the domain of considered attributes.

6.3 Mapping of categories

Before we continue to describe our method for mapping the values of category attributes, first of all we should define the problem and consider different types of mappings.

6.3.1 Problem definition

As a reminder (see Section 1.4), it is assumed that entities from the real world are described as *tuples* which are characterized by a number of *attributes* $a \in \mathcal{A}$.

A *schema* $\mathcal{R}$ of a given dataset is identified by a non-empty and finite subset of $\mathcal{A}$. For each attribute $a \in \mathcal{A}$, let $dom(a)$ denote the domain of a (the set of possible values for attribute a) and $dom'(a)$ denote the subset of the domain of a comprising values of a actually present in the dataset.

Two datasets are considered. The source dataset over the schema $\mathcal{R}_S$ with a set of attributes $A_S = \{a_1^S, \ldots, a_n^S\}$ is denoted as S, while the target dataset over the schema $\mathcal{R}_T$ with a set of attributes $A_T = \{a_1^T, \ldots, a_m^T\}$ is denoted as T. We assume that the schema matching is known:

$$M : A_S \rightarrow A_T \tag{6.1}$$

Moreover, at least one *category* attribute $a_C^S \in A_S$ is distinguished with values $c^S \in dom(a_C^S)$ (called categories for short), and similarly its corresponding (mapped) attribute is denoted by $a_C^T \in A_T$ with values $c^T \in dom(a_C^T)$. The category attribute is an ordinal attribute [131] of which the values (categories) are (partially) ordered by means of a generalization/specialization relation $\leq_{a_c}$. In this context we assume that in each partial order relation $\leq_{a_c}$ the most general category called *root* exists for which all other categories are specializations (in contrast to an order relation $\leq_a$ which is constructed by the DOC algorithm in Chapter 4 for which this constraint does not hold). This assumption is commonly satisfied in real-world datasets. Morever, $\leq_{a_c}$ satisfies all other properties of $\leq_a$, namely, it is reflexive, antisymmetric and transitive (see Chapter 4). In other words, the set of categories, namely $dom(a_C^S)$ or $dom(a_C^T)$, is an upper semilattice, because it is a partially ordered set that has a least upper bound (supremum) for any nonempty finite subset, i.e., for any set of values of the category attribute a supremum exist.

The one-to-many categories mapping is defined as follows.

Definition 20 (One-To-Many Category Mapping, $\bowtie_{1:n}$-mapping)**.**
A pair $(c^S, \{c_1^T, \ldots, c_p^T\})$ is called a one-to-many category mapping and it maps a category $c^S \in dom'(a_C^S)$ to a nonempty subset of categories $\{c_1^T, ..., c_p^T\} \subseteq dom(a_C^T)$, called the candidate categories set, representing the coreferent categories of $c^S \in dom'(a_C^S)$ in $dom(a_C^T)$: where $dom'(a_C^S)$ is a subset of the domain of the attribute a_C^S comprising values of a_C^S actually present in the dataset S. A set of $\bowtie_{1:n}$-mappings will be denoted as $\gamma_{1:n}$.

A $\bowtie_{1:n}$-mapping should be seen as an intermediate step towards determining a one-to-one mapping which is defined below.

Definition 21 (One-To-One Category Mapping: $\bowtie_{1:1}$-mapping)**.**
A pair (c^S, c^T) is referred to as a one-to-one category mapping and maps a category $c^S \in dom'(a_C^S)$ to exactly one category $c^T \in dom(a_C^T)$ representing its coreferent category. A set of $\bowtie_{1:1}$-mappings will be denoted as $\gamma_{1:1}$.

Thus, our ultimate goal is determining a $\bowtie_{1:1}$-mappings for values of a category attribute a_C^S and establishing a $\bowtie_{1:n}$-mapping is only the first step. In fact, a $\bowtie_{1:n}$-mapping $(c^S, \{c_1^T, \ldots, c_p^T\})$ should be interpreted as a set of $\bowtie_{1:1}$-mappings $\{(c^S, c_1^T), \ldots, (c^S, c_p^T)\})$ sharing the same first element, c^S.

In our approach we also use PTVs to express the (uncertainty about the) results of a comparison (cf., Section 1.4.4). As a reminder, hereby, a PTV is a (normalized) possibility distribution [31] defined over the set of Boolean values $\mathbb{B}$ [35]. In the context considered here, a PTV denoted as $\tilde{\bowtie}_{1:1}^i$ expresses the uncertainty about the validity of a $\bowtie_{1:1}$-mapping (c_i^S, c_i^T).

6.3.2 Equivalent and non-equivalent mappings

Let us consider our exemplary datasets S and T shown in Fig. 6.1 and Fig. 6.2, respectively. The mappings shown in Fig. 6.3 may be intuitively conceived. The following subgroup may be distinguished among these mappings: (“movie theater”, “Cinema”), (“campground”, “Camp Site”), (“car repair”, “Garage”). These mappings are valid in both directions, i.e., from the dataset S to the dataset T and inversely, because these categories represent coreferent information on the same level of abstraction. These mappings are called *equivalent* mappings.

In contrast, mappings such as the one between the concepts “lodging” and “Hotel” are asymmetric, in a sense. On the one hand, not every “lodging” is a “Hotel”. Therefore “lodging” should be mapped to a more general concept than “Hotel”. On the other hand, each “Hotel” is a “lodging”. These categories describe different levels of abstraction; they are not equivalent, i.e. “lodging” is a more general concept than “Hotel” and “Hotel” is a specialization of “lodging”. Therefore, these mappings are called *non-equivalent* mappings, which are further divided into two subclasses. The first one, called *generalized* mappings, contains mappings in which the target category is a generalization of the source category and is a valid mapping but on a different level of abstraction. In contrast to that, a mapping of which the target category is a specialization of the source category is called a *specialized* mapping and it may be an invalid mapping, e.g., not each “lodging” is “Hotel”; however, a strong semantical relation still exists between those categories.

Due to the conditions described above, the direction of mapping has to be considered during the data integration. In this chapter we consider directional mappings from the source dataset S to the target dataset T.

6.3.3 Algorithm

Our novel method consists of three main phases: preprocessing, category mapping and category selection. First algorithm, Algorithm 6.1 presented in Section 6.3.3.1, detects coreferent categories and establishes, in general, one-to-many mappings

Algorithm 6.1 CATEGORYMAPPINGALGORITHM

Require: Datasets S and T, Partial Order Relations $\leq_{a_c}^S$ and $\leq_{a_c}^T$, Mappers $M = M_E \cup M_I$
Ensure: A set of $\bowtie_{1:n}$-mappings $\gamma_{1:n}$
1: $\gamma_{1:n} \leftarrow \emptyset$
2: **for all** $m \in M_E$ **do**
3: $\gamma_{1:n} \leftarrow \gamma_{1:n} \cup$ m.getMappings($dom'(a_C^S)$,$dom(a_C^T)$,S,T)
4: **end for**
5: **for all** $m \in M_I$ **do**
6: $\gamma_{1:n} \leftarrow \gamma_{1:n} \cup$ m.getMappings($\gamma_{1:n}$,$\leq_{a_c}^S$,$\leq_{a_c}^T$)
7: **end for**

between them. Examples of such mappings are shown using dotted arrows in Figure 6.3. Finally, these mappings are further processed by Algorithm 6.2 presented in Section 6.3.3.2, which yields one-to-one mappings. Such a processing as well as classification of a mapping as being equivalent or non-equivalent may be needed in data integration and will be studied in the following sections.

6.3.3.1 Category mapping

The Category Mapping Algorithm (CMA, Algorithm 6.1) creates mappings for attributes equipped with a partial order relation that reflects a notion of generality. Therefore, the datasets S (source) and T (target), a partial order relation (upper semi-lattice) $\leq_{a_c}^S$ on the domain of the source category attribute $a_C^S \in A_S$ and a partial order relation $\leq_{a_c}^T$ on the domain of the target category attribute $a_C^T \in A_T$ are assumed to be given. Moreover, an extensible set of mapping methods, called mappers, are also given. These mappers are classified as *explicit* mappers M_E and *implicit* mappers M_I and are described in detail in Section 6.4 and 6.5, respectively. In the first step of the Algorithm 6.1 (lines 1-3), explicit mappings are established which are based on instance data of the datasets or external sources. In the second step (lines 4-6), implicit mappings are inferred which are based on the explicit mappings and partial order relations $\leq_{a_c}^S$ and $\leq_{a_c}^T$. These mappings are produced by a method *getMappings* which is different for each mapper. The output of the algorithm is a set $\gamma_{1:n}$ of $\bowtie_{1:n}$-mappings, stating the coreference of categories.

The coreference of a pair of values is assumed to be a binary notion, i.e., two values are coreferent or not. However, one may be uncertain if it does or does not hold for a given pair of values. Thus, all the mappers associate a PTV with each $\bowtie_{1:1}$-mapping they produce (remember that a $\bowtie_{1:n}$-mapping is just a set of $\bowtie_{1:1}$-mappings). This PTV denotes the uncertainty about the coreference. Each mapping with a PTV equal or close to (1,0) is considered as holding with high

confidence. In contrast, a mapping with PTV close to (0,1) or (1,1) means that there is not enough information available to claim the coreference. Therefore, only mappings associated with a PTV below a predefined threshold for the possibility of false are taken into account (see Section 1.4.4).

6.3.3.2 Category selection

Next, the Category Selection Algorithm (CSA, Algorithm 6.2) selects for the category $c^S \in dom'(a_C^S)$ of each tuple $t \in S$ exactly one category $c^T \in dom(a_C^T)$ based on the partial order relation $\leq_{a_c}^T$ defined on the domain of the category attribute a_C^T from the target dataset T. Therefore, the input for the algorithm comprises the source dataset S, the domain of the category attribute a_C^T, the order relation $\leq_{a_c}^T$ and the set $\gamma_{1:n}$ of $\bowtie_{1:n}$-mappings, which are established by Algorithm 6.1. The objective of our algorithm is to select as many as possible true and, if possible, equivalent $\bowtie_{1:1}$-mappings for tuples from the source S.

In the first step (lines 2-6 in Alg. 6.2), for each category c_S from the source the method selectDefaultMapping selects *the best possible* one-to-one mapping (called *default* mapping) from the set $\gamma_{1:n}$ of $\bowtie_{1:n}$-mappings using the heuristics which are described in Section 6.6. *The best possible* mapping means here that the desired mapping is an equivalent mapping if there exists one, or is a non-equivalent but the most specific one otherwise. Determining whether a mapping is equivalent is based on the proposed heuristics. One of these heuristics is based on the intuition that the most certain mapping or the most popular mapping is likely to be equivalent. The others are based on the following idea. For instance, suppose that the category "lodging" is mapped to "Hotel", "Camp Site" and "Guest Room" in Figure 6.3. Without extra information, we do not know whether "lodging" is "Hotel", "Camp Site" and "Guest Room". Therefore, a partial order relation is used to select a concept which generalizes all of these three categories, in this case it is "Accommodation".

In the second step (lines 7-13 in Alg. 6.2), for each object's category from the source the *possible most specific* category is selected, in the sense that the algorithm first tries to find a mapping that is specific for the considered object (lines 8-9 in Alg. 6.2) and is called a *specific* mapping. For instance, let us consider the object "Carlton Hotel" with the category "lodging" in Table 6.1. In the first step the selectDefaultMapping method returns an equivalent default mapping ("lodging", "Accommodation"). However, in the domain of the target category attribute ($dom(a_C^T)$) there exists a specific category ("Hotel") for this object and in fact this category is contained in the name of the considered object. This allows to use the mapping ("lodging","Hotel") for this object. The high precision of this mapping is ensured by selecting only that category which is in the relation with the target category of the default mapping for the considered object's category. For example, for the considered object's category "lodging" the target category of the default map-

Algorithm 6.2 CATEGORYSELECTIONALGORITHM

Require: Dataset S, $dom(a_C^T)$, Partial Order Relation $\leq_{a_c}^T$, $\bowtie_{1:n}$-mappings $\gamma_{1:n}$
Ensure: $\bowtie_{1:1}$-mappings $\gamma_{1:1}$, Dataset S^*

```
1:  γ_{1:1} ← ∅
2:  for all c^S ∈ dom'(a_C^S) do
3:      ⋈_{1:n} ← getMapping(c^S, γ_{1:n})
4:      ⋈_{1:1} ← selectDefaultMapping(⋈_{1:n}, ≤_{a_c}^T)
5:      γ_{1:1} ← γ_{1:1} ∪ ⋈_{1:1}
6:  end for
7:  for all o ∈ S do
8:      if existObjectSpecificMapping(o.name, dom(a_C^T)) then
9:          o.c^S ← getObjectSpecificCategory(o, dom(a_C^T))
10:     else
11:         o.c^S ← getDefaultCategory(o.c^S, γ_{1:1})
12:     end if
13: end for
```

ping is "Accommodation" which is in the relation $\leq_{a_c}^T$ with the detected category "Hotel". Moreover, if more than one category mapping exists for the particular object then the selection mapping methods from the first step are applied.

If a specific mapping for the category of the particular object from the source does not exist then the considered category is mapped according to the established default mapping in the first step (line 11 in Alg. 6.2). Finally, the considered object from the source is mapped to exactly one category from the target and can be later imported to the target dataset.

It should be clear that the proper mapping for any object is not always an equivalent mapping of the object's category (e.g., the default mapping between "lodging" and "Accommodation"). In some cases it is possible to establish a mapping which is more appropriate for the particular object, e.g., the specific mapping between "lodging" and "Hotel".

6.4 Explicit mappers

In this section the novel explicit mappers which establish explicit mappings of the source and target categories using various information are defined. They are used in Algorithm 6.1 presented in Section 6.3. Namely, a *Definition mapper* is based on information extracted from an external source (e.g., World Wide Web). A *Lexical mapper* is based on the lexical similarity of categories. A *Description mapper* and *Identification mappers* are based on data about each category which are extracted from the source or target datasets. These mappers are described in the following subsections.

The considered mappers (except *Lexical mapper*) are based on a textual description of each category value (called *category description*), which is constructed from mapper-dependent information but in a similar way to that which is explained below.

Category description. The generation of the category description is divided into two phases: *terms preprocessing* and *terms importance calculation*.

Terms preprocessing. During the preprocessing phase, a representation in the form of a set of words/terms is obtained for each category value in the following way. The starting point is a collection of relevant strings, e.g., values of selected attributes, such as "name" in Tables 6.1 or 6.2, of all tuples belonging to the same category in the source/target dataset. First, *stop words* are filtered out from these strings. These are words such as "a", "and" or "to" in English. A popular predefined set of stop words contains 527 words [156] and is used also in our computational experiments. Second, special characters appearing in the strings, i.e. dash, semicolon, dot etc., are replaced by white space, which gives a string of terms. Third, the strings are split into terms where the splitter is the white space character. Afterwards, each category is described by a set of terms which is further preprocessed by applying an algorithm for suffix stripping. Terms are stemmed using the Porter stemmer [157]. For instance, terms "connection", "connecting" and "connections" are transformed to their stem which is "connect".

Terms importance calculation. The result of the term preprocessing phase is a clean and unified set of terms for each category value. The preprocessing increases the quality of term's importance calculation in the final phase of generating the category description which has a direct impact on the quality of mappings. Namely, thanks to that, the true mappings have higher certainty. The importance of each term is expressed by the *tfidf* [158, 159] (term frequency and inversed document frequency) weighting scheme. This coefficient reflects that a term that occurs often but not in many other descriptions tends to be more relevant and informative than a term that appears in many descriptions. *Tfidf* combines the frequency of a term t in a description d ($t \in d$) with a factor that discounts its importance with its appearances in the whole description's collection D of the single dataset, and is defined by:

$$\mathit{tfidf}(t,d) = \mathit{tf}(t,d) \times log\frac{|D|}{\mathit{df}(t)} \tag{6.2}$$

where $\mathit{tf}(t,d)$ is the frequency of term t in the description d and is expressed by Equation 6.3, $|D|$ is the size of the whole descriptions collection and $df(t)$ is the number of descriptions in which term t occurs (term t occurs at least in one description).

$$\mathit{tf}(t,d) = \frac{card(t,d)}{\sum_i^{|d|} card(t_i,d)} \tag{6.3}$$

where $card(t, d)$ is the number of occurrences of term t in a description d and is divided by the sum of occurrences of all terms in description d to prevent a bias towards longer documents. Moreover, *tfidf* weights are normalized as follows [158]:

$$\textit{tfidf}^*(t, d) = \frac{\textit{tfidf}(t, d)}{\sqrt{\sum_i^{|d|} \textit{tfidf}(t_i, d)^2}} \tag{6.4}$$

where *tfidf** is the normalized *tfidf*. The final category description is represented as a set $\{(t_i, \textit{tfidf}_i^*)_{i=1,\ldots,n}\}$ that consists of unique terms t_i with a normalized associated weight $\textit{tfidf}_i^*$.

These methods are used in the next subsections where the explicit mappers are defined.

6.4.1 Description mapper

The description mapper creates mappings based on the comparison of category descriptions of the source and the target categories which are generated using values of selected object attributes. This mapper works as follows.

First of all, the category description for each category from the source and the target is constructed (see paragraph Category description in Section 6.4). To this aim, objects from the dataset are grouped into clusters of the same category. For each cluster, values of *selected* attributes are used to generate the category description. The selection of attributes can be done automatically by selecting all textual attributes. For some attributes, such as "name", stemming is not employed because they contain many proper nouns which should be preserved in their original form.

Afterwards, constructed category descriptions of the source and target categories are compared in the pairwise manner. This comparison estimates the possibility that two given categories (their descriptions) are coreferent. The common terms (with weights) of both descriptions are added to the *intersection* that is a subset of common terms in both descriptions while the remaining terms (with weights) are added to the subset called *errors* and have an influence on the computed possibility that categories are (not) coreferent.

This mapper generates a PTV which expresses the uncertainty about the coreference of the compared descriptions as described above. The possibility that a proposition p, stating that two categories are coreferent, is true ($\mu_{\tilde{p}}(T)$) and the possibility that p is false ($\mu_{\tilde{p}}(F)$) are calculated by the following equations:

$$\mu_{\tilde{p}}(T) = \frac{possT}{factor} \tag{6.5}$$

$$\mu_{\tilde{p}}(F) = \frac{possF}{factor} \tag{6.6}$$

where *possT*, *possF* and *factor* are obtained from:

$$possT = \left(\sum_{i=1}^{|intersection|} \text{getTfidf}(\textit{tfidf}_i^{*S}, \textit{tfidf}_i^{*T}) \right)^{pow} \tag{6.7}$$

$$possF = \sum_{i=1}^{|errors|} \textit{tfidf}_i^{*} \tag{6.8}$$

$$factor = \max(possT, possF). \tag{6.9}$$

Hereby, $|intersection|$ denotes the number of common terms, and $|errors|$ is the number of the remaining terms present in the representation of the source and target categories, $\textit{tfidf}_i^{*S}$ ($\textit{tfidf}_i^{*T}$) are the weights of the common terms from the description of the category from the source (the target respectively) and $\textit{tfidf}_i^{*}$ are the weights of the remaining terms from the description of the category from the source or from the target.

On the one hand, $possT$ is the sum of common terms weights (which are calculated by the getTfidf method) and raised to the power pow which makes all mappers comparable (the setting of this parameter is evaluated in Section 6.7.2.1). We propose four different ways to calculate a combined *tfidf* of a source and a target term. They are also evaluated experimentally in Section 6.7.2.2 and are defined as follows:

- the average of $\textit{tfidf}_i^{*S}$ and $\textit{tfidf}_i^{*T}$

$$\text{getTfidf}(\textit{tfidf}_i^{*S}, \textit{tfidf}_i^{*T}) = \frac{\textit{tfidf}_i^{*S} + \textit{tfidf}_i^{*T}}{2} \tag{6.10}$$

- the minimum of $\textit{tfidf}_i^{*S}$ and $\textit{tfidf}_i^{*T}$

$$\text{getTfidf}(\textit{tfidf}_i^{*S}, \textit{tfidf}_i^{*T}) = \min(\textit{tfidf}_i^{*S}, \textit{tfidf}_i^{*T}) \tag{6.11}$$

- the maximum of $\textit{tfidf}_i^{*S}$ and $\textit{tfidf}_i^{*T}$

$$\text{getTfidf}(\textit{tfidf}_i^{*S}, \textit{tfidf}_i^{*T}) = \max(\textit{tfidf}_i^{*S}, \textit{tfidf}_i^{*T}) \tag{6.12}$$

- the source preference which equals $\textit{tfidf}_i^{*S}$ or $\textit{tfidf}_i^{*T}$ depending on the predefined preference assigned to the source or the target dataset

$$\text{getTfidf}(\textit{tfidf}_i^{*S}, \textit{tfidf}_i^{*T}) = \text{pref}(\textit{tfidf}_i^{*S}, \textit{tfidf}_i^{*T}) \tag{6.13}$$

On the other hand, $possF$ is computed as the sum of the weights $\textit{tfidf}_i^{*}$ of the remaining terms (from errors that are found during comparison). Finally, $factor$ is the maximum of $possT$ and $possF$ and is used to normalize both possibilities. As

a reminder, only if the PTV of a mapping of a category description of the source and a category description of the target is such that its possibility of false is below a predefined threshold then the mapping between considered categories is added to $\gamma_{1:n}$ (see Sections 6.3.3.1 and 1.4.4).

For instance, consider categories "lodging" from the source and "Hotel" from the target. Let the source dataset contain objects with category "lodging" as shown in Table 6.1. A part of the description of "lodging" contains the following terms with *tfidf** shown: ("Hotel", 0.85), ("Bed", 0.22), ("Breakfast", 0.22), ("Resort", 0.08), ("Inn", 0.03), ("Carlton", 0.012), ("Borluut", 0.006), etc. On the one hand, the terms "Carlton" and "Borluut" get low weights because they appear infrequently in the tuples of the category "lodging" and, moreover, they are not specific for this category. On the other hand, the terms "Bed" and "Hotel" get higher weights because they are specific terms for accommodation and they appear in many objects of this category. Meanwhile, a part of the description of the target category "Hotel" which is based on the names of objects from the target, shown in Table 6.2, is the following: ("Hotel", 0.96), ("Inn", 0.14), ("Resort", 0.08), ("Bed", 0.01), ("Breakfast", 0.01), ("Carlton", 0.005), etc. Afterwards, these sets are compared and the common terms are detected. Finally, the PTV of this mapping is calculated using Equations (6.5) and (6.6) and equals (1,0.06) which supports high confidence in the coreference of the considered categories.

Moreover, it should be clear that the mappings that are established by this mapper do not depend on the category values themselves but depend on the co-occurring values of other attributes. That feature allows the creation of many valuable mappings, e.g., between categories in different languages.

6.4.2 Definition mapper

In contrast to the previous mapper, this method is based on the extraction of additional knowledge from an external source. More precisely, for each category a textual description is extracted from the Web, in particular, from Wikipedia[3]. The textual description is a webpage (e.g., for the category "lodging" it is a webpage at the URL http://en.wikipedia.org/wiki/lodging) which is processed by such a parser as JSoup HTML Parser[4]. Extracted texts are used to construct the *category description* (see paragraph Category description in Section 6.4). Afterwards, the category descriptions are compared in a pairwise manner and uncertainty as to their matching is quantified as in case of the *description* mapper.

Description extraction. The extraction works as follows. First of all, for each category a predefined number of paragraphs from a relevant Wikipedia web page are extracted. Next, hyperlinks present in the extracted paragraphs are followed

[3]Wikipedia, http://www.en.wikipedia.org
[4]JSoup HTML Parser, http://www.jsoup.org

and further paragraphs are extracted from the target web pages. This is continued until the predefined *reference level* is reached. The reference level states the limit of such a recursive hyperlink navigation. We set the reference level to 1 and the number of extracted paragraphs to 2 based on experimental results (see details in Section 6.7.2.3).

For instance, a part of the extracted textual description of category "lodging" from the source dataset is the following (terms in bold refer to linked pages and recursive extraction was executed for them):

Lodging *(or a holiday accommodation) is a type of residential* ***accommodation*** *(*author's note: refer to ***dwelling****). People who* ***travel*** *and stay away from home for more than a day need lodging for sleep, rest, safety, shelter from cold temperatures or rain, storage of luggage and access to common household functions. Lodging is done in a* ***hotel****,* ***motel****,* ***hostel*** *or* ***hostal****, a private home (commercial, i.e. a* ***bed and breakfast****, a* ***guest house****, a* ***vacation rental****, or non-commercially, with members of* ***hospitality services*** *or in the home of friends), in a* ***tent****, caravan****/camper*** *(often on a* ***campsite****) (...)*[5].

While a part of the textual description of category "Accommodation" from the target is the following:

Accommodation *may refer to: a* ***dwelling****, a place of temporary* ***lodging*** *(...)*[6]. *A* ***dwelling*** *(also residence, abode) is an important legal concept which defines a self-contained unit of accommodation used by one or more* ***households*** *as a* ***home****, such as a* ***house****,* ***apartment****, (...)*[7].

Category description. These texts are used to generate category descriptions (see paragraph Category description in Section 6.4). As a reminder, first, the preprocessing phase is applied which gets a unified set of terms. Namely, stop words are filtered out, special characters are removed and the algorithm for stemming is executed. Second, normalized *tfidf** weights of each term are calculated by Equation (6.4). For example, a part of the description of "lodging" contains the following most important terms (their stems) with weights: ("lodg", 0.3957), ("backpack", 0.3359), ("hous", 0.3051), ("room", 0.2277), ("accommod", 0.2257), ("residenti", 0.099), ("facil", 0.078), ("home", 0.081), ("household", 0.064), etc. While a part of the description of the category "Accommodation" from the target is the following: ("home", 0.3977), ("lodg", 0.3808), ("accommod", 0.1458), ("household", 0.1434), ("residenti", 0.131), ("facil", 0.041), etc.

Pairwise comparison. Finally, the category descriptions of under consideration are compared and a PTV is returned which expresses (un)certainty about

[5]The text is extracted from the definition of the term "lodging" in Wikipedia, `http://en.wikipedia.org/wiki/Lodging`

[6]The text is extracted from the definition of the term "Accommodation" in Wikipedia, `http://en.wikipedia.org/wiki/Accommodation`

[7]The text is extracted from the definition of *dwelling* in Wikipedia, `http://en.wikipedia.org/wiki/Dwelling`

the coreference of them. It is calculated like for the Description mapper in Section 6.4.1 by Equations (6.5) and (6.6). For example, the descriptions comparison of "lodging" and "Accommodation" returns the mapping with PTV equal (1,0.17). The uncertainty about this mapping is low because the description of "Accommodation" contains the description of "lodging". Thus, the results confirm common understanding that "lodging" is coreferent to "Accommodation".

6.4.3 Identification mappers

The Identification mapper identifies a category from the target in the *category description* of a category from the source (type I) or vice versa, detects a category from the source in the *category description* of a category from the target (type II). The category description can be constructed like for the Description mapper in Section 6.4.1 (then that mapper is called Category in description I or II) or like for the Definition mapper in Section 6.4.2 (then that mapper is called Category in definition I or II). Thus, we obtain four additional mappers. In what follows we discuss the details of type I Identification mapper.

First of all, the category description d^S for each category $c^S \in dom(a_C^S)$ from the source is constructed. Next, all categories $c^T \in dom(a_C^T)$ from the target are preprocessed by removing special characters and splitting it into single terms. Moreover, if a category description is constructed using stemming then also stemming is applied for this preprocessing of the all categories $c^T \in dom(a_C^T)$. Afterwards, if all terms of any category c^T are detected in the category description d^S of any category c^S then the mapping (c^S,c^T) is added to the set $\gamma_{1:n}$ of $\bowtie_{1:n}$-mappings, provided that the certainty of the mapping exceeds the threshold. The detection of category terms is based on the low level string comparison method [8], which is explained in Section 6.4.4. This technique has been chosen because of its efficiency and ability to take into account misspellings and abbreviations. In the literature a multitude of algorithms for string comparison has been proposed and these may also be employed here. An example of an interesting survey concerning string comparison in general is given in [2].

The low-level comparison method compares each pair of terms (a term of the target category with a term of the source category description) and generates PTVs which express the uncertainty about the coreference of the compared pairs of terms. Afterward, each term of the considered target category is mapped to the term from the source category description with the highest certainty about term coreference and the highest *tfidf** weight in case there exists more than one such a term.

Finally, this mapper generates a PTV which expresses the uncertainty about the coreference of the compared categories. The possibility that a proposition p, stating that two categories are coreferent, is true ($\mu_{\tilde{p}}(T)$) and the possibility that p

is false ($\mu_{\tilde{p}}(F)$) are calculated by the following equations:

$$\mu_{\tilde{p}}(T) = \frac{possT}{factor} \tag{6.14}$$

$$\mu_{\tilde{p}}(F) = \frac{possF}{factor} \tag{6.15}$$

where *possT*, *possF* and *factor* are equal:

$$possT = \frac{\sum_{i=1}^{|intersection|} \textit{tfidf}_i^{*S}}{|intersection|} \tag{6.16}$$

$$possF = (1 - possT)^{pow} \tag{6.17}$$

$$factor = \max(possT, possF) \tag{6.18}$$

where $|intersection|$ denotes the number of common terms and $\textit{tfidf}_i^{*S}$ is a weight of the common term from the description of the category from the source.

For instance, let us consider the categories "lodging" from the source and "Hotel" from the target. First of all, the category description for the "lodging" category is constructed like for the Description mapper (see Section 6.4.1). To this aim, tuples from the source dataset (cf., e.g., Table 6.1) are grouped into clusters of the same category. For each cluster values of tuple names are preprocessed (using such operations as stop words and special characters removal, cf., paragraph Category description in Section 6.4) which gives a unified set of terms for which normalized *tfidf** weights are calculated by Equation (6.4). The part of the description of "lodging" contains the following terms with *tfidf** (calculated for the entire source dataset): ("Hotel", 0.85), ("Bed", 0.22), ("Breakfast", 0.22), ("Resort", 0.08), ("Inn", 0.03), ("Carlton", 0.012), ("Borluut", 0.006), etc. Next, the target category "Hotel" is preprocessed (the same preprocessing is applied like for the constructed category description) which gives a term "Hotel". Finally, the term "Hotel" of the target category is compared to each term in the description of the source category "lodging" using a low-level string comparison method. In fact, the description contains a term "Hotel" which is equal to the term of the target category so the comparison gets PTV equal to (1,0). The weight of the detected term equals 0.85. Therefore, using the above equations the uncertainty of the mapping ("lodging","Hotel") equals (1,0.03) for pow equal 2.

6.4.4 Lexical mapper

The Lexical mapper compares the categories from the source with categories from the target and generates lexical mappings that are added to $\gamma_{1:n}$ if the certainty of

a mapping exceeds a predefined threshold. For instance, a Lexical mapper creates a mapping between the category "neighborhood" from the source and "Neighbourhood" from the target. However, the vast majority of the mappings that are established by this mapper are also suggested by other mappers, e.g. the Definition mapper and the Identification mappers, what have been experimentally confirmed in Section 6.7.2.5.

A one-level string comparison technique proposed in [8] (cf., also Section 2.3.3.2) is used to compare the categories. As a reminder, this low level comparison method estimates the possibility that two given strings are coreferent and is based on an approximation of *weak intersections*. It uses the concept of a moving window to construct the intersection of the two strings under comparison.

6.5 Implicit mappers

In addition to the explicit mappings, our method returns implicit mappings (lines 5-7 in Algorithm 6.1). We consider two types of implicit mappings. The first type of implicit mapping (Implicit mapping I) is established when no explicit mapping exists for the specific category c_1^S from the source. Let us assume that there exists the *closest more general* concept c_2^S in a partial order relation $\leq_{a_c}^S$ of the considered category $c_1^S \in dom(a_C^S)$ for which there exists an explicit mapping $\bowtie_{1:n} \in \gamma_{1:n}$. More specifically, the closest more general concept c_2^S is returned by the balanced selection (see Section 5.4), which uses a partial order relation $\leq_{a_c}^S$ and a set of all generalization of c_1^S (including the root concept, see Section 6.3.1). Thus, there exists only one the closest more general concept c_2^S of c_1^S, in the worse case the balance selection will return the root category.

Then the implicit mapping for c_1^S is thus each category of $\{c_1^T, ..., c_j^T\} \subseteq dom(a_C^T)$ such that $\{c_1^T, ..., c_j^T\}$ are the target categories to which c_2^S is mapped via $\bowtie_{1:n}$. It should be stressed that the quality of the implicit mappings depends on the correctness of the explicit mappings and the partial order relation.

For instance, $\leq_{a_c}^S$ in Figure 6.1 contains the following pairs of elements: ("establishment", "store"), ("store", "florist"), where "establishment" is the most general concept. Suppose that the set of already detected mappings contains the mapping ("store", "Shop"). If the category "florist" is not mapped explicitly then the algorithm returns an implicit mapping of "florist" to "Shop".

Moreover, the set of mappings is extended by the second type of implicit mappings (Implicit mapping II) which work as follows. For each target category c_j^T of a mapping (c_i^S,c_j^T), categories from the target partial order relation $\leq_{a_c}^T$ which are generalizations of the considered category c_j^T are extracted. Next, mappings between c_i^S and these extracted generalizations are established.

For instance, suppose that a mapping between "lodging" from the source and "Hotel" from the target exists. Using the partial order relation shown in Fig. 6.2

the generalizations of "Hotel" are the categories "Accommodation" and "POI". Thus, the mappings ("lodging", "Accommodation") and ("lodging", "POI") are established.

6.6 Selection heuristics

A set of novel heuristics is considered to establish the set $\gamma_{1:1}$ of *the best possible* one-to-one category mappings ($\bowtie_{1:1}$-mappings) from the set $\gamma_{1:n}$ of one-to-many categories mappings ($\bowtie_{1:n}$-mappings, one-to-one components of which are called candidate mappings), using the partial order relation $\leq_{a_c}^{T}$.

6.6.1 Uncertainty based mapping selection

This method selects a $\bowtie_{1:1}$-mapping for the particular source category which is most certain with respect to the associated PTV. It applies the order relation on PTVs (Equation 1.17). In case there exists more than one best mapping with the same PTV then the Simple Balanced Mappings Selection is applied, which is described below.

For instance, consider again the category "lodging" from the source and assume that it is mapped to a category "Accommodation" with uncertainty (1,0.12) and to a category "Hotel" with uncertainty (1,0.26). Considering the order relation on PTVs the mapping ("lodging","Accommodation") is selected.

6.6.2 The simple balanced mapping selection

This method is based on the balanced selection (see Section 5.4), which uses a partial order relation, in our case $\leq_{a_c}^{T}$. As a reminder, the balanced selection is the most specific category from the candidate categories provided that the selected category is comparable to all other candidate categories (that category is called *total*), otherwise the balanced selection is unspecified. As a consequence, the heuristics returns a mapping to the most specific *total* category if it exists or to a common more general category of candidate categories from $\leq_{a_c}^{T}$. For instance, consider again the category "lodging" from the source and assume that it is mapped only to the following candidate categories from the target in Figure 6.3:

- "Accommodation" and "POI". They are total so the balanced selection is the most specific category, "Accommodation"
- "Hotel", "Guest Room", "Accommodation" and "POI". Then the balanced selection is the category "Accommodation" because it is the most specific total category

- "Hotel" and "Guest Room". Then the balanced selection is unspecified because these categories are not total. Thus, the "lodging" will be mapped to "Accommodation" because it is the common closest more general category of candidate categories in the order relation $\leq_{a_c}^{T}$. More specifically, the common closest more general concept is returned by the balanced selection (see Section 5.4), which uses a partial order relation $\leq_{a_c}^{T}$ and a set of all generalization of candidate categories (including the root concept, see Section 6.3.1). Thus, there exists only one the common closest more general concept, in the worse case the balance selection will return the root category.

- "Guest Room" and "POI". They are total so the balanced selection chooses the most specific category, "Guest Room". However, that mapping is not equivalent and may not be true in both directions.

6.6.3 The increasing threshold balanced mapping selection

This method is similar to the Simple balanced mapping selection described in Subsection 6.6.2 but with one difference which may help to avoid the situation where the answer of the balanced selection is unspecified, i.e., if no category in the candidate set is total. With this method, categories from the candidate category set are eliminated by iteratively increasing the threshold for the certainty of candidate mappings (of candidate categories) until categories of the candidate set may be fused using the balanced selection or the candidate set is empty. Next, the non-empty reduced candidate set is used to select the proper mapping by the Simple balanced mappings selection method in Subsection 6.6.2 or if the reduced set is empty then a mapping to the common closest more general category of candidate categories from $\leq_{a_c}^{T}$ is selected.

For instance, consider again the category "lodging" from the source and assume that it is mapped only to the following candidate categories from the target and only mappings associated with a PTV below a threshold equal to 0.6 for the possibility of false are taken into account: "Restaurant" with uncertainty (1,0.52), "Accommodation" (1,0.12), "Guest Room" (1,0.33) and "Hotel" (1,0.26). For those candidate categories the balanced selection is unspecified because there is no total category based on the order relation in Figure 6.3. Therefore, in the next iteration the threshold for the certainty of candidate mappings is increased (threshold equal to 0.5 for the possibility of false). That eliminates the "Restaurant" category from the candidate set. The reduced candidate set is used to select the proper mapping by the Simple balanced mappings selection method, ("lodging","Accommodation").

6.6.4 The subset balanced mapping selection

This method also works like the Simple balanced mapping selection described in Subsection 6.6.2 with one difference which again may help to avoid the situation where the answer of the balanced selection is unspecified. Namely, the candidate category set is split into subsets and a subset with the largest cardinality, of which categories may be fused using the balanced selection, is used to select the proper mapping by the Simple balanced mappings selection method of Subsection 6.6.2. The criterion to split the candidate set into subsets is that a subset has to contain at least one category that is total. However, in the worst case where each subset is a singleton (a single element set), a mapping to the closest common more general category of candidate categories from $\leq_{a_c}^{T}$ is selected. Moreover, if there exist more than one subset with the largest cardinality, then these subsets are processed separately by the Simple Balanced Mappings Selection. Afterwards, the obtained results are fused again by the Simple balanced mappings selection.

This method (and the above heuristics) reduces the set of candidate categories which helps to select a mapping that points to a category different than the most general category in the partial order relation.

For instance, consider again the category "lodging" from the source and assume that it is mapped only to the following candidate categories from the target: "Restaurant", "Accommodation", "Guest Room" and "Hotel". For those candidate categories the balanced selection is unspecified because there is no total category based on the order relation in Figure 6.3. However, the candidate set can be split into two subsets that contain total categories. The first contains only "Restaurant" while the second consists of the other categories. Following the rules of this heuristics the second subset is selected. Finally, the balanced selection selects the most specific total category of it, i.e., "Accommodation".

6.6.5 The most popular mapping selection

This heuristics selects the mapping for the particular source category based on the frequency of mappings. A mapping between the source and target category has a high frequency if it is established by many different mappers. In case of indistinguishable mappings, i.e. equally frequent, the Simple Balanced Mappings Selection is applied. In other words, this method reduces the $\bowtie_{1:n}$-mapping to a $\bowtie_{1:1}$-mapping of which the source category maps to the candidate target category with the highest frequency.

For instance, consider again the category "lodging" from the source and assume that it is mapped only to the following candidate categories from the target: "Accommodation" by the Definition mapper; "Hotel" by the Description mapper; "POI", "Accom Shelter" and "Accommodation" (one more time) by the Implicit II Mapper. Following the rules of this heuristics the mapping to "Accommodation"

is selected.

6.6.6 The combined mapping selection

The above methods are combined in the last heuristics, the Combined mapping selection, whose details are presented in Algorithm 6.3. First of all, in lines 1-5, the $\bowtie_{1:n}$-mapping is reduced to a $\bowtie'_{1:n}$-mapping preserving only those mappings which are assigned the PTV (1,0). Next, three cases are considered.

Algorithm 6.3 COMBINEMAPPINGSELECTION

Require: Candidate $\bowtie_{1:n}$-mapping, Partial Order Relation $\leq^T_{a_c}$
Ensure: $\bowtie_{1:1}$-mapping
1: **for all** Mapping $\bowtie_{1:1} \in \bowtie_{1:n}$ **do**
2: **if** $\tilde{\bowtie}_{1:1}$ = (1,0) **then**
3: $\bowtie'_{1:n} \leftarrow \bowtie_{1:1}$
4: **end if**
5: **end for**
6: **if** $\bowtie'_{1:n}$ is $\bowtie_{1:1}$ **then**
7: **return** $\bowtie_{1:1}$
8: **else if** $\bowtie'_{1:n} \neq \varnothing$ **then**
9: $\bowtie'_{1:n} \leftarrow$ getMostPopular($\bowtie'_{1:n}$)
10: **if** $\bowtie'_{1:n}$ is $\bowtie_{1:1}$ **then**
11: **return** $\bowtie_{1:1}$
12: **else**
13: $\bowtie_{1:1} \leftarrow$ getBalancedSubset($\bowtie'_{1:n}, \leq^T_{a_c}$)
14: **if** $c^T \in \bowtie_{1:1} \neq$ root **then**
15: **return** $\bowtie_{1:1}$
16: **else**
17: $\bowtie_{1:1} \leftarrow$ getBalancedInc($\bowtie'_{1:n}, \leq^T_{a_c}$)
18: **return** $\bowtie_{1:1}$
19: **end if**
20: **end if**
21: **else if** $\bowtie'_{1:n} = \varnothing$ **then**
22: $\bowtie_{1:1} \leftarrow$ getBalancedSubset($\bowtie_{1:n}, \leq^T_{a_c}$)
23: **if** $c^T \in \bowtie_{1:1} \neq$ root **then**
24: **return** $\bowtie_{1:1}$
25: **else**
26: $\bowtie_{1:1} \leftarrow$ getBalancedInc($\bowtie_{1:n}, \leq^T_{a_c}$)
27: **return** $\bowtie_{1:1}$
28: **end if**
29: **end if**

First, lines 6-7 in Algorithm 6.3, if the reduced $\bowtie'_{1:n}$-mapping is de facto $\bowtie_{1:1}$-

mapping, i.e., the source category is mapped to exactly one target category, then that mapping is selected.

Otherwise, lines 8-20 in Alg. 6.3, if the reduced $\bowtie'_{1:n}$-mapping is not empty (line 8) then first of all the Most popular mapping selection heuristics (getMostPopular) of Section 6.6.5 is applied (line 9). The method getMostPopular reduces the $\bowtie'_{1:n}$-mapping to a new $\bowtie'_{1:n}$-mapping of which the source category maps to the candidate target category with the highest frequency (or categories if there exist more than one mapping with the highest frequency, in contrast to the original implementation, which returns only one category). Next, if the new reduced $\bowtie'_{1:n}$-mapping is a $\bowtie_{1:1}$-mapping, i.e., if the source category is mapped to exactly one target category, then that mapping is selected (lines 10-11). Otherwise, the Subset balanced mapping selection (getBalancedSubset) of Section 6.6.4 is applied (line 13). The method getBalancedSubset gets the $\bowtie_{1:1}$-mapping and is selected if it does not map to the root (lines 14-15). Otherwise, the Increasing threshold balanced mapping selection (getBalancedInc) of Section 6.6.3 is applied and the method getBalancedInc returns the $\bowtie_{1:1}$- mapping (line 17-18).

Otherwise, lines 21-29 in Algorithm 6.3, if the reduced $\bowtie'_{1:n}$-mapping is empty, i.e., if the source category does not map to any target category with certainty equal to (1,0), then the Subset balanced mapping selection (getBalancedSubset) of Section 6.6.4 and the Increasing threshold balanced mapping selection (getBalancedInc) of Section 6.6.3 are applied, like in the second case but using the input candidate $\bowtie_{1:n}$-mapping.

6.7 Evaluation and discussion

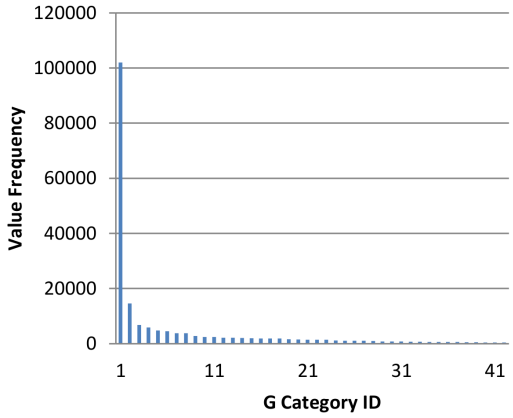
In this section an experimental evaluation of our method shows the influence of the parameters and the benefits of using semantical mappers and partial order relations with respect to the Lexical mapper.

6.7.1 Datasets

The evaluation of our approach is conducted based on four real-world datasets. The first and second dataset contain information about points of interest (POIs, see Section 6.1). Within the context of the experiments, the attribute of interest is the category of the POI, which specifies the type and/or function of the location. The first dataset contains 193816 tuples, which are extracted from the Google Maps database by the Google Places API[6] (a part of this dataset is used in Chapter 3). A partial order relation on the set consists of 98 categories and is constructed by experts. The second dataset contains 436616 normalized tuples and a partial order

[6]Google Places, http://developers.google.com/places/

Figure 6.4 Typical distribution of values for the category attribute (G dataset).



relation on the set of 502 categories, which are shared by RouteYou[7] (it is the R-EN dataset which is described in Section 2.4).

Two other datasets are restaurant datasets [9]. The on-line guides *Zagat* and *Fodor*, which are also described in Section 2.4. As a reminder, like for the POI datasets, the attribute of interest is the category of the restaurant, which specifies the cuisine of the restaurant. Zagat contains 331 tuples and a partial order relation on the set of 169 categories, which is downloaded from the Zagat website[8][9]. On the other hand, Fodor contains 533 tuples and a partial order relation on the set of 32 categories, which is constructed by experts.

6.7.1.1 Values distribution

Values of the attribute "category" have a typical *long tail* distribution for each dataset considered in this evaluation. This means that a few values occur many times, while most values occur very seldom. That typical distribution for the category attribute in the Google (G) dataset is shown in Figure 6.4. Plots for other datasets are omitted because they are similar.

6.7.1.2 Test cases

Datasets of the same domains are pairwise compared and their categories are mapped, namely POI datasets and restaurant datasets, respectively. Thus, in our experiments we consider four test cases:

[7]RouteYou, http://routeyou.com/
[8]Zagat, http://zagat.com
[9]This data was removed from the Zagat website after redesign on 29 July 2013.

- the categories of tuples from the Google dataset (the source, G) are mapped to the categories of the RouteYou dataset (the target, R), called the G-R dataset for short
- the categories of tuples from the RouteYou dataset (the source, R) are mapped to the categories of the Google dataset (the target, G), called the R-G dataset for short
- the categories of tuples from the Zagat dataset (the source, Z) are mapped to the categories of the Fodor dataset (the target, F), called the Z-F dataset for short
- the categories of tuples from the Fodor dataset (the source, F) are mapped to the categories of the Zagat dataset (the target, Z), called F-Z dataset for short

Remark. In addition to general POIs (which also exist in the RouteYou dataset, e.g., touristic attractions or public places) the Google dataset also contains many business locations, e.g., "lawyer" or "plumber". In contrast, the RouteYou dataset is a specialized dataset that is focused on route planning for outdoor activities. Thus, we will not find business locations in the RouteYou dataset except those that may be useful for a user, e.g., "bicycle shop" or "rent a bike company". Whereas, the main difference between the restaurant datasets is that the Zagat dataset is more detailed than the Fodor dataset, namely Zagat contains more specialized categories which do not exist in Fodor. Thus, the known taxonomy of Zagat's categories is larger and more complex.

Table 6.3 Datasets details

Datasets	# eq.	# gen.	# spec.	# all	$\lvert dom'(a_C^S)\rvert$
G-R	61	208	289	558	98
R-G	56	279	180	515	424
Z-F	26	160	61	247	59
F-Z	33	153	313	499	34

6.7.1.3 Real mappings

Considering the above datasets, it should be clear that not all categories are mapped. The sets of possible real coreferent equivalent and non-equivalent mappings are provided manually by experts for each dataset (test case), namely G-R, R-G, Z-F and F-Z. For the datasets which are mapped reversely the mappings are different because there are removed mappings of categories which do not exist in the

Table 6.4 Datasets details: distribution of real category mappings without mappings to the root of the target.

Datasets	# 1:1 trivial	# 1:n non-trivial	# non-mapped
G-R	17 (18%)	68 (69%)	13 (13%)
R-G	146 (34%)	115 (28%)	163 (38%)
Z-F	3 (5%)	52 (88%)	4 (7%)
F-Z	0 (0%)	34 (100%)	0 (0%)

source dataset (we mapped only the categories that were actually present in the source dataset). Moreover, each category from the source can be mapped by Definition 20 to the most general category (according to the partial order relation $\leq_{a_c}^{T}$) of the target, called *root*, which in our case is "POI" in dataset R, "establishment" in dataset G and "Specialties" in datasets F and Z. However, the objective of our approach is to be more specific and to establish as many mappings different from the mapping to the root as possible. Thus, we do not consider real mappings as well as the detected mappings to the root to make our evaluation reliable.

Table 6.3 contains a summary of the real mappings sets. Columns 2-4 represent the number of equivalent, generalized and specialized real mappings for each dataset, respectively. Column 5 (# all) contains a sum of them. The last column ($|dom'(a_C^S)|$) shows the number of distinct categories from the source dataset, which should be mapped to the target categories in each test case. However, not all of them can be mapped because of a lack of a suitable (proper) category in the target. Table 6.4 presents how many source categories are mapped only to one target category (# 1:1 trivial), to more than one (# 1:n non-trivial) and any at all (# non-mapped). In Figure 6.5 the detailed distribution of real $\bowtie_{1:n}$-mappings is shown. The most popular are mappings with cardinality between 1 and 10. It reveals that there exist many alternative mappings which makes the detection of (proper) coreferent categories a challenging task.

Finally, tuples that are mapped by the 1:1 and 1:n real category mappings for each dataset are reported in Table 6.5 in columns 4 and 5. Table 6.5 also contains the number of all tuples (# tuples) and tuples with a category different from the root (# tuples') of the source dataset in each test case. Each dataset, besides the dataset F, contains many tuples which are not mapped by any real category mapping. It means that the target dataset does not contain corresponding category and confirms the differences of datasets. Moreover, most of the tuples in each test case are mapped by non trivial 1:n mappings. This confirms the need of selection heuristics.

Figure 6.5 Distribution of cardinality of the real $\bowtie_{1:n}$-mappings for each dataset (in the sense of "n" in 1:n).

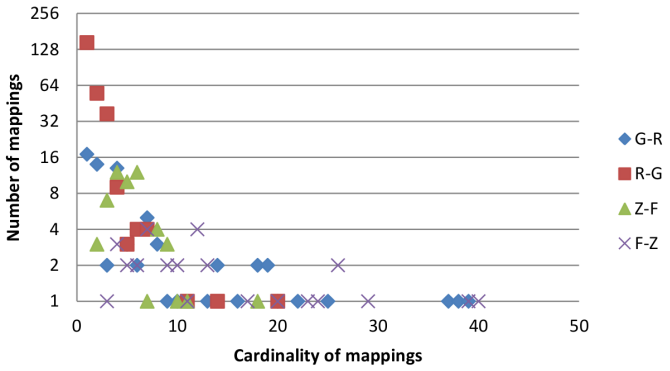


Table 6.5 Datasets details: tuples mapped by real category mappings without mappings to the root of the target.

Datasets	# tuples	# tuples'	# 1:1 mappings	# 1:n mappings
G-R	193816	91822	13165	64385
R-G	436616	399197	101681	207013
Z-F	341	341	19	307
F-Z	550	550	0	550

6.7.1.4 Evaluation metrics

The quality of our method is evaluated using two standard measures of recall and precision. The precision is the fraction of detected real coreferent mappings among all detected mappings; the recall is the number of detected real coreferent mappings divided by the number of all real coreferent mappings.

6.7.2 Category mapping

In this section the Category Mapping Algorithm 6.1 is evaluated in detail. First of all, the parameters of the mappers are selected based on experimental results. Next, the detected mappings by our method are compared to real mappings and mappings generated only by a Lexical mapper (to show the advantages of using our semantical mappers).

Table 6.6 Calibrated pow parameter for the particular mapper

Mapper	**Parameter pow**
Description	2.2
Definition	1.4
Category in Description	7
Category in Definition	9

6.7.2.1 Experiment: Calibrated pow parameter for the particular mapper

Goal. Our algorithm employs threshold to decide on category coreference, which is set to 0.5 for $\mu_{\tilde{p}}(F)$ of each mapper. As a reminder, if $\mu_{\tilde{p}}(F)$ is lower than the threshold then the coreference is declared (see Section 1.4.4).

In order to make the result of each mapper comparable (i.e. each mapper gets a set of mappings with similar precision for the same threshold), we have to set the parameter pow in Equation (6.7) of Description and Definition mappers and also in Equation (6.16) of Identification mappers (it raises to the pow power a component of equations which calculate possibility of mapping). The selection of proper values for the parameter pow is experimentally confirmed in the following experiment.

Procedure. For the parameter pow, a range from 1 to 20 is considered. The parameter pow for each mapper is set to get first of all mappings with high precision and if possible also high recall for predefined threshold equal to 0.5. However, the high recall for each mapper considered separately is of a minor importance because real mappings that are not detected by one mapper can be detected by others. On the other hand, a high precision is extremely important because false positive mappings are propagated by Implicit Mappers and they can significantly decrease the quality of the final result.

Result. The value of the parameter pow for each mapper has been experimentally confirmed based on the dataset G-R to make the results of all types of mappers comparable. Figure 6.6 shows precision and recall for different values of the parameter pow for each mapper (uninterested results are omitted). Table 6.6 summarizes the selected values of pow for each mapper.

6.7.2.2 Experiment: Comparison of category description aggregation (cf. Section 6.4.1)

Goal. In Section 6.4.1 different equations (6.10)-(6.13) to calculate the PTVs are proposed. This experiment evaluates these calculations.

Figure 6.6 Setup mappers threshold: parameter pow.

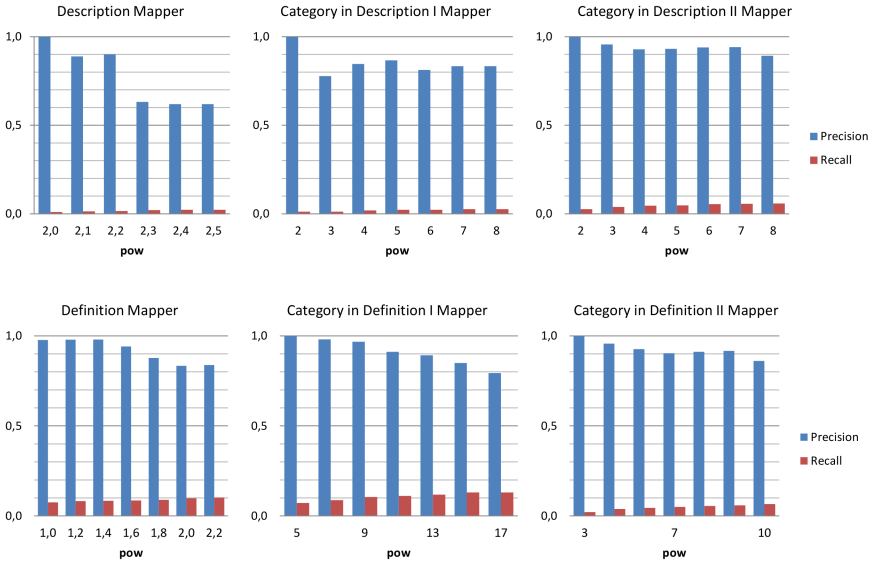


Table 6.7 Results of calculating PTV of mappings by different equations (6.10)-(6.13).

$\bowtie_{1:1}$-mapping	Avg	Max	Min	Pr. S	Pr. T
storage - Warehouse	(1,0.08)	(1,0.02)	(1,0.03)	(1,0.02)	(1,0.03)
restaurant - Road Rest.	(1,0.28)	(1,0.06)	(1,0.12)	(1,0.06)	(1,0.11)
lodging - Hotel	(1,0.26)	(1,0.01)	(1,0.02)	(1,0.02)	(1,0.11)
church - Place of Worship	(1,0.30)	(1,0.01)	(1,0.02)	(1,0.03)	(1,0.12)

Procedure. The Description mapper and Definition mapper are executed in sequence with different possibility computations.

Result. Table 6.7 compares the uncertainty of the mappings detected in the G-R dataset, the two mappings above the bar are detected by the Definition mapper, the two mappings below the bar by the Description mapper. The calculation used are: Average Tfidf (Equation 6.10) (column 3), Max Tfidf (Equation 6.12) (column 4), Min Tfidf (Equation 6.11) (column 5) and Source Preference (Equation 6.13) (columns 6-7). The Average measure best captures the expert opinion (i.e. reasonable uncertainty exists for not equivalent categories like “lodging” and “Hotel”). Thus, it is selected for further evaluation.

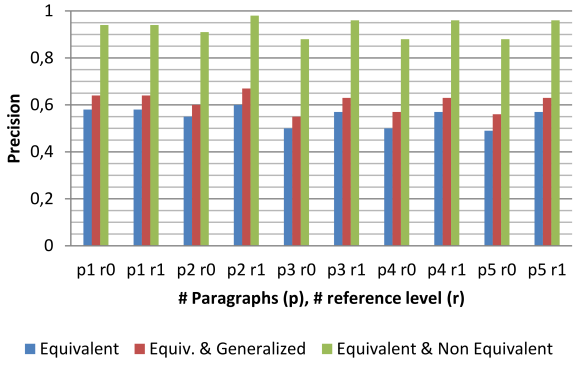
6.7.2.3 Experiment: Tuning of the definition mapper

Goal. In this experiment the influence of the number of extracted paragraph and the reference level of the Definition mapper (see Section 6.4.2) on the uncertainty of the detected mappings is presented.

Procedure. The number of the extracted paragraphs from an external source (e.g., from Wikipedia article) and reference level are ranged over a set of alternatives in order to investigate their influence on the G-R dataset. For the number of extracted paragraphs, a range from 1 to 5 is considered. For the reference level, values 0 and 1 are considered. It gives us 10 test cases whose results are as follows.

Result. Figure 6.7 presents the precision of the test cases. The highest precision is achieved for 2 paragraphs and reference level 1. Namely, 0.6 for equivalent mappings, 0.67 for equivalent and generalized mappings, and 0.98 for equivalent and non-equivalent mappings. This reflects that, i.e. the most defining keywords, synonyms or more general concepts are usually located at the beginning of a concept definition which can also be observed by investigating references.

Figure 6.7 Parameters setup of the Definition mapper.



6.7.2.4 Experiment: Comparison to real mappings

Goal. In this experiment mappings detected by explicit and implicit mappers (see Section 6.4 and 6.5) are compared to real mappings which are established by experts (see details of evaluated datasets in Section 6.7.1).

Procedure. Our algorithm is applied to respectively detect mappings by each explicit mapper separately, by all explicit mappers together and by a combination of

all explicit mappers and all implicit mappers. Established mappings are compared to real mappings using precision and recall measures.

Result. Parts of the results are presented in Table 6.8 and 6.9 for the G-R and F-Z datasets respectively (the R-G and Z-F datasets are omitted). Each mapping consists of four values: the source category, the target category, the PTV and the name of the mapper that produced the specific mapping. There can be distinguished equivalent mappings (i.e. "lodging" and "Accommodation", "Barbeque" and "BBQ"), generalized non-equivalent mappings (i.e. "church" and "Place of Worship", "Barbeque" and "American") and specialized non-equivalent mappings (i.e. "lodging" and "Hotel", "Mexican" and "Spanish").

Tables 6.10, 6.11, 6.12 and 6.13 present the quality measures of our approach for equivalent mappings (column 2), equivalent and generalized non-equivalent mappings (column 3) and equivalent and all non-equivalent mappings (column 4) for G-R, R-G, Z-F and F-Z datasets respectively. Additionally, the last column in these Tables contains recall and precision for *any but not the most general* mappings (Not Root). This column shows if there is **at least one** coreferent mapping established for each category from the source which is not a mapping to the most general category in the partial order relation $\leq_{a_c}^{T}$. Thus, in this case the precision and recall are calculated as follows. The precision is the number of distinct source categories (different from the root) of detected real coreferent mappings divided by the number of distinct source categories (different from the root) of all detected mappings, the recall is the number of distinct source categories (different from the root) of detected real coreferent mappings divided by the number of distinct source categories (different from the root) of all real coreferent mappings.

First of all, the recall and precision are calculated for explicit mappers in Tables 6.10, 6.11, 6.12 and 6.13 (above the bar). On the one hand, the Definition, Category in Definition I/II and also Lexical mappers get equivalent mappings with high precision and recall compared to the Description and Category in Description I/II mappers (column 2). On the other hand, all explicit mappers return high precision but low recall for equivalent and non-equvalent mappings (column 4). The high precision is important because these mappings are used by the implicit mappers: any false positive mapping is propagated by the implicit mappers, which degrades the overall result. Besides that, it is not crucial to detect all possible coreferent mappings but it is sufficient if the algorithm creates at least one coreferent mapping for each category that turns out to be the proper mapping (the selection of mappings is investigated in Section 6.7.3). The results in the last column confirm that. For about half of the categories from the source in the G-R dataset (one-third in the R-G dataset), at least one coreferent mapping is established that is different from the root with a precision that equals to 1 (0.98 in the R-G, respectively). Meanwhile the recall equals 0.58 for the Z-F dataset or even 0.79 for the

Table 6.8 Some results of the Category Mapping Algorithm: mappings of values from the source and target datasets (G-R).

Category S	**Category** T	**PTV**	**Mapper**
restaurant	Eatery	(1,0)	Definition
restaurant	Eat and Drink	(1,0)	ImplicitII
restaurant	POI	(1,0)	ImplicitII
restaurant	Road Restaurant	(1,0.28)	Definition
restaurant	Road Restaurant	(1,0.12)	Category in definition
restaurant	Hotel	(1,0.37)	Description
church	church	(1,0)	Definition
church	Place of Worship	(1,0)	ImplicitII
church	POI	(1,0)	ImplicitII
church	Place of Worship	(1,0.3)	Description
lodging	Hotel	(1,0.26)	Description
lodging	Hotel	(1,0.03)	Category in description
lodging	Accomodation	(1,0.12)	Category in definition
lodging	Accom Shelter	(1,0.26)	ImplicitII
lodging	POI	(1,0.26)	ImplicitII
food	Restaurant	(1,0.29)	Category in definition
food	Eatery	(1,0.29)	ImplicitII
food	Eat and Drink	(1,0.1)	ImplicitII
food	POI	(1,0.1)	ImplicitII
meal delivery	Restaurant	(1,0.29)	Implicit
meal delivery	Eatery	(1,0.29)	ImplicitII
meal delivery	Horeca	(1,0.29)	ImplicitII
meal delivery	Eat and Drink	(1,0.29)	ImplicitII
meal delivery	POI	(1,0.29)	ImplicitII

F-Z dataset with a precision that equals 1.

Next, recall and precision are calculated for the explicit and implicit mappings combined. The implicit mappers decrease the precision for equivalent mappings (column 2 below the bar in Tables 6.10, 6.11, 6.12 and 6.13) because they create mainly non-equivalent mappings. Thus, for non-equivalent mappings (column 3 and 4), these mappers in particular increase the recall, e.g. the recall increases from 0.18 to 0.41 for equivalent and generalized mappings of the G-R dataset in

Table 6.9 Some results of Category Mapping Algorithm: mappings of values from the source and target datasets (F-Z).

Category S	**Category** T	**PTV**	**Mapper**
Barbeque	BBQ	(1,0.22)	Category in description
Barbeque	Cuisine	(1,0.22)	Implicit II
Barbeque	American	(1,0.22)	Implicit II
Barbeque	BBQ	(1,0.43)	Category in definition
Barbeque	BBQ	(1,0)	Definition
Mexican	Mexican	(1,0)	Definition
Mexican	Mexican	(1,0)	Lexical
Mexican	Mexican	(1,0.07)	Category in definition
Mexican	Tex-Mex	(1,0)	Category in definition
Mexican	Southwestern	(1,0.01)	Category in definition
Mexican	Spanish	(1,0.21	Lexical

Table 6.10 (column 3). Besides that, the recall of at least one coreferent mapping that is different from the root (last column) is also increased at the expense of a slight decrease in precision.

Table 6.10 Precision (P) and Recall (R) of established one-to-many mappings: G-R dataset.

Mapper	**Eq.**		**Eq. & Gen.**		**Eq. & Non-eq.**		**Not Root**	
	P	**R**	**P**	**R**	**P**	**R**	**P**	**R**
Description	0.14	0.03	0.29	0.01	0.64	0.02	0.78	0.08
Definition	0.60	0.48	0.67	0.12	0.98	0.08	1.00	0.33
Cat. in desc. I	0.71	0.08	0.86	0.02	1.00	0.01	1.00	0.08
Cat. in desc. II	0.60	0.15	0.67	0.04	1.00	0.03	1.00	0.13
Cat. in def. I	0.44	0.44	0.49	0.11	0.97	0.10	1.00	0.31
Cat. in def. II	0.67	0.39	0.81	0.11	0.92	0.06	1.00	0.34
Lexical	0.69	0.41	0.69	0.09	0.92	0.06	0.96	0.29
All	0.36	0.61	0.49	0.18	0.90	0.16	1.00	0.49
All w/o Lexical	0.36	0.57	0.50	0.18	0.93	0.16	1.00	0.47
All +Impl.	0.18	0.62	0.54	0.41	0.80	0.29	0.96	0.54

Table 6.11 Precision (P) and Recall (R) of established one-to-many mappings: R-G dataset.

Mapper	Eq.		Eq. & Gen.		Eq. & Non-eq.		Not Root	
	P	R	P	R	P	R	P	R
Description	0.14	0.04	0.50	0.02	0.64	0.02	0.88	0.03
Definition	0.60	0.52	0.94	0.13	0.98	0.09	1.00	0.17
Cat. in desc. I	0.60	0.16	0.93	0.04	1.00	0.03	1.00	0.06
Cat. in desc. II	0.71	0.09	0.86	0.02	1.00	0.01	1.00	0.03
Cat. in def. I	0.67	0.43	0.81	0.09	0.92	0.06	0.91	0.12
Cat. in def. II	0.44	0.48	0.92	0.17	0.97	0.11	1.00	0.21
Lexical	0.69	0.45	0.92	0.10	0.92	0.06	0.97	0.13
All	0.36	0.66	0.79	0.24	0.90	0.18	0.98	0.31
All w/o Lexical	0.36	0.63	0.80	0.23	0.93	0.17	0.98	0.30
All +Impl.	0.13	0.70	0.69	0.63	0.74	0.44	0.83	0.59

Table 6.12 Precision (P) and Recall (R) of established one-to-many mappings: Z-F dataset.

Mapper	Eq.		Eq. & Gen.		Eq. & Non-eq.		Not Root	
	P	R	P	R	P	R	P	R
Description	–	–	–	–	–	–	–	–
Definition	0.86	0.92	0.93	0.14	1.00	0.11	1.00	0.44
Cat. in desc. I	1.00	0.04	1.00	0.01	1.00	0.01	1.00	0.02
Cat. in desc. II	–	–	–	–	–	–	–	–
Cat. in def. I	0.72	0.50	0.83	0.08	1.00	0.07	1.00	0.29
Cat. in def. II	0.68	0.65	1.00	0.13	1.00	0.10	1.00	0.44
Lexical	0.88	0.85	1.00	0.13	1.00	0.10	1.00	0.45
All	0.64	0.96	0.90	0.19	1.00	0.16	1.00	0.58
All w/o Lexical	0.64	0.96	0.90	0.19	1.00	0.16	1.00	0.58
All +Impl.	0.19	0.96	0.88	0.63	0.96	0.52	1.00	0.84

6.7.2.5 Experiment: Lexical vs semantical mappers

Goal. This experiment was conducted to show the advantages of using our semantical explicit mappers over the Lexical mapper.

Table 6.13 Precision (P) and Recall (R) of established one-to-many mappings: F-Z dataset.

Mapper	Eq.		Eq. & Gen.		Eq. & Non-eq.		Not Root	
	P	**R**	**P**	**R**	**P**	**R**	**P**	**R**
Description	–	–	–	–	–	–	–	–
Definition	0.86	0.72	0.93	0.14	1.00	0.06	1.00	0.71
Cat. in desc. I	–	–	–	–	–	–	–	–
Cat. in desc. II	1.00	0.03	1.00	0.01	1.00	0.01	1.00	0.03
Cat. in def. I	0.68	0.52	0.68	0.09	1.00	0.05	1.00	0.56
Cat. in def. II	0.72	0.39	0.89	0.09	1.00	0.04	1.00	0.47
Lexical	0.88	0.67	0.88	0.12	1.00	0.05	1.00	0.65
All	0.64	0.76	0.74	0.16	1.00	0.08	1.00	0.79
All w/o Lexical	0.64	0.76	0.74	0.16	1.00	0.08	1.00	0.79
All +Impl.	0.27	0.76	0.87	0.44	0.98	0.18	1.00	0.97

Procedure. First of all, mappings that are detected by explicit mappers (except the Lexical mapper) are compared to mappings that are established only by the Lexical mapper. Next, these mappings (i.e., mappings that are detected by explicit mappers except the Lexical mapper) are compared to mappings that are established by all explicit mappers.

Result. Rows 7-9 in Tables 6.10-6.13 present the results of this comparison. The Lexical mapper gets slightly better results than other combined mappers only for equivalent mappings (rows 7 and 9). This is understandable because the equivalent mappings are mostly syntactically equal. However, for non-equivalent mappings our other mappers get better results than the Lexical mapper (higher recall with high precision).

In addition, most of the mappings that are established by the Lexical mapper are suggested also by the other mappers, especially by the Definition mapper. In general, we get almost the same result for mappings that are established by all mappers with or without the Lexical mapper (rows 8 and 9). However, for equivalent mappings the Lexical mapper slightly increases the recall, which makes it worth using with our method.

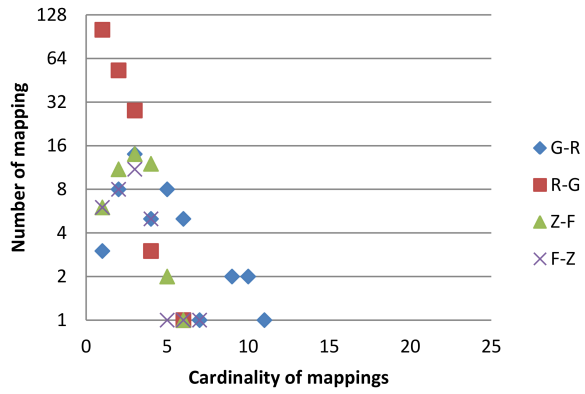
6.7.3 Selection of mappings

In this section the Category Mapping Selection Algorithm 6.2 is evaluated in detail. Namely, the reduction of the set $\gamma_{1:n}$ of $\bowtie_{1:n}$-mappings to the set $\gamma_{1:1}$ of $\bowtie_{1:1}$-

mappings is evaluated. $\bowtie_{1:n}$-mappings are established by explicit and implicit mappers with parameters that are set based on the experimental results that are reported in Section 6.7.2.

First of all, in Figure 6.8, the detailed distribution of detected $\bowtie_{1:n}$-mappings is shown. The most popular mappings are those with a cardinality between 1 and 5, while the maximum cardinality does not exceed 11. This shows that many alternative mappings still exist. This makes the detection of (proper) coreferent categories a challenging task.

Figure 6.8 Distribution of cardinality of the detected $\bowtie_{1:n}$-mappings (in the sense of "n" in 1:n).



6.7.3.1 Experiment: Object specific mappings selection

Goal. The object specific category mappings, which are established by the method in line 6 of Algorithm 6.2, are evaluated in this experiment.

Procedure. The method is executed in sequence with the following datasets: G-R, R-G, Z-F and F-Z.

Result. The results of this experiment are presented in the rows of group 3 in Table 6.14. The tuples that are mapped with precision between 0.79 and 1, are only 1.28%-2.55% of all tuples from the considered datasets. However, each of these tuples is mapped with high precision to the category that is specific for the considered object, i.e. a hotel with the category "lodging" is mapped to "Hotel" or a car repair company with the category "establishment" is mapped to "Garage", etc.

Moreover, the remainder of the input tuples (namely between 72% and 97% depending on the dataset) are mapped using the default category mapping selection

heuristic from Section 6.6 (line 8 in Algorithm 6.2 and are evaluated in details in the next experiment). More precisely, the number and percentage of mapped tuples with a category different from the root are shown in Table 6.14 in the rows of group 4, while in the rows of group 5 tuples with a category equal to the root are mapped. In total, 75%-99% of the tuples are mapped (the rows of group 2). The unmapped tuples in the rows of group 6 (1%-25%) are tuples of which the category is not mapped by any mapper.

Table 6.14 Results of tuples mapping.

	Dataset	**G-R**	**R-G**	**Z-F**	**F-Z**
1	# all tuples	193816	436616	341	550
2	# all mapped tuples	144729	337063	297	545
	% all mapped tuples	74.67%	77.20%	87.10%	99.09%
3	# tuples: Specific Mapping	4303	5585	8	14
	% tuples: Specific Mapping	2.22%	1.28%	2.35%	2.55%
	Precision: Specific Mapping	0.79	0.94	1.00	1.00
4	# tuples: Default Mapping	40576	294964	289	531
	% tuples: Default Mapping	20.93%	67.56%	84.75%	96.55%
5	# tuples: Default Mapping Root	99850	36514	0	0
	% tuples: Default Mapping Root	51.52%	8.36%	–	–
6	# not mapped tuples	49087	99553	44	5
	% not mapped tuples	25.33%	22.80%	12.90%	0.91%

6.7.3.2 Experiment: Default mappings selection

Goal. A comparison of the selection heuristics that are proposed in Section 6.6 is the goal of this experiment.

Procedure. These heuristics are executed in sequence on the set $\gamma_{1:n}$ of $\bowtie_{1:n}$-mappings, which are suggested by our mappers. This experiment is repeated for each dataset, namely G-R, R-G, Z-F and F-Z.

Result. Table 6.15 presents the precision and recall for the heuristics used in our experiments. In general the best result, i.e. a high precision with a high recall, was found for three heuristics, namely uncertainty-based selection, popularity-based selection and combined selection (see Sections 6.6.1, 6.6.5 and 6.6.6). The good result of the popularity-based heuristics is due to a high number of mappers (and

their quality) that are used in our approach. Meanwhile, the good result of the uncertainty-based heuristics confirms the effectiveness of the uncertainty model applied to our framework. Finally, the high quality of the results of the Combined selection is understandable because this method relies mostly on the other winning heuristics.

Table 6.15 Precision (P) and Recall (R) of mappings selection.

Dataset Heuristics	**Eq.**		**Eq. & Gen.**		**Eq. & Non-eq.**	
	P	**R**	**P**	**R**	**P**	**R**
G-R Simple	0.59	0.44	0.88	0.36	0.97	0.38
G-R Subset	0.47	0.51	0.69	0.40	0.90	0.51
G-R Incr. Thr.	0.59	0.60	0.83	0.45	0.93	0.49
G-R Uncertainty	0.65	0.71	0.73	0.43	0.90	0.51
G-R Popularity	0.65	0.71	0.84	0.49	0.96	0.54
G-R Combine	0.69	0.76	0.86	0.50	0.94	0.53
R-G Simple	0.18	0.55	0.70	0.55	0.75	0.50
R-G Subset	0.17	0.57	0.69	0.57	0.75	0.53
R-G Incr. Thr.	0.19	0.63	0.70	0.58	0.76	0.54
R-G Uncertainty	0.20	0.68	0.75	0.62	0.80	0.56
R-G Popularity	0.20	0.68	0.75	0.62	0.80	0.56
R-G Combine	0.20	0.68	0.74	0.61	0.78	0.55
Z-F Simple	0.43	0.73	1.00	0.80	1.00	0.80
Z-F Subset	0.46	0.81	1.00	0.84	1.00	0.84
Z-F Incr. Thr.	0.43	0.73	1.00	0.80	1.00	0.80
Z-F Uncertainty	0.54	0.96	1.00	0.84	1.00	0.84
Z-F Popularity	0.54	0.96	1.00	0.84	1.00	0.84
Z-F Combine	0.52	0.92	0.98	0.82	1.00	0.84
F-Z Simple	0.58	0.69	0.81	0.76	0.94	0.85
F-Z Subset	0.58	0.73	0.79	0.79	0.94	0.91
F-Z Incr. Thr.	0.58	0.69	0.81	0.76	0.94	0.85
F-Z Uncertainty	0.73	0.92	0.94	0.94	1.00	0.97
F-Z Popularity	0.73	0.92	0.94	0.94	1.00	0.97
F-Z Combine	0.73	0.92	0.94	0.94	1.00	0.97

6.7.3.3 Experiment: Influence of mappings set

Goal. In this experiment the influence of the cardinality and quality of one-to-many mappings on the mapping selection heuristics is evaluated. In other words, it is studied how selection heuristics depend on the relaxed mapper threshold for $\mu_{\tilde{p}}(F)$.

Procedure. The Most popular selection method (see Section 6.6.5) is executed for each dataset in sequence with a threshold for $\mu_{\tilde{p}}(F)$ ranging from 0.1 to 0.9. A more relaxed threshold returns more candidates and also decreases the quality of mappings. The Most popular heuristics is selected because it is one of the methods that gets the best results in our evaluation in Section 6.7.3.2.

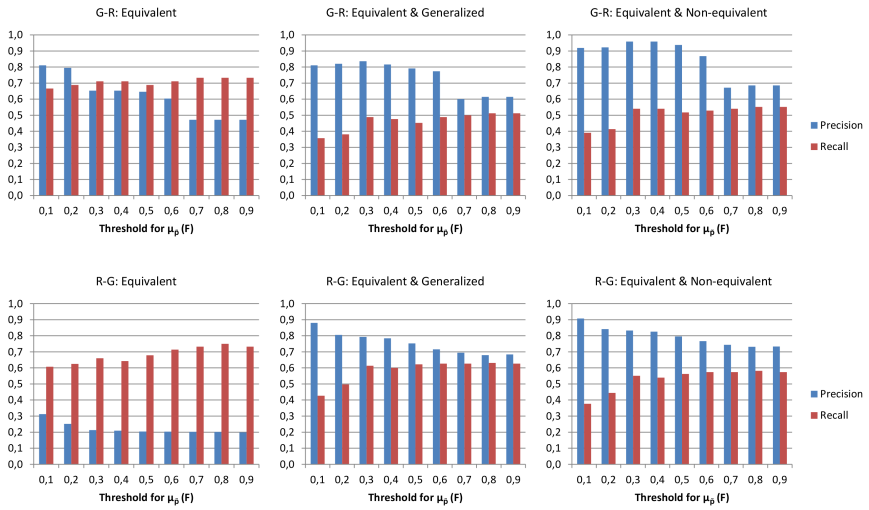
Result. Figure 6.9 presents the recall and precision of the Most popular selection heuristics for different thresholds for $\mu_{\tilde{p}}(F)$ for datasets G-R and R-G. On the one hand, the best results for equivalent mappings are obtained for a threshold $\mu_{\tilde{p}}(F)$ smaller than 0.2. It is understandable that in accordance to expertise the equivalent mappings have the lowest uncertainty. On the other hand, this is not so obvious for non-equivalent mappings. The best result are achieved for a threshold value between 0.4-0.5. The other datasets, Z-F and F-Z, return good results for any value of the threshold for $\mu_{\tilde{p}}(F)$ (consider for example the threshold value 0.5 in the experiment in Section 6.7.3.2) because the one-to-many mapping sets contain mappings with high certainty, i.e, for almost each category from the source there exists at least one true positive mapping with uncertainty lower than (1,0.1). In addition, it should be clear that a correct one-to-many mappings help to find precise one-to-one mappings.

6.7.3.4 Experiment: Lexical vs semantical mappings

Goal. This experiment was conducted to show the advantages of using our semantical explicit mappers (the Description, Definition, Identification mappers) over the Lexical mapper in the context of mappings selection by our heuristics in Section 6.6.

Procedure. First of all, the selection of mappings by the Most popular mappings heuristics is executed on the set of $\bowtie_{1:n}$-mappings, which are established by semantical explicit mappers and next on the set of $\bowtie_{1:n}$-mappings, which are suggested by the Lexical mapper. The results for each dataset (G-R, R-G, Z-F, F-Z) are compared by precision and recall. The "Most popular" heuristics is again selected because it is one of the methods that gets the best results in our evaluation in Section 6.7.3.2.

Figure 6.9 Influence of the set of mappings on Recall and Precision of the Most popular selection heuristics for datasets G-R and R-G.



Result. Table 6.16 presents the result of this comparison. In general, the semantical mappers get mappings with high precision and higher recall (even twice higher for the R-G dataset) than the Lexical mapper.

Table 6.16 Selection of mappings by the Most popular mappings heuristics on the set of semantical explicit mappings (*Sem. Explicit*) and lexical mappings (*Lexical*): Precision and Recall.

Dataset Mappings	**Eq.**		**Eq. & Gen.**		**Eq. & Non-eq.**	
	P	**R**	**P**	**R**	**P**	**R**
G-R Sem. Explicit	0.68	0.62	0.78	0.39	0.98	0.46
G-R Lexical	0.96	0.56	0.96	0.30	0.96	0.29
R-G Sem. Explicit	0.43	0.61	0.86	0.31	0.96	0.29
R-G Lexical	0.74	0.45	0.97	0.15	0.97	0.13
Z-F Sem. Explicit	0.75	0.92	1.00	0.58	1.00	0.58
Z-F Lexical	0.88	0.85	1.00	0.45	1.00	0.45
F-Z Sem. Explicit	0.73	0.92	0.94	0.94	1.00	0.97
F-Z Lexical	1.00	0.85	1.00	0.67	1.00	0.65

6.8 Conclusions

With respect to the last research question in the Introduction Chapter, we present a novel automatic method to establish semantical mappings between attribute values from different domains in heterogeneous data collections that can be sorted by means of an order relation that reflects a notion of generality. This method applies an extensible set of novel mappers that are based on the dynamically constructed textual descriptions of considered values and employs information retrieval techniques for further processing. Moreover, we have also shown how a known partial order relation defined on the domain of considered attributes can be used to create and select proper mappings.

Our approach generates a set of mappings where each value from the source dataset is related to at least one value from the target dataset. In general, it produces alternative mappings for the same value. However, it may be crucial to select exactly one value from the target for each value from the source. Thus, a more sophisticated selection method than the method that is based on the lowest uncertainty about mapping has to be investigated. For this purpose we proposed novel selection heuristics which are based on the concept of majority (the frequency of mappings) and the balanced selection (the most specific category from the candidate categories provided that the selected category is comparable to all other candidate categories) over the candidate mappings set. In addition, we also proposed a novel algorithm that tries to find a mapping specific for the particular object based on onomastic information. Our method can be easily applied for data integration or interoperability tasks. The established semantical mappings help to maintain consistency, decrease the number of coreferent tuples in the integrated dataset and allow to apply fusion functions (e.g., those presented in the previous chapter), which has an extreme influence on data quality. This in turn descreases the cost of database maintenance.

7
Overall conclusions

The dissertation dealt with the following research questions which were raised in the Introduction Chapter.

Question 1: How does one establish schema matching based only on the schema itself?

The novel XML schema matching algorithm which is based on schema information only is proposed in Chapter 2 as a response to this research question. Our technique is able to establish one-to-one matching of corresponding XML schema elements of heterogeneous data sources comparing element names (steps) and their sequences (paths) in an automatic fashion; thus it is a syntactical (lexical) schema matching. This, in turn, may help to further decide on the coreference of XML schemas. We treat coreference as a binary notion, i.e., two steps, paths or schemas are either coreferent or not. However, we assume that the results of coreference detection may be uncertain, which is represented by employing possibilistic truth values (PTVs).

More specifically, we consider here three levels on which coreference is detected. On the basic level, i.e. step level, the steps of any two paths are pairwise compared to determine their coreference. All steps pairs of the two paths are associated with PTVs, which are, in turn, aggregated to a single PTV which represents coreference on the path level. However, not all steps in a path are equally important with respect to coreference detection, thus, step weights are defined by the novel heuristic and are used by the aggregation operator. The steps weights depend on

the position of the steps in their respective paths and are set according to the rule that two paths are more likely to be coreferent if they have more similar steps at the end. Finally, on the schema level, the information on the paths' coreference, as it is also represented by PTVs, is aggregated to determine the coreference of the two schemas. However, schema elements are not equally important when schemas similarity (coreference/matching) is considered. Thus, a novel heuristics to derive elements importance weights is proposed and used by the aggregation operator. These heuristics are based on expert opinions that the most important element is unique (occurs in the schema only once) and mandatory (occurs in the schema at least once), and is also located closer to the root.

Question 2: How does one establish schema matching based on content data?

The response to this research question is the novel schema matching algorithm which is described in Chapter 3. In contrast to the above approach, this method is based on content data only but also assumes that the results of coreference detection may be uncertain, which is represented by employing possibilistic truth values and fuzzy integers. More specifically, our novel approach is composed of three phases. In the first phase the vertical schema matchings are established by a statistical and lexical comparison of attribute domains, and the matchings are a basis for the second phase. Namely, the vertical matchings help to detect coreferent tuples efficiently across heterogeneous data sources, and, in turn, the coreferent tuples are used to derive the horizontal schema matchings. Finally, ambiguous matchings are resolved in the third phase. Conflict resolution is based on the uncertainty of matching and on the number of matched attributes, and also on the idea that a matching which is repeated by many coreferent tuples is more certain than a matching which is not frequent.

Contrary to the schema matching based on schema information only, these techniques allow to establish a semantical matching between corresponding attributes (schema elements in general), and to cope with attribute granularity and data coverage problems. Namely, content data supply additional information about the considered attributes, which helps to reveal their real meaning. In turn, a comparison of attribute values helps to detect coreferent attributes with different granularity (one-to-many matchings) across heterogeneous data sources, i.e. when the same piece of information is represented by a different number of attributes in the considered schemas; whereas the data coverage problem is identified by a comparison of the number of coreferent terms in the particular attribute domains. However, the content data itself can also be insufficient because the same information can be represented in different ways, e.g. there can be synonyms. Thus, some knowledge base may be necessary to detect coreference and, thus, the construction of

a knowledge base is an important aspect in improving the detection of coreferent objects and is considered in the next research question.

Question 3: How can a dynamically constructed knowledge base improve the detection of coreferent tuples and data fusion in homogeneous or heterogeneous data collections?

The novel techniques which are proposed in Chapters 4 and 5 are a response to this research question. Namely, in Chapter 4 the novel Dynamical Order Construction (DOC) algorithm is introduced and its impact on the detection of coreferent tuples is investigated. More specifically, DOC constructs, using coreferent tuples in homogeneous or heterogeneous data collections, a partial order relation over the domain of an attribute, whose values may need a semantical comparison and can be sorted by means of an order relation that reflects a notion of generality. In Chapter 4 three novel heuristics are proposed to measure the generality of the considered values. These heuristics depend on frequency (number of occurrences of each value) and uniqueness (distinction of values in coreferent tuples or in the considered dataset), and are evaluated in-depth on real-life datasets. Moreover, our knowledge base construction has the advantage that it automatically adapts to the values present in the dataset and that predefined taxonomical knowledge on the attribute is not necessary.

DOC in the context of data fusion is studied in Chapter 5. Namely, the novel approach to combine coreferent values of the considered attribute of an entity into a single value is proposed. This technique is based on the new strategy for selection, called *balanced* selection, which adopts the sort-and-select principle. First, values are sorted by using the generality relation and next the most specific value that is comparable to all others is selected as a result of the fusion function. The advantages and disadvantages of our methods are experimentally investigated on large real-life datasets.

Question 4: How does one establish semantical mappings between the values of categorical attributes and what impact does a partial order relation have on the data fusion of heterogeneous data sources?

The novel techniques are proposed in Chapter 6 as a response to the last research question. In contrast to the DOC algorithm, these methods are not based on coreferent tuples but on textual descriptions, which are compared using information retrieval techniques to establish semantical mappings between the values of categorical attributes from different domains that can be sorted by means of an order relation that reflects a notion of generality. The textual description is constructed

from the values of other attributes and from a definition of the particular value which is extracted from the World Wide Web. Thus, these mappings are called *explicit* mappings and are derived by our novel *explicit* mappers. Moreover, we also proposed novel *implicit* mappers which create additional mappings using explicit mapping and known partial order relations defined on the domain of the considered attributes. Explicit and implicit mappings are called *default* mappings, because they do not depend on the particular object. Besides this we also proposed a novel algorithm which derives mappings specific for a particular object (tuple) using onomastic information and partial order relations to guarantee high precision of the mappings.

Our approach produces a set of one-to-many mappings between the coreferent values of categorical attributes from different domains, i.e. any value from one source can be mapped to more than one value in another source. However, it may be crucial to reduce the set of one-to-many mappings to a set of one-to-one mappings. In this case the uncertainty-based selection method can be insufficient. Thus, in Chapter 6 some novel heuristics are proposed which select proper one-to-one mappings. These are based on the concept of majority and supremum over the candidate mappings set, and also on the balanced selection function defined in Chapter 5 with partial order relations.

The novel techniques proposed here are evaluated in-depth on real-life datasets and have the advantages that help to maintain consistency, detect coreferent tuples in databases and allow to apply the fusion function, e.g., the balanced selection function which is presented in Chapter 5. As a consequence, this helps to increase data quality and to decrease the cost of database maintenance.

References

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

[2] A. Elmagarmid, P. Ipeirotis, and V. Verykios. *Duplicate Record Detection: A Survey*. IEEE Transactions on Knowledge and Data engineering, 19(1):1–16, 2007.

[3] I. Fellegi and A. Sunter. *A Theory for Record Linkage*. American Statistical Association Journal, 64(328):1183–1210, 1969.

[4] M. Jaro. *Unimatch: A Record Linkage System: User's Manual*. Technical report, US Bureau of the Census, 1976.

[5] W. Winkler and Y. Thibaudeau. *An Application of the Fellegi-Sunter Model of Record Linkage to the 1990 US Decennial Census*. Technical Report RR91/09, Statistical Research Report Series, 1991.

[6] W. Winkler. *Improved Decision Rules in the Fellegi-Sunter Model of Record Linkage*. Technical Report RR93/12, Statistical Research Report Series, 1993.

[7] W. Winkler. *Methods for Record Linkage and Bayesian Networks*. Technical Report RRS2002/05, Statistical Research Report Series, 2002.

[8] A. Bronselaer and G. De Tré. *A possibilistic approach on string comparison*. IEEE Transactions on Fuzzy Systems, 17(1):208–223, 2009.

[9] S. Tejada, C. Knoblock, and S. Minton. *Learning object identification rules for information integration*. Information Systems, 26(8):607–633, 2001.

[10] R. G. G. Cattell, editor. *The Object Database Standard: ODMG 2.0*. Morgan Kaufmann, 1997.

[11] A. Bronselaer and G. De Tré. *Semantical evaluators*. In Proceedings of the joint 2009 International Fuzzy Systems Association world congress and 2009 European Society for Fuzzy Logic and Technology conference, pages 663–668. European Society for Fuzzy Logic and Technology (EUSFLAT), 2009.

[12] V. Levenstein. *Binary Codes Capable of Correcting Deletions, Insertions and Reversals*. Physics Doklady, 10(8):707–710, 1966.

[13] F. Naumann, A. Bilke, J. Bleiholder, and M. Weis. *Data Fusion in Three Steps: Resolving Inconsistencies at Schema-, Tuple-, and Value-level*. In Bulletin of the Technical Committee on Data Engineering, pages 21–31, 2006.

[14] X. L. Dong, L. Berti-Equille, and D. Srivastava. *Truth discovery and copying detection in a dynamic world*. volume 2, pages 562–573, 2009.

[15] M. Szymczak, S. Zadrożny, and G. De Tré. *Coreference detection in XML metadata*. In W. Pedrycz and M. Reformat, editors, Proceedings of 2013 Joint IFSA World Congress NAFIPS Annual Meeting, pages 1354–1359, 2013.

[16] M. Szymczak, A. Bronselaer, S. Zadrożny, and G. De Tré. *Semantical mappings of attribute values for data integration*. In Proceedings of NAFIPS 2014, pages 1–8. IEEE, 2014.

[17] J. Madhavan, P. A. Bernstein, and E. Rahm. *Generic Schema Matching with Cupid*. In Proceedings of the 27th International Conference on Very Large Data Bases, VLDB '01, pages 49–58, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc.

[18] H.-h. Do and E. Rahm. *COMA - A system for flexible combination of Schema Matching Approaches*. In Proceedings of the VLDB 2002, pages 610–621, 2002.

[19] A. Bilke and F. Naumann. *Schema matching using duplicates*. In Proceedings of the 28th International Conference on Data Engineering (ICDE), 2005.

[20] R. Dhamankar, Y. Lee, A. Doan, A. Halevy, and P. Domingos. *iMAP: discovering complex semantic matches between database schemas*. In Proceedings of the 2004 ACM SIGMOD International Conference on Management of Data, ACM. Press, 2004.

[21] M. Szymczak and J. Koepke. *Matching Methods for Semantic Annotation-based XML Document Transformations*. In K. Atanassov, et al. (Eds.), New Developments in Fuzzy Sets, Intuitionistic Fuzzy Sets, Generalized Nets and Related Topics. Applications. Volume II, pages 297–308. SRI PAS, 2012.

[22] R. Ananthakrishna, S. Chaudhuri, and V. Ganti. *Eliminating Fuzzy Duplicates in Data Warehouses*. In Proceedings of the 28th International Conference on Very Large Databases (VLDB 2002), 2002.

[23] A. Doan, Y. Lu, Y. Lee, and J. Han. *Object Matching for Information Integration: A Profiler-Based Approach*. In Proceedings of the IJCAI-03 Workshop on Information Integration on the Web. 2003, pages 53–58, 2003.

[24] J. Kang, D. Lee, and P. Mitra. *Identifying Value Mappings for Data Integration: An Unsupervised Approach.* In A. H. H. Ngu, M. Kitsuregawa, E. J. Neuhold, J.-Y. Chung, and Q. Z. Sheng, editors, Proceedings of WISE, volume 3806 of *Lecture Notes in Computer Science*, pages 544–551. Springer, 2005.

[25] M. Craven, D. DiPasquo, D. Freitag, A. McCallum, T. Mitchell, K. Nigam, and S. Slattery. *Learning to construct knowledge bases from the World Wide Web*. Artificial Intelligence, 118:69 – 113, 2000.

[26] W. W. Cohen. *Integration of Heterogeneous Databases Without Common Domains Using Queries Based on Textual Similarity*. In Proceedings of the 1998 ACM SIGMOD International Conference on Management of Data, pages 201–212, 1998.

[27] S. Tejada, C. A. Knoblock, and S. Minton. *Learning domain-independent string transformation weights for high accuracy object identification*. In Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining, KDD '02, pages 350–359, New York, NY, USA, 2002. ACM.

[28] H. Lu, W. Fan, C. H. Goh, S. Madnick, and D. Cheung. *Discovering and reconciling semantic conflicts: a data mining prospective*. In Proceedings of IFIP Working Conference on Data Semantics (DS-7), 1997.

[29] N. Press. *Understanding Metadata*. National Information Standards Organization Press, 2004.

[30] L. A. Zadeh. *Fuzzy Sets*. Information and Control, 8:338–353, 1965.

[31] L. Zadeh. *Fuzzy sets as a basis for a theory of possibility*. Fuzzy Sets Syst., 100:9–34, April 1999.

[32] R. Yager. *On the theory of bags*. International Journal of General Systems, 13(1):23–27, 1986.

[33] E. F. Codd. *A Relational Model of Data for Large Shared Data Banks*. Communications of the ACM, 13(6):377–387, 1970.

[34] D. C. Fallside. *XML Schema Part 0: Primer*. W3C, 2001.

[35] H. Prade. *Possibility sets, fuzzy sets and their relation to Lukasiewicz logic*. In Proceeding of 12th Int Symp on Multiple-Valued Logic, pages 223–227, 1982.

[36] T. Calvo, G. Mayor, and R. Mesiar, editors. *Aggregation Operators: New Trends and Applications*. Physica-Verlag GmbH, Heidelberg, Germany, Germany, 2002.

[37] A. Bronselaer and G. D. Tré. *Properties of Possibilistic String Comparison*. IEEE Transactions on Fuzzy Systems, 18(2):312–325, April 2010.

[38] M. Sugeno. *Theory of Fuzzy Integrals and its Applications*. PhD thesis, Tokyo, Japan, 1974.

[39] G. de Cooman. *Towards a Possibilistic Logic*. In D. Ruan, editor, Fuzzy Set Theory and Advanced Mathematical Applications, volume 4 of *International Series in Intelligent Technologies*, pages 89–133. Springer US, 1995.

[40] A. Hallez, G. De Tré, J. Verstraete, and T. Matthé. *Application of fuzzy quantifiers on possibilistic truth values*. In Proceedings of EUROFUSE EURO WG on Fuzzy Sets, pages 252–254. EXIT, 2004.

[41] E. Herrera-Viedma, E. Peis, J. M. Morales-del Castillo, S. Alonso, and K. Anaya. *A fuzzy linguistic model to evaluate the quality of Web sites that store XML documents*. International Journal of Approximate Reasoning, 46(1):226 – 253, 2007.

[42] E. Herrera-Viedma, G. Pasi, A. G. Lopez-Herrera, and C. Porcel. *Evaluating the information quality of Web sites: A methodology based on fuzzy computing with words*. Journal of the American Society for Information Science and Technology, 57(4):538–549, 2006.

[43] Z. Bellahsene, A. Bonifati, and E. Rahm, editors. *Schema Matching and Mapping*. Springer, 2011.

[44] A. Das Sarma, L. Fang, N. Gupta, A. Y. Halevy, H. Lee, F. Wu, R. Xin, and C. Yu. *Finding related tables*. In SIGMOD Conference, pages 817–828, 2012.

[45] R. Nayak and T. Tran. *A Progressive Clustering Algorithm to Group the XML Data by Structural and Semantic Similarity.* IJPRAI, 21(4):723–743, 2007.

[46] R. Nayak and W. Iryadi. *XML schema clustering with semantic and hierarchical similarity measures.* Knowledge-Based Systems, 20(4):336–349, 2007.

[47] A. Algergawy, R. Nayak, and G. Saake. *Element similarity measures in XML schema matching.* Inf. Sci., 180(24):4975–4998, 2010.

[48] R. Fagin, L. M. Haas, M. A. Hernández, R. J. Miller, L. Popa, and Y. Velegrakis. *Clio: Schema Mapping Creation and Data Exchange*. In Conceptual Modeling: Foundations and Applications, pages 198–236, 2009.

[49] B. S. Lerner. *A Model for Compound Type Changes Encountered in Schema Evolution*. ACM Transactions on Database Systems, 25(1):83–127, March 2000.

[50] L. Palopoli, D. Sacca, and D. Ursino. *Semi-Automatic Semantic Discovery of Properties from Database Schemas*. In Proceedings of IDEAS, pages 244–253, 1998.

[51] L. Palopoli, D. Saccá, and D. Ursino. *An automatic technique for detecting type conflicts in database schemes*. In Proceedings of the seventh international conference on Information and knowledge management, CIKM '98, pages 306–313, New York, NY, USA, 1998. ACM.

[52] L. Palopoli, D. Saccá, G. Terracina, and D. Ursino. *A Unified Graph-Based Framework for Deriving Nominal Interscheme Properties, Type Conflicts and Object Cluster Similarities*. In Proceedings of the Fourth IECIS International Conference on Cooperative Information Systems, COOPIS '99, pages 34–45, Washington, DC, USA, 1999. IEEE Computer Society.

[53] S. Bergamaschi, S. Castano, and M. Vincini. *Semantic Integration of Semistructured and Structured Data Sources*. SIGMOD Record, 28:54–59, 1999.

[54] A. Doan, P. Domingos, and A. Y. Levy. *Learning Source Description for Data Integration*. In WebDB (Informal Proceedings), pages 81–86, 2000.

[55] S. Castano, V. De Antonellis, and S. De Capitani di Vimercati. *Global Viewing of Heterogeneous Data Sources*. IEEE Transactions on Knowledge and Data Engineering, 13(2):277–297, 2001.

[56] J. Lu, J. Wang, and S. Wang. *XML Schema Matching*. International Journal of Software Engineering and Knowledge Engineering, 17(5):575–597, 2007.

[57] G. A. Miller. *WordNet: A Lexical Database for English*. Communication ACM, 38(11):39–41, November 1995.

[58] D. Milne and I. H. Witten. *An Effective, Low-Cost Measure of Semantic Relatedness Obtained from Wikipedia Links*. In Proceedings of AAAI 2008, 2008.

[59] T. K. Landauer, P. W. Foltz, and D. Laham. *An Introduction to Latent Semantic Analysis*. Discourse Processes, (25):259–284, 1998.

[60] C. De Maio, G. Fenza, M. Gaeta, V. Loia, F. Orciuoli, and S. Senatore. *RSS-based e-learning recommendations exploiting fuzzy FCA for Knowledge Modeling*. Applied Soft Computing, 12(1):113 – 124, 2012.

[61] V. Loia, G. Fenza, C. De Maio, and S. Salerno. *Hybrid methodologies to foster ontology-based knowledge management platform*. In Proceedings of 2013 IEEE Symposium on Intelligent Agent (IA), pages 36–43, April 2013.

[62] P. Mitra, G. Wiederhold, and J. Jannink. *Semi-automatic Integration of Knowledge Sources*. In Proceedings of the 2nd International Conference on Information Fusion, 1999.

[63] P. Mitra, G. Wiederhold, and M. L. Kersten. *A Graph-Oriented Model for Articulation of Ontology Interdependencies*. In C. Zaniolo, P. C. Lockemann, M. H. Scholl, and T. Grust, editors, Proceedings of Advances in Database Technology - EDBT 2000, 7th International Conference on Extending Database Technology, Konstanz, Germany, March 27-31, 2000, volume 1777 of *Lecture Notes in Computer Science*, pages 86–100. Springer, 2000.

[64] D. Beneventano, S. Bergamaschi, F. Guerra, and M. Vincini. *The MOMIS Approach to Information Integration*. In Proceedings of ICEIS (2), pages 194–198, 2001.

[65] P. T. T. Thuy, Y.-K. Lee, and S. Lee. *S-Trans: Semantic transformation of XML healthcare data into OWL ontology.* Knowledge-Based Systems, 35:349–356, 2012.

[66] T. Milo and S. Zohar. *Using Schema Matching to Simplify Heterogeneous Data Translation*. In Proceedings of 24th Int. Conf. Very Large Data Bases, VLDB, pages 122–133, 1998.

[67] S. Melnik, H. Garcia-Molina, and E. Rahm. *Similarity Flooding: A Versatile Graph Matching Algorithm and Its Application to Schema Matching*. In Proceedings of the 18th International Conference on Data Engineering,

ICDE '02, pages 117–128, Washington, DC, USA, 2002. IEEE Computer Society.

[68] K. Zhang and D. Shasha. *Simple fast algorithms for the editing distance between trees and related problems*. SIAM J. Comput., 18(6):1245–1262, December 1989.

[69] K. Zhang, D. Shasha, and J. T.-L. Wang. *Fast Serial and Parallel Algorithms for Approximate Tree Matching with VLDC's*. In Proceedings of the Third Annual Symposium on Combinatorial Pattern Matching, CPM '92, pages 151–161, London, UK, UK, 1992. Springer-Verlag.

[70] D. Shasha and K. Zhang. *Approximate Tree Pattern Matching*. In Pattern Matching Algorithms, pages 341–371. Oxford University Press, 1997.

[71] J. Tsong-li, W. K. Zhang, K. Jeong, and D. Shasha. *A System for Approximate Tree Matching*. IEEE Transactions on Knowledge and Data Engineering, 6:559–571, 1992.

[72] J. Liu, Z. M. Ma, and L. Yan. *Efficient processing of twig pattern matching in fuzzy XML*. In Proceedings of the 18th ACM conference on Information and knowledge management, CIKM '09, pages 117–126, New York, NY, USA, 2009. ACM.

[73] D.-B. Dao and J. Cao. *A Glance on Current XML Twig Pattern Matching Algorithms*. In O. Gervasi, B. Murgante, A. Laganà, D. Taniar, Y. Mun, and M. Gavrilova, editors, Computational Science and Its Applications – ICCSA 2008, volume 5073 of *Lecture Notes in Computer Science*, pages 307–321. Springer Berlin Heidelberg, 2008.

[74] C. Clifton, E. Housman, and A. Rosenthal. *Experience with a Combined Approach to Attribute-Matching Across Heterogeneous Databases*. In Proceedings of the IFIP Working Conference on Data Semantics DS-7, 1997.

[75] A. Rajesh and S. Srivatsa. *XML Schema Matching - Using Structural Information*. International Journal of Computer Applications, 8(2):34–41, 2010.

[76] N. Tansalarak and K. T. Claypool. *QMatch - Using paths to match XML schemas.* Data Knowledge Engineering, 60(2):260–282, 2007.

[77] J. Farrell and H. Lausen. *Semantic Annotations for WSDL and XML Schema*. W3C working draft, World Wide Web Consortium, 2007.

[78] J. Köpke and J. Eder. *Semantic annotation of XML-schema for document transformations*. In Proceedings of the 2010 international conference on On the move to meaningful internet systems, OTM'10, pages 219–228, Berlin, Heidelberg, 2010. Springer-Verlag.

[79] A. Bronselaer, A. Hallez, and G. De Tré. *Evaluation in the possibilistic framework for object matching*. In Proceedings of IPMU 2008, 2008.

[80] P. Calado, M. Herschel, and L. Leitão. *An Overview of XML Duplicate Detection Algorithms*. In Soft Computing in XML Data Managment, pages 193–224. 2010.

[81] S. Zadrożny, J. Kacprzyk, and G. Sobota. *Avoiding duplicate records in a database using a linguistic quantifier based aggregation - A practical approach*. In Proceedings of FUZZ-IEEE, pages 2194–2201, 2008.

[82] J. R. Munkres. *Algorithms for the Assignment and Transportation Problems*. Journal of the Society for Industrial and Applied Mathematics, 5(1):32–38, March 1957.

[83] P. D. Zadeh and M. Z. Reformat. *Fuzzy semantic similarity in linked data using the OWA operator*. In Proceeding of NAFIPS 2012, pages 1–6, Aug. 2012.

[84] H.-H. Do and E. Rahm. *Matching large schemas: Approaches and evaluation*. Inf. Syst., 32(6):857–885, September 2007.

[85] C. J. V. Rijsbergen. *Information Retrieval*. Butterworth-Heinemann, Newton, MA, USA, 2nd edition, 1979.

[86] M. Szymczak, S. Zadrożny, A. Bronselaer, and G. De Tré. *Coreference detection in an XML schema*. Information Sciences, 296:237–262, 2015.

[87] O. A. Mehdi, H. Ibrahim, and L. S. Affendey. *Instance based Matching using Regular Expression.* Procedia CS, 10:688–695, 2012.

[88] M. Perkowitz, R. B. Doorenbos, O. Etzioni, and D. S. Weld. *Learning to Understand Information on the Internet: An Example-Based Approach*. Journal of Intelligent Information Systems, 8(2):133–153, March 1997.

[89] C. E. H. Chua, R. H. L. Chiang, and E.-P. Lim. *Instance-based attribute identification in database integration*. The VLDB Journal, 12(3):228–243, October 2003.

[90] R.-D. Reiss and M. Thomas. *Statistical Analysis of Extreme Values: with Applications to Insurance, Finance, Hydrology and Other Fields*. Birkhäuser Basel, 3rd ed. edition, 2007.

[91] R. J. A. Little and D. B. Rubin. *Statistical Analysis with Missing Data*. John Wiley & Sons, Inc., New York, NY, USA, 1986.

[92] A. K. Jain, R. P. W. Duin, and J. Mao. *Statistical Pattern Recognition: A Review*. IEEE Trans. Pattern Anal. Mach. Intell., 22(1):4–37, January 2000.

[93] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer Series in Statistics. Springer New York Inc., New York, NY, USA, 2001.

[94] A. Bronselaer, A. Hallez, and G. De Tré. *Extensions of fuzzy measures and the Sugeno integral for possibilistic truth values*. International Journal of Intelligent Systems, 24(2):97–117, 2009.

[95] S.-S. Weng, H.-J. Tsai, S.-C. Liu, and C.-H. Hsu. *Ontology construction for information classification*. Expert Systems with Applications, 31:1–12, 2006.

[96] C.-S. Lee, Y.-F. Kao, Y.-H. Kuo, and M.-H. Want. *Automated ontology construction for unstructured text documents*. Data and Knowledge Engineering, 60:547–566, 2007.

[97] M. Y. Dahab, H. Hassan, and A. Rafea. *TextOntoEx: Automatic ontology construction from natural English text*. Expert Systems with Applications, 34:1474–1480, 2008.

[98] X. Liu, Y. Song, S. Liu, and H. Wang. *Automatic Taxonomy Construction from Keywords*. In KDD 2012, 2012.

[99] S. Warshall. *A theorem on Boolean matrices*. Journal of the ACM, 9(1):11–12, 1962.

[100] R. Floyd. *Algorithm 97: Shortest path*. Communications of the ACM, 5(6):345, 1962.

[101] H. S. Warren. *A modification of Warshall's algorithm for the transitive closure of binary relations*. Communications of the ACM, 18(4):218–220, 1975.

[102] E. Nuutila. *Efficient Transitive Closure Computation in Large Digraphs*. PhD thesis, Acta Polytechnica Scandinavica, Mathematics and Computing in Engineering, 1995.

[103] A. Bronselaer. *Coreference of atomic and complex objects*. PhD thesis, Ghent University, 2010.

[104] A. Tversky. *Features of similarities*. Psychological Review, 84:327–352, 1977.

[105] M. Friedman. *The use of ranks to avoid the assumption of normality implicit in the analysis of variance*. Journal of the American Statistical Association, 32:675–701, 1937.

[106] M. Kendall and B. B. Smith. *The Problem of m Rankings*. The Annals of Mathematical Statistics, 10(3):275–287, 1939.

[107] F. Wilcoxon. *Individual comparisons by ranking methods*. Biometrics Bulletin, 1(6):80–83, 1945.

[108] C. Batini and M. Scannapieca. *Data quality: concepts, methodologies and techniques*. Springer-Verlag, 2006.

[109] J. Bleiholder and F. Naumann. *Declarative data fusion: Syntax, semantics, and implementation*. In Proceedings of the East European Conference on Advances in Databases and Information Systems (ADBIS), pages 58–73, 2005.

[110] C. Galindo-Legaria. *Outerjoins as disjunctions*. In Proceedings of SIGMOD, pages 348–358, 1994.

[111] César Galindo-Legaria and Arnon Rosenthal. *Outerjoin simplification and reordering for query optimization*. Transactions on Database Systems, 22(1):43–74, 1997.

[112] L. L. Yan and T. Özsu. *Conflict Tolerant Queries in AURORA*. In Proceedings of the Fourth IECIS International Conference on Cooperative Information Systems, pages 279–290, 1999.

[113] S. Greco, L. Pontieri, and E. Zumpano. *Integrating and managing conflicting data*. In The 4th International Andrei Ershov Memorial Conference on Perspectives of System Informatics, pages 187–192, 2001.

[114] J. Bleiholder, S. Szott, M. Herschel, F. Kaufer, and F. Naumann. *Subsumption and complementation as data fusion operators*. In EDBT, pages 513–524, 2010.

[115] J. Bleiholder. *Data Fusion and Conflict Resolution in Integrated Information Systems*. PhD thesis, Universität Potsdam, 2010.

[116] J. Bleiholder, M. Herschel, and F. Naumann. *Eliminating NULLs with Subsumption and Complementation*. IEEE Data Engineering Bulletin, 34(3):18–25, 2011.

[117] A. Bilke, J. Bleiholder, C. Böhm, K. Draba, F. Naumann, and M. Weis. *Automatic Data Fusion with HumMer*. In Proceedings of VLDB, pages 1251–1254, 2005.

[118] A. Motro and P. Anokhin. *Fusionplex: resolution of data inconsistencies in the integration of heterogeneous information sources*. Information Fusion, 7(2):176–196, 2006.

[119] Y. Papakonstantinou, S. Abiteboul, and H. Garcia-Molinam. *Object fusion in mediator systems*. In Proceedings of the International Conference on Very Large Databases (VLDB), pages 413–424, 1996.

[120] C. Naiman and A. Ouksel. *A Classification of Semantic Conflicts in Heterogeneous Database Systems*. Journal of Organizational Computing, 5(2):167–193, 1995.

[121] W. Fan, H. Lu, S. Madnick, and D. Cheung. *Discovering and reconciling value conflicts for numerical data integration*. Information Systems, 26:635–656, 2001.

[122] S. Ram and J. Park. *Semantic Conflict Resolution Ontology (SCROL): An Ontology for Detecting and Resolving Data and Schema-Level Semantic Conflicts*. IEEE Transactions on Knowledge and Data Engineering, 16(2):189–202, 2004.

[123] Q. Liu, T. Huang, S.-H. Liu, and H. Zhong. *An Ontology-based Approach for Semantic Conflict Resolution in Database Integration*. Journal of Computer Science and Technology, 22(2):218–227, 2007.

[124] A. Hajmoosaei and S. Abdul-Kareem. *An ontology-based approach for resolving semantic schema conflicts in the extraction and integration of query-based information from heterogeneous web data sources*. In Proceedings of the Third Australasian Workshop on Advances in Ontologies, pages 35–43, 2007.

[125] X. L. Dong and F. Naumann. *Data fusion: resolving data conflicts for integration*. volume 2, pages 1654–1655, 2009.

[126] L. Berti-Equille, A. Das Sarma, X. L. Dong, A. Marian, and D. Srivastava. *Sailing the information ocean with awareness of currents: discovery and application of source dependence*. In CIDR, 2009.

[127] H. Wache, T. Vögele, U. Visser, H. Stuckenschmidt, G. Schuster, H. Neumann, and S. Hübner. *Ontology-based Integration of Information - A survey of existing approaches*. In Proceedings of IJCAI-01 Workshop: Ontologies and Information Sharing, pages 108–117, 2001.

[128] A. Bronselaer, D. Van Britsom, and G. De Tré. *A framework for multiset merging*. Fuzzy Sets and Systems, 191:1–20, 2012.

[129] A. Motro, P. Anokhin, and A. Acar. *Utility-based resolution of data inconsistencies*. In Proceedings of the 2004 international workshop on Information quality in informational systems, pages 34–43, 2004.

[130] J. Lin and A. Mendelzon. *Knowledge Base Merging by Majority*. In Dynamic Worlds: From the Frame Problem to Knowledge Management. Kluwer, 1994.

[131] S. S. Stevens. *On the Theory of Scales of Measurement*. Science, 103(2684):677–680, 1946.

[132] R. Yager and A. Rybalov. *Noncommutative self-identity aggregation*. Fuzzy Sets and Systems, 85(1):73–82, 1997.

[133] J. Bleiholder and F. Naumann. *Data fusion*. ACM Computing Surveys, 41(1), 2008.

[134] F. Suchanek, G. Kasneci, and G. Weikum. *Yago - A Core of Semantic Knowledge*. In Proceedings of the 16th International World Wide Web conference (WWW 2007), 2007.

[135] S. Chaudhuri and L. Gravano. *Evaluating top-k selection queries*. In Proceedings of the 25th International Conference on Very Large Databases (VLDB), 1999.

[136] N. Bruno, S. Chaudhuri, and L. Gravano. *Top-k Selection Queries over Relational Databases: Mapping Strategies and Performance Evaluation*. ACM Transactions on Database Systems, 27(2):153–187, 2002.

[137] K. Arrow. *Social choice and individual values*. Wiley, New York, 1963.

[138] O. Dunn. *Multiple Comparisons Among Means*. Journal of the American Statistical Association, 56(293):52–64, 1961.

[139] M. Weis and F. Naumann. *Detecting Duplicate Objects in XML Documents.* In F. Naumann and M. Scannapieco, editors, Proceedings of IQIS, pages 10–19. ACM, 2004.

[140] M. Weis and F. Naumann. *DogmatiX tracks down duplicates in XML.* In Proceedings of the 2005 ACM SIGMOD international conference on Management of data, pages 431–442, New York, NY, USA, 2005. ACM.

[141] M. Jarmasz and S. Szpakowicz. *S.: Rogets thesaurus and semantic similarity*. In Proceedings of the RANLP-2003, pages 212–219, 2003.

[142] M. Strube and S. P. Ponzetto. *WikiRelate! Computing Semantic Relatedness Using Wikipedia.* In Proceedings of AAAI. AAAI Press, 2006.

[143] C. Leacock and M. Chodorow. *Combining local context and WordNet similarity for word sense identification*. In C. Fellfaum, editor, MIT Press, pages 265–283, Cambridge, Massachusetts, 1998.

[144] E. Gabrilovich and S. Markovitch. *Computing Semantic Relatedness Using Wikipedia-based Explicit Semantic Analysis*. In Proceedings of the 20th International Joint Conference on Artifical Intelligence, IJCAI'07, pages 1606–1611, San Francisco, CA, USA, 2007. Morgan Kaufmann Publishers Inc.

[145] S.-J. Choi, H.-J. Song, S.-B. Park, and S.-J. Lee. *A POI Categorization by Composition of Onomastic and Contextual Information*. In Proceedings of IEEE/WIC/ACM International Conference on Web Intelligence (WI). IEEE, 2014.

[146] P. Shvaiko and J. Euzenat. *Ontology Matching: State of the Art and Future Challenges*. IEEE Transactions on Knowledge and Data Engineering, 25(1):158–176, 2013.

[147] J. Euzenat and P. Valtchev. *An integrative proximity measure for ontology alignment*. In A. Doan, A. Halevy, and N. Noy, editors, Proceedings of the 1st Intl. Workshop on Semantic Integration, volume 82 of *CEUR*, 2003.

[148] X. Su, S. Hakkarainen, and T. Brasethvik. *Semantic Enrichment for Improving Systems Interoperability*. In Proceedings of the 2004 ACM Symposium on Applied Computing, SAC '04, pages 1634–1641, New York, NY, USA, 2004. ACM.

[149] P. Lambrix and H. Tan. *SAMBO-A System for Aligning and Merging Biomedical Ontologies*. Web Semant., 4(3):196–206, September 2006.

[150] P. Lambrix, H. Tan, and Q. L. 0002. *SAMBO and SAMBOdtf Results for the Ontology Alignment Evaluation Initiative 2008.* In P. Shvaiko, J. Euzenat, F. Giunchiglia, and H. Stuckenschmidt, editors, OM, volume 431 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2008.

[151] Y. R. Jean-Mary and M. R. Kabuka. *ASMOV: Results for OAEI 2008*. In Proceedings of the Third International Workshop on Ontology Matching, 2008.

[152] Y. R. Jean-Mary, E. P. Shironoshita, and M. R. Kabuka. *Ontology Matching with Semantic Verification*. Web Semant., 7(3):235–251, September 2009.

[153] G. Pirró and D. Talia. *UFOme: An ontology mapping system with strategy prediction capabilities*. Data & Knowledge Engineering, 69(5):444 – 471, 2010.

[154] A. Gross, M. Hartung, T. Kirsten, and E. Rahm. *Mapping Composition for Matching Large Life Science Ontologies*. In Proceedings of ICBO, pages 109–116, 2011.

[155] V. V. Cross, P. Silwal, and X. Chen. *Semantic similarity measures in ontology alignment*. In Proceedings of IFSA/NAFIPS, pages 442–447, 2013.

[156] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten. *The WEKA Data Mining Software: An Update*. SIGKDD Explor. Newsl., 11(1):10–18, November 2009.

[157] M. F. Porter. *Readings in Information Retrieval*. chapter An Algorithm for Suffix Stripping, pages 313–316. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1997.

[158] G. Salton and C. Buckley. *Term Weighting Approaches in Automatic Text Retrieval*. Technical report, Ithaca, NY, USA, 1987.

[159] W. B. Frakes and R. Baeza-Yates, editors. *Information Retrieval: Data Structures and Algorithms*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1992.