HAL
open science

Automatic discovery of ontology mappings

Rémi Tournaire

To cite this version:

Rémi Tournaire. Automatic discovery of ontology mappings. Computer Science [cs]. Université de Grenoble, 2010. English. NNT : . tel-00526791

HAL Id: tel-00526791

https://theses.hal.science/tel-00526791v1

Submitted on 15 Oct 2010

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Université de Grenoble

Thèse

Pour obtenir le grade de

Docteur de l'Université de Grenoble

Spécialité : Informatique

Arrêté ministériel : 7 août 2006

Présentée et soutenue publiquement par

Rémi Tournaire

le 8 octobre 2010

Découverte automatique de correspondances entre ontologies

Thèse dirigée par Marie-Christine Rousset et codirigée par Jean-Marc Petit

Jury

Chantal	Reynaud	Professeur Paris XI	Présidente
Colin	de la Higuera	Professeur Université de Nantes	Rapporteur
Stefano	Spaccapietra	Professeur EPFL	Rapporteur
Jean-Marc	Petit	Professeur INSA Lyon	Examinateur
Marie-Christine	Rousset	Professeur Université de Grenoble	Examinatrice
Alexandre	Termier	Maître de conférences	Examinateur

Thèse préparée au sein du Laboratoire d'Informatique de Grenoble, dans l'École Doctorale Mathématiques Sciences et Technologies de l'Information, Informatique

Mots-clefs :
alignement d'ontologies, Web sémantique, correspondance, probabilités, logique, instances, taxonomie, passage á l'échelle, Web Sémantique

Résumé :
Dans cette thèse, nous adoptons une approche formelle pour définir et découvrir des mappings d'inclusion probabilistes entre deux taxonomies avec une sémantique claire, dans l'optique d'échange collaboratif de documents. Nous comparons deux façons de modéliser des mappings probabilistes tout en étant compatible avec les contraintes logiques déclarées dans chaque taxonomie selon une propriété de monotonie, puis nous montrons que ces modèles sont complémentaires pour distinguer les mappings pertinents. Nous fournissons un moyen d'estimer les probabilités d'un mapping par une technique bayésienne basée sur les statistiques des extensions des classes impliquées dans le mapping. Si les ensembles d'instances sont disjoints, on utilise des classifieurs pour les fusionner. Nous présentons ensuite un algorithme de type "générer et tester" qui utilise les deux modèles de mappings pour découvrir les plus probables entre deux taxonomies. Nous menons une analyse expérimentale fouillée de ProbaMap. Nous présentons un générateur de données synthétiques qui produit une entrée contrôlée pour une analyse quantitative et qualitative sur un large spectre de situations. Nous présentons aussi deux séries de résultats d'expériences sur des données réelles : l'alignement du jeu de donnée "Directory" d'OAEI, et une comparaison pour l'alignement de Web Directories sur lesquels ProbaMap obtient de meilleurs résultats que SBI (IJCAI 2003). Les perspectives pour ces travaux consistent à concevoir un système de réponse à des requêtes probabilistes en réutilisant des mappings probabilites, et la conversion des coefficients retournés par les méthodes de matching existantes en probabilités.

Thèse effectuée au Laboratoire d'Informatique de Grenoble
Laboratoire LIG - UMR 5217
Maison Jean Kuntzmann - 110 av. de la Chimie - Domaine Universitaire de
Saint-Martin-d'Hères - BP 53
38041 Grenoble cedex 9 - France
Tel. : +33 (0)4 76 51 43 61 - Fax : +33 (0)4 76 51 49 85

Automatic discovery of ontology mappings

Keywords:
ontology matching, semantic web, mapping, probability, logic, instances, taxonomy, scalability, taxonomy, Semantic Web

Abstract:
In this thesis, we investigate a principled approach for defining and discovering probabilistic inclusion mappings between two taxonomies, with a clear semantic, in a purpose of collaborative exchange of documents. Firstly, we compare two ways of modeling probabilistic mappings which are compatible with the logical constraints declared in each taxonomy according to a monotony property, then we show that they are complementary for distinguishing relevant mappings. We provide a way to estimate the probabilities associated to a mapping by a Bayesian estimation technique based on classes extensions involved in the mapping, and using classifiers in order to merge the instances of both taxonomies when they are disjoint. Then we describe a generate and test algorithm called ProbaMap which minimizes the number of calls to the probability estimator for determining those mappings whose probability exceeds a chosen threshold. A thorough experimental analysis of ProbaMap is conducted. We introduce a generator that produce controlled data that allows to analyse the quality and the complexity of ProbaMap in a large and generic panel of situations. We present also two series of results for experiments conducted on real-world data: an alignment of the Directory dataset of the Ontology Alignment Evaluation Initiative (OAEI), and a comparative experiment on Web directories, on which ProbaMap outperforms the state-of-the-art contribution SBI (IJCAI'03). The perspectives of this work are the reuse of probabilistic mappings for a probabilistic query answering setting and a way to convert similarities coefficients of existing matching methods into probabilities.

RÉSUMÉ

Dans cette thèse, nous adoptons une approche formelle pour définir et découvrir des mappings probabilistes entre deux taxonomies.

En dépit de la quantité de méthodes existantes pour l'alignement d'ontologies, un défi d'envergure consiste à associer une sémantique formelle aux correspondances tout en prenant en compte leur incertitude inhérente, à l'échelle du Web. De plus, les correspondances d'inclusions (que nous appelons mappings) sont de première importance pour l'échange de données, et sont plus susceptibles d'exister que des équivalences. A notre connaissance, peu de travaux se sont concentrés sur ces correspondances d'inclusions jusqu'à présent.

Dans un premier temps, nous comparons deux façons de modéliser des mappings probabilistes tout en étant compatible avec les contraintes logiques déclarées dans chaque taxonomie. Nous présentons deux fonctions probabilistes qui associent à chaque mapping une valeur de confiance, en modélisant les classes comme des évènements probabilistes dans un univers d'instances. La première fonction probabiliste P_c associe au mapping $A \sqsubseteq B$ la probabilité conditionnelle de B sachant A, et la seconde fonction P_i lui associe la probabilité de l'évènement $\overline{A} \cup B$, qui provient de l'implication logique.

En analysant les propriétés de ces deux fonctions probabilistes, nous montrons qu'elles sont complémentaires, en particulier pour reconnaître des mappings pertinents. Nous prouvons une propriété de monotonie pour les deux fonctions probabilistes P_c et P_i par rapport à l'implication logique: un mapping doit avoir une probabilité inférieure à celle de toutes ses conséquences. C'est vrai directement pour P_i (propriété forte), et valable pour P_c si on restreint les conséquences aux mappings qui ont la même classe à gauche (propriété faible).

De plus, nous montrons un résultat plus général: sous certaines hypothèses, toutes les fonctions pour mesurer le degré de confiance des mappings qui respectent la propriété forte de monotonie doivent être de la forme $f(P_i(A \sqsubseteq B))$ avec f croissante.

Nous fournissons un moyen d'estimer les probabilités d'un mapping par une estimation bayésienne basée sur les statistiques des extensions des classes impliquées dans le mapping.

Pour le cas où les ensembles d'instances des deux taxonomies sont disjoints, nous expliquons comment appliquer une méthode de classification supervisée pour les fusionner.

Nous présentons ensuite l'algorithme ProbaMap de type "générer et tester" qui utilise les deux modèles de mappings probabilistes pour découvrir les plus probables entre deux taxonomies.

Nous donnons tous les détails d'implémentation qui sont nécessaires pour obtenir le passage à l'échelle de cet algorithme.

Nous menons une analyse expérimentale fouillée de ProbaMap. Premièrement, nous fournissons les principes d'un générateur de données synthétiques. Ces données servent notamment d'entrées contrôlées de ProbaMap: taxonomies, instances et mappings à découvrir. Ce générateur respecte une bonne propriété qui justifie son utilisation pour évaluer ProbaMap. De telles données contrôlées permettent d'analyser la qualité et l'efficacité de ProbaMap sur de nombreuses situations, variées et représentatives.

Deuxièmement, nous présentons deux séries de résultats d'expériences faites sur des données réelles. La première partie concerne l'alignement du jeu de donnée "Directory" d'Ontology Alignement Evaluation Initiative. La deuxième partie concerne l'alignement de Web Directories provenant de Yahoo! et Google, sur lesquels une comparaison en terme de qualité avec SBI [ITH03] (IJCAI 2003) montre que ProbaMap donne de meilleurs résultats.

Nous concluons en donnant deux perspectives pour ces travaux. La premiŕe consiste à utiliser les probabilité des mappings dans un processus de raisonnement pour répondre à des requêtes et associer des probabilités aux réponses, dans l'esprit des travaux actuels sur les bases de données probabilistes. La seconde perspective concerne un algorithme de post-traitement pouvant s'appliquer à des méthodes de matching existantes, afin de transformer en probabilités les coefficients de confiance associés à chaque mapping par ces méthodes.

ABSTRACT

In this thesis, we investigate a principled approach for defining and discovering probabilistic correspondences between two taxonomies.

In spite of the numerous existing methods for ontology matching, an important remaining challenge consists in associating a clear and formal semantics for correspondences while handling their inherent uncertainty at the Web scale. In addition, we claim that inclusion correspondences (that we call mappings here) are very important in particular for enabling a collaborative exchange of document and are more likely to happen than equivalence correspondences. To the best of our knowledge, only a few work have focused on inclusion corespondences until now.

Firstly, we compare two ways of modeling probabilistic mappings which are compatible with the logical constraints declared in each taxonomy. Hence, we introduce two probabilistic functions that associate a confidence value to each mapping $A \sqsubseteq B$ where A and B are ontology classes, by modeling the classes as probabilistic events in an instance universe. The first probabilistic function P_c associates $A \sqsubseteq B$ to the conditional probability of B given A, and the second function P_i associates $A \sqsubseteq B$ to the probability of the event $\overline{A} \cup B$, that comes from the logical implication.

By analyzing the properties of both probabilistic functions, we show that they are complementary for distinguishing relevant mappings. We prove a property of monotony for the probabilistic functions P_i and P_c with regard to the logical entailments: a mapping should have a probability value lower than all its consequences. We have pointed out that this is true for P_i in any cases (strong property), and true for P_c only if the class on the left-hand side of the mappings remains the same (weak property).

We even show a more general result: under additional assumptions, all confidence functions that respect the strong property of monotony should be of the form $f(P_i(A \sqsubseteq B))$ with f a monotonous increasing function in $[0; 1]$

We provide a way to estimate the probabilities associated to a mapping by a Bayesian estimation technique based on statistics on classes extensions involved in the mapping. In the case where the two sets of instances of both taxonomies are disjoint, we describe an optional classification phase intended to merge the instances.

Then we describe a generate and test algorithm called ProbaMap which minimizes the number of calls to the probability estimator for determining those mappings whose probability exceeds

a particular threshold. We give all details on the implementation of ProbaMap that enable its scalability.

A thorough experimental analysis of ProbaMap is conducted. Firstly, we introduce a generator in order to produce synthetic data to be given to ProbaMap as a controlled input: taxonomies and instances, with the reference mappings to be discovered. This generator respects a property that justifies the evaluation of ProbaMap on the data it generates. Controlled data allows to analyse the quality and the complexity of ProbaMap in a large and generic panel of situations.

Secondly, we present two series of results for experiments conducted on real-world data: an alignment of the Directory dataset of the Ontology Alignment Evaluation Initiative (OAEI), and an experiment on parts of Web directories Yahoo! and Google, on which ProbaMap outperforms a state-of-the-art contribution SBI [ITH03] (IJCAI'03).

We conclude by giving two perspectives. The first one reuse probabilistic mappings by integrating them in a reasoning process for query answering, in order to obtain a probability for each answer to a query, in the spirit of the current works on probabilistic databases. A second perspective is presented for the transformation of confidence coefficients usually associated with mappings by many existings matching methods into mapping probabities.

REMERCIEMENTS

Ce travail est avant tout celui d'un groupe, ma directrice Marie-Christine ROUSSET, mon co-directeur Jean-Marc PETIT, et Alexandre TERMIER. Par leur rigueur, leurs idées, leurs connaissances expertes en informatique, ils ont contribué de façon majeure aux résultats de cette thèse. Je voudrais souligner la constance, la rigueur, les idées et l'énorme travail d'encadrement de Marie-Christine. Il faut relever les conseils avisés, le recul expérimenté, ainsi que les idées de Jean-Marc qui ont notamment permis de lier ce travail au domaine des bases de données. Enfin, il convient de saluer l'expertise d'Alexandre sur les aspects expérimentaux, ainsi que ses conseils clairs et quotidiens, qui font de lui un contributeur majeur de ces travaux. Les qualités scientifiques de ces trois professeurs n'enlèvent rien à leurs qualités relationnelles, et je les ai beaucoup appréciés.

Je tiens également à remercier l'équipe HADAS et le laboratoire LIG dans lesquels j'ai évolué, présenté notre travail et recueilli des avis pertinents. Un merci particulier à Christine COLLET, chef d'équipe HADAS.

Je remercie tous les membres du jury qui ont montré leur intérêt particulier pour ce travail. Merci beaucoup aux deux rapporteurs, Colin DE LA HIGUERA et Stefano SPACCAPIETRA pour l'acuité de leurs remarques et le temps consacré à leur rapport. Merci beaucoup également à Chantal REYNAUD pour son intérêt ainsi que son rôle dans le jury.

Je remercie la région Rhône-Alpes et le projet WebIntelligence (cluster ISLE) pour le financement de cette thèse.

Un grand merci à ma famille et tout particulièrement à mes parents qui m'ont soutenu avec une constance et une foi infinie autant sur les aspects matériels que relationnels, leurs efforts ont été considérables. Enfin, une mention spéciale à mes amis, et aux collègues Benjamin NEGREVERGNE et Sattisvar TANDABANY, qui ont soutenu cette thèse autant que je m'apprête à le faire le 8 octobre 2010.

This work is actually a joint work conducted by my supervisor Marie-Christine Rousset, my co-supervisor Jean-Marc Petit, and Alexandre Termier. They have contributed in a prominent way to this thesis with their rigour, their ideas and their expert knowledge in the informatics domain. I would like to emphasize the constancy of Marie-Christine, her exactness, her ideas and the huge amount of work for supervision she has done. One should point out the relevant advices, the experience and the ideas of Jean-Marc, that especially lead to connect this work with the domain of databases. Finally, I should underline the competences of Alexandre on experimental aspects, and also its clear and dayly advices, which make him a major contributor of this thesis. Their qualities of scientists should not cover their human qualities which have been very important for me.

I thank a lot the HADAS group and the LIG laboratory where I have worked everyday, and gathered a lot of relevant remarks. A particular thank is dedicated to Christine Collet, the HADAS chief.

I give thanks to all the members of the jury who have shown their particular interest for this work. Thank a lot to the two referees, Colin De La Highera and Stefano Spaccapietra for their very relevant and precise comments and the amount of time they have spent for their reports. A great thank for Chantal Reynayd for its interest and the role she plays in the jury.

I thank the Rhône-Alpes region and the WebIntelligence project (ISLE cluster) for the territorial funding.

A very large thank to my family and especially to my parents who have constantly supported me in every respects, their efforts have been significant. Finally, a special mention to my friends and my collegues Benjamin NEGREVERNE and Sattisvar TANDABANY, who have helped me more than they actually think.

Contents

I Introduction **1**
- I.1 The Semantic Web . . . 3
- I.2 The need of correspondences between ontologies . . . 7
  - I.2.1 Context and motivation . . . 7
  - I.2.2 A motivating example: the Somewhere project . . . 8
  - I.2.3 Ontology matching . . . 10
- I.3 Issues addressed in this thesis . . . 10

II Preliminaries and problem statement **13**
- II.1 Taxonomies: classes and instances . . . 13
- II.2 Logical semantics . . . 15
- II.3 Correspondences, mappings and alignments . . . 16
- II.4 Probability measure . . . 20
- II.5 Bayesian estimation . . . 23
- II.6 Problem statement . . . 23

III State-of-the-art **25**
- III.1 Overview of existing methods for ontology matching . . . 25
  - III.1.1 String based and language-based techniques . . . 25
  - III.1.2 Structural matching techniques . . . 26
  - III.1.3 Instance-based matching techniques . . . 28
  - III.1.4 Matching based on logical reasoning . . . 33
  - III.1.5 Properties of matching methods . . . 34
  - III.1.6 Illustrative examples of real methods . . . 35
- III.2 Remaining challenges . . . 36
  - III.2.1 Handling uncertainty of correspondences . . . 37
  - III.2.2 Scalability . . . 37

Part I Models and algorithm: combining logic and probability **39**

IV Probabilistic models of mappings **41**
- IV.1 Conditional probabilistic model . . . 42
- IV.2 Implication probabilistic model . . . 43
- IV.3 Comparative study of the two probabilistic models . . . 44

IV.4 Estimation of probabilities 51

V ProbaMap: an algorithm for discovering mappings 55
V.1 Candidate mapping generation 56
V.2 Pruning the candidate mappings to test 59
V.3 The ProbaMap algorithm 60
V.3.1 Implementation of ProbaMap 62
V.3.2 Implementation of the computation of probabilities estimations . . . 63
V.3.3 Implementation of the pruning functions 63
V.4 Classification phase 68
V.5 Complexity 69

Part II Experiments 75

VI Experiments on controlled synthetic data 77
VI.1 Synthetic data generation 78
VI.2 Experimental protocol 85
VI.3 Experimental results 86

VII Experiments on real-world data 97
VII.1 OAEI directory benchmark 97
VII.1.1 Dataset and experimental settings 97
VII.1.2 Results 101
VII.2 Comparative analysis on collected Web Directories 104
VII.2.1 Experimental protocol 105
VII.2.2 Results 106

VIII Conclusion 115

A Proofs 119
A.1 Proofs for the characterization of monotonous confidence functions 119
A.1.1 Proof of the Lemma IV.α 119
A.1.2 Proof of the Theorem IV.2 119
A.2 Proof of the correctness of the generator 123
A.2.1 Proof of the proposition VI.1 123
A.2.2 Proof of the Theorem VI.1 124

B Résumé long en Français 127
B.1 Introduction 127
B.2 Préliminaires et position du problème 130
B.3 Etat de l'art 133
B.4 Modèles probabilistes de mappings 135
B.5 ProbaMap: un algorithme de découverte de mappings 138

B.6 Expérimentations sur données synthétiques 140
B.7 Expérimentations sur données réelles 142
B.8 Conclusion 145

List of Figures **148**

References **151**

CHAPTER

ONE

INTRODUCTION

Following the exponential development of Internet and the amount of connected machines, data able to be exchanged through the entire world has reached an incredible level raising more and more challenging issues.

Figure I.1 shows that the World Wide Web has grown up to more than 100 million hosts, corresponding to around 20 billion pages, and 1.7 billion users.

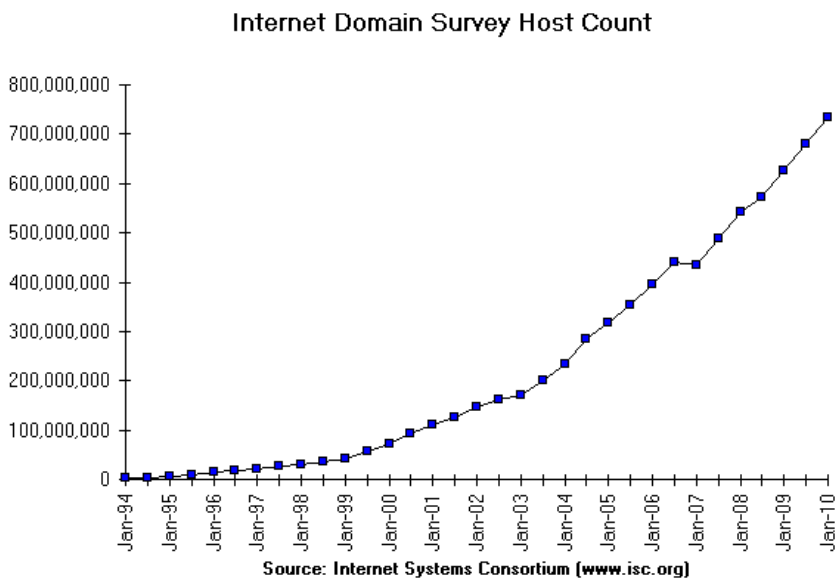


Figure I.1: Size evolution of the World Wide Web (hosts) - ISC estimation (Feb. 2010)

The Web is constituted by heterogeneous contents like HTML documents for the main part, multimedia documents as music and video files, and other documents everybody stores on his computer. Most people query the web via keyword search engines like Google, Yahoo!, Bing,

Search Engine	Self-Reported Size (Billions)	EstimatedSize (Billions)	Coverage of Indexed Web (%)	Coverage of Total Web (%)
Google	8.1	8.0	76.2	69.6
Yahoo	4.2 (est.)	6.6	69.3	57.4
Ask	2.5	5.3	57.6	46.1
MSN (beta)	5.0	5.1	61.9	44.3
Indexed Web	N/A	9.4	N/A	N/A
Total Web	N/A	11.5	N/A	N/A
Note: "Indexed Web" refers to the part of the Web considered to have been indexed by search engines.				

Figure I.2: Web size and indexation coverage

Baidu....All of them index part of the Web by crawling it and parsing the text in the web documents. Then they are able to select and rank these documents according to a textual user query. Documents containing the words in query are ranked by relevance computed both on textual and popularity criteria. Figure I.2 illustrates the coverage by main search engines estimated in 2005 [GS05]. Note that no indexation exceeds 70%, even if such coverages are impressive in term of absolute figures. In the year 2010, Yahoo!, Google and Bing are independently indexing more than several tens billion pages[1].

It should be emphasized that there is a sharp contrast between the amount of available data and the lack of expressive and qualitative ways to access to it. For instance, up to recently, classical Web search engines have not taken into account synonyms or homonyms ; they ignore the meaning of common words like “of”, “with”, “by” occurring between two words ; they do not provide exact answers to a query, but a very large list of Web pages you are invited to read to find your answer instead. A fortiori they do not allow to cross multiple source of information to obtain an answer to a complex query like “In which town died the mother of J.S. Bach ?”. Due to the current upper bound of 70% coverage, we can not expect any completeness of the answer returned by search engine (while assuming the set of correct and complete answers to be small and then human readable.) Finally, the context in which the user is querying the Web is not taken into account, and this leads to a lot of incorrect answers.

Obviously, these drawbacks are currently tackled by the main search engine companies and a lot of improvements have been made to guess the context of a query, to avoid bad interpretations and to start handling synonyms and homonyms. But such solutions are based on the huge amount of statistics gathered from users: they are ad-hoc solutions given a particular search engine and they do not deeply change the core of the method, i.e. keyword search on textual indexed documents.

Drawbacks of search engines are not imputable to their intrinsic quality but rather to the poor expressive level of the data they manipulate. Indeed, current web documents are mainly constituted by text or multimedia content organized into HTML files. HTML can express the style (with CSS), layout and behavior of a web pages, but nothing about the meaning of each con-

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

tent object. Therefore, although most web pages respect last HTML standards, their contents remain not understandable by machines. This semantic gap between the syntactic content and the meaning of a web page is at the origin of the emergence of a new Web of data: the Semantic Web. The infrastructure capable of connecting billions of computers to the Web now appears to be a mature and well-managed challenge, but the semantic management of this huge amount of data constitutes a challenging second phase in the Web history.

I.1 The Semantic Web

Two main parts have to be distinguished in the construction of the Semantic Web: the semantic description of data and the way to process data.

In order to allow documents components to provide an explicit meaning, they should be annotated and completed by elements of special semantic languages. Figure I.3 shows an example of two Web pages annotated by using labels declared in an *ontology* (at the top), expressed in the RDF Schema languages [Hay04]. An ontology is a structured vocabulary composed of names of classes (or concepts) and properties on these classes. They are defined in formal languages like descriptions logic or conceptual graphs, for instance. In the setting of the Web, two standard are provided by the W3C: RDFS [Hay04] and OWL 1 & 2 [DS04, MPP+08]. Like the first-order logic, all these languages have their own defined semantics. In the example of Figure I.3, two home pages of people of Karlsruhe University at the urls
`http://www.aifb.uni-karlsruhe.de/WBS/sha` and
`http://www.aifb.uni-karlsruhe.de/WBS/sst`
are annotated using respectively classes `PhD Student` and `AssProf` which are both subclasses of `AcademicStaff`. Both urls are then respectively instances of `PhD Student` and `AssProf`. The PhD Student Siegfried Handschuh is asserted to `cooperate_with` the assistant professor Steffen Staab. This relationship is possible because the ontology asserts that an instance of `PhD Student` can be linked to an instance of `AssProf`. (that is the meaning in RDFS of `rdfs:domain` and `rdfs:range`). A triple like

```
(http://www.aifb.uni-karlsruhe.de/WBS/sha, rdf:type,PhD Student)
```

declares the url of Siegfried Handschuh to be an instance of the class `PhD Student`. It is called a RDF triple and is the basic element constituting the Semantic Web.

Figure I.4 represents the most common generic view of the Semantic Web in term of levels, from low-level of data, e.g. encoding language then structured language, to the richest abstractions of data, i.e. logical language then inference rules.

For instance, data dictionaries, thesauri, XML schemas, database schemas, taxonomies are examples of ontologies (ordered by increasing expressive power), that can all be expressed in OWL. Taxonomies are essential components of ontologies: they represent the skeleton hierarchy of classes for every ontology. Taxonomies contains classes and subsumptions relations between them, and are themselves a particular kind of ontologies.

Below are listed several kinds of ontologies. Their applications, their size and their expressivities could be very different:

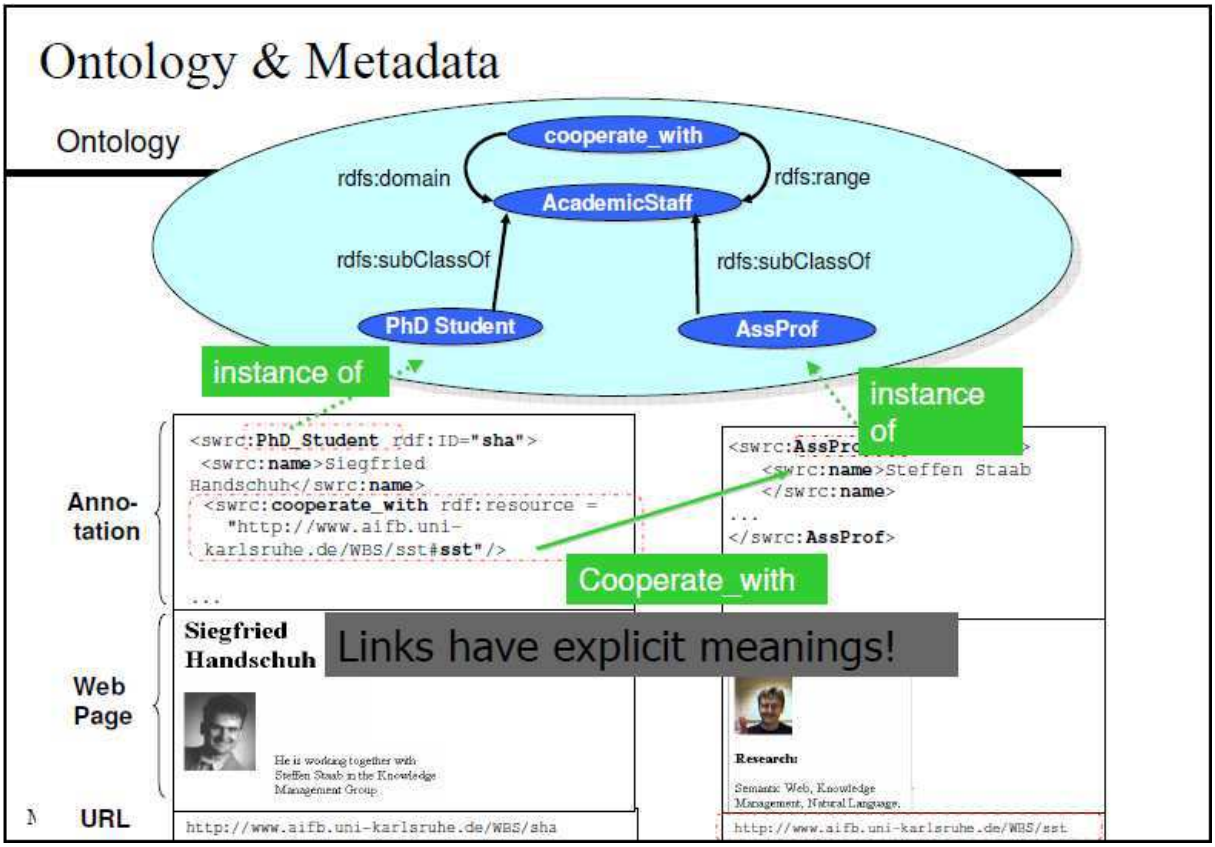


Figure I.3: Semantic annotation of Web pages

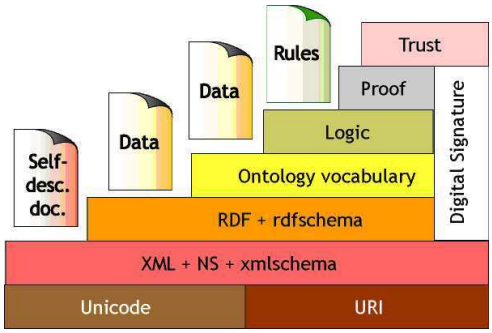


Figure I.4: Semantic Web Stack [BL00]

- Domain-specific ontologies are used to describe a particular domain of interest: for instance: ontologies of diseases, classification of life in medical sciences and biology, classification of musical genres and phylogenetic classification of life.

  Most of the domain-specific ontologies are large (more than 100 classes or entities) and engineered by specialists.

  For example,

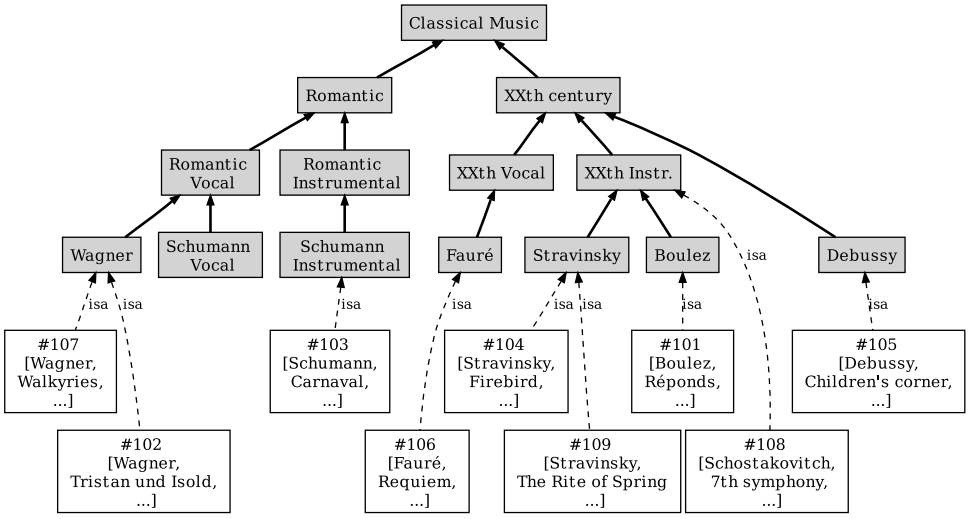
  - the Foundational Model of Anatomy ontology[2] is a "knowledge source for biomedical informatics. It is concerned with the representation of classes or types and relationships necessary for the symbolic representation of the phenotypic structure of the human body in a form that is understandable to humans and is also navigable, parseable and interpretable by machine-based systems" (description extracted from the web page).
  - RAMEAU[3] is an ontology about the indexation vocabulary designed for the French National library.
  - GTAA[4] is the "Common Thesaurus for Audiovisual Archives" used at the Netherlands Institute for Sound and Visio. It contents descriptions about the topics, people, location, genre, makers/presentators, corporations names/music bands mentioned in TV programs.

- Folksonomies are taxonomies independently created by people who want to organize or express knowledge from an user point-of-view. They are often leight-weight ontologies (small). Two examples of folksonomies which are taxonomies are given in Figure I.5. We can see that these taxonomies in this example may be used to annotate and organize musical documents in two different ways.

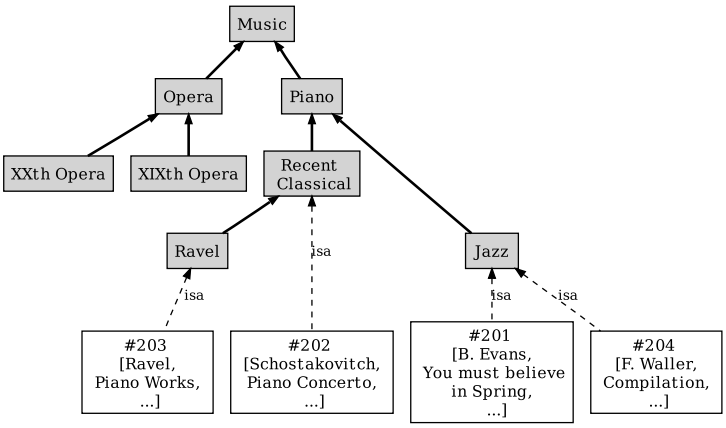
- Cyc [MCWD06] is an attempt to gather all usual knowledge everybody knows. It is not domain-specific because of its large coverage of almost everything. Conversely, it is very huge, and this project born more than 20 years ago has not managed to fit the initial goal.

Figures I.6 shows the index size of Swoogle [FDP+05] which indexes RDF statements. In 2010, 10,000 ontologies, 3.7 million Semantic Web documents and more than a billion RDF statements are indexed by Swoogle. In November 2009, a lower bound was 13.1 billion RDF triples collected in the Linking Open Data project[5]. This shows that the Semantic Web is still small compared to the Web (magnitude of 1000). But such an amount of ontologies is already large and should grow up in an exponential way. Therefore there is a real need of structured knowledge in specialized, commercial and all-public fields. The standardization of ontologies languages and the fair degree of maturity in ontology knowledge domain will facilitate the emergence of such a Semantic Web.

[2] `sig.biostr.washington.edu/projects/fm/AboutFM.html`
[3] `http://www.cs.vu.nl/STITCH/rameau/`
[4] `http://oaei.ontologymatching.org/2009/vlcr/#gtaa`
[5] `http://esw.w3.org/SweoIG/TaskForces/CommunityProjects/LinkingOpenData`



(a) Taxonomy $\mathcal{T}_1$



(b) Taxonomy $\mathcal{T}_2$

Figure I.5: 2 small personalized taxonomies

admin_dt	2010-03-17 23:59:04	Datetime Watched
url_total	9,930,096	Number of URLs being discovered
url_pinged	5,139,276	Number of URLs being pinged
total_swd	3,233,224	Number of Semantic Web Documents (regardless of embedded or containing some errors) be comfirmed.
total_swd_strict	1,803,941	Number of error-free pure Sematic Web Documents
total_swd_embed	1,146,677	Number of documents (except SWDs, PDF, and JPEG) embedding Semantic Web Data
triple_total	1,075,282,347	Number of triples could be parsed from all Semantic Web Documents.

Figure I.6: Figures about the Swoogle index

I.2 The need of correspondences between ontologies

We describe here the reasons for which ontology matching is necessary to enable exchange of annotated data through the Web.

I.2.1 Context and motivation

At the web scale, multiple users or organizations are likely to declare their own knowledge ontology for describing and annotating their shared documents. In this setting, discovering correspondences between ontologies is a major issue for the Semantic Web, for instance to enable collaborative exchange of documents. Many domain-ontology describing the same domain coexist as well as many folksonomies independently created by different users. Multiple (human-specific or not) reasons explains heterogeneity while describing a common knowledge [ES07] :

1. *terminological heterogeneity*: this kind of heterogeneity arises when different terms are used to describe exactly the same concepts, for instance when using multiple languages.

2. *semiotic heterogeneity*: concerns the way by which entities are interpreted by people, depending of the context of usage. This is hardly handled by machines, because of the poor knowledge of the real context of users.

3. *difference on coverage*: the covered domains are not the same, so the classes (concepts) used in the ontologies do not represent the same things in two different ways. For example, the taxonomy in Figure I.5(a) only deals with classical music whereas the second taxonomy (Figure I.5(b) covers classical music *and* jazz. A parallel can be made with geographical maps as represented in Figure I.7. Figure I.7(d) represent only a part of France that is the Auvergne region, whereas Figures I.7(c) and I.7(a) represent the whole country.

4. *difference on granularity*: it occurs when the same domain is described by different levels of details. In Figure I.5, the classes are more general in average in $\mathcal{T}_2$ than in $\mathcal{T}_1$, because of the presence of a lot of classes by composers. Again, a parallel can me made with maps: Figure I.7(b) is more specific than I.7(a) because administrative subdivisions of regions are detailed.

5. *difference on perspective*: the point of view of an ontology engineer let him to emphasize some particularity leading to a particular hierarchy for concepts based on his own criteria. This may result on multiple ways to create an ontology and conducts to a strong heterogeneity. For example, it can be pointed out that $\mathcal{T}_1$ has been created by a more systematic way than $\mathcal{T}_2$ because there is a criteria used by level of class (period, vocal/instrumental, and then composer), whereas in $\mathcal{T}_2$, *Ravel* is the only composer who corresponds to a class, and the distinction criterion between *Jazz* and *Recent classical* is not a time distinction like the one between the two subclasses of *Opera*. The difference between the Figures I.7(c) and I.7(a) gives an illustration of multiple points of view of the same part of the world: the first one is administrative, the second one is topological.

As a consequence these reasons represent a serious issue when attempting to give a machine-understandable semantics to Web components at the Web scale. In this context, *establishing semantic correspondences is the key to enable the exchange of collaborative data in the Web.*

I.2.2 A motivating example: the Somewhere project

A motivating example for this thesis is the Somewhere infrastructure for managing distributed data [ACG+05, ACG+06, Adj06] in a peer-to-peer context. In such a setting, each peer user annotates its documents with classes of his own simple and personalized taxonomy like these on Figure I.5. The natural process and interest of is the possibility to exchange documents: each peer can access to every document stored in another peer, depending on what he asks for via a query expressed in its own vocabulary. For example, if the peer P_1 annotates his resources with $\mathcal{T}_1$, and the peer P_2 annotates its ones with $\mathcal{T}_2$ he can submit the query $XXth\ Instrumental_1$, in order to find more songs of this class he has created. As the vocabularies are disjoint (here by indexing each class label by an identifier of the peer which declares it), no other peer knows nor uses the label $XXth\ Instrumental$. At this point, the system needs some correspondences between the $XXth\ Instrumental$ class and other classes asserted by other peers of the network. Provided such correspondences (supposed given), the Somewhere distributed reasoning algorithm (called DeCA) computes the answer to the query through all the network and collects all the corresponding documents.

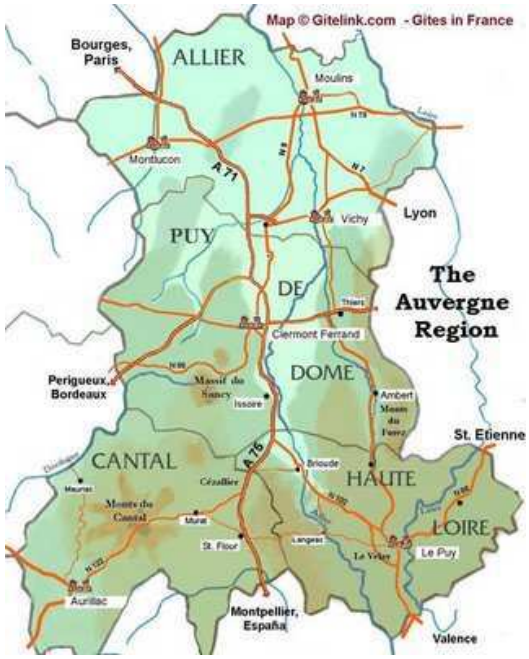
Somewhere illustrates a vision of the Semantic Web seen as a huge collaborative peer-to-peer data management system based on simple ontologies and correspondences distributed at a large scale.

In Somewhere, the correspondences are declared manually by users, which is a counter-intuitive task for them: the interest of such a P2P setting is that each user defines its own personalized taxonomy without knowing much about other peers. In addition to that, such a task may be incomplete and the number of possible correspondences makes the problem hardly tractable and strongly repetitive for humans at the Web Scale. Therefore the ontology matching domain tackles the issue of discovering automatically such correspondences.

(a) Physical map

(b) Political map 2

(c) Political map 1



(d) Regional map (Auvergne region)

Figure I.7: Heterogeneity of ontologies: parallel with maps

I.2.3 Ontology matching

In response to the generalized heterogeneity on the growing amount of available ontologies on Internet, a specific domain has appeared since fifteen years: the so-called *Ontology matching* domain. Ontology matching studies the ways to automatically discover some correspondences between two (or more) ontology entities (classes, relations, properties, instances) belonging to different ontologies [ES07]. Such correspondences may be of different kinds according the discovered relation. For instance, an equivalence discovered between two classes imply that one can replace the name of the first one by the name of the other everywhere. Ontology matching is needed in different tasks like ontology, schema or data integration from multiple sources, ontology engineering, evolving or merging, query rewriting between ontologies (using correspondences as substitution relations), Web service composition,... Thus, ontology matching enables the knowledge and data expressed in the matched ontologies to interoperate.

By nature, automatic discovery of correspondences between different ontologies is a very complicated task. Deep reasons of heterogeneity between them are not explicitly known by machines as explained before.

Since the last fifteen years plenty of work has been made on ontology matching (see [SE05, RB01] for surveys). Existing methods are based on lexical or linguistic similarities on labels, comparison on the graph structures of relations in ontologies. Some of them exploit characteristics of the data declared as instances of the classes, while other ones exploit an external resource as an additional knowledge. Many implemented methods are actually some combinations of these atomic techniques [MBR01, WX09].

Since 2005 the yearly international contest *Ontology Alignment Evaluation Initiative (OAEI for short)*[6] [EFH+09] is organized in order to make a qualitative comparison between several methods on a pool of very different benchmarks.
A current conclusion of all of these evaluations is the following: there is considerable progress made since the beginnings of this research domain but there is not a clear winner yet. An integrated solution which is robust enough to be the basis for future development and usable by non expert users still lacks.

I.3 Issues addressed in this thesis

In this thesis, we focus on automatic discovering of correspondences between taxonomies, that represent the essential component of the ontologies (i.e. the class hierarchy). Our first claim is that *inclusion* correspondences between classes of two pre-existing taxonomies are more likely to exist than *equivalence* correspondences. When taxonomies are used as query interfaces between users and data, inclusion correspondences between taxonomies can be used for query reformulation exactly like the subclass relationship within a taxonomy. For instance, a correspondence $Opera \sqsubseteq Vocal$ between the class $Opera$ of a taxonomy and the class $Vocal$ of a second taxonomy may be used to find additional answers to a query asking data about $Vocal$ by returning data categorized in the class $Opera$ in the first taxonomy. Since most of the approaches are based on similarity functions that are symmetric, the correspondences that are returned with

[6] E.g., OAEI `http://oaei.ontologymatching.org/2009/`

high similarity scores are interpreted as *equivalence correspondences*. As far as we know, few approaches (e.g., [BSZ03, HSNR09, SVV08, JK07]) with regard to the amount of existing ones handle inclusion correspondences between classes. Except until very recently, only equivalence correspondences have been considered in the OAEI campaigns.

Then we claim that *uncertainty* is intrinsic to correspondence discovery, due to the inherent heterogeneity of ontologies: it is very unlikely that two classes exactly overlap each other. Beside the work done along the line of uncertainty handling in the ontology matching research domain (e.g. [MGR+02], [MNJ05], [CFL+08]), there is still a need to better understand the foundations of modeling uncertainty in order to improve detection of correspondences causing inconsistencies, e.g., via probabilistic reasoning, or to identify where the user feedback is maximally useful [SE08].
In particular, almost all existing matching systems return for every candidate pair of elements a coefficient in the range [0;1] which denotes the strength of the semantic correspondence between those two elements. Once again, those coefficients are the basis for yearly international comparative evaluation campaigns [EFH+09]. Those approaches usually consider each candidate correspondence in isolation. In particular, they do not take into account possible logical implications between correspondences, which can be inferred from the logical inclusion axioms declared between classes within each ontology. This raises a crucial issue: the similarity coefficients returned by the existing ontology or schema matching systems cannot be interpreted as probabilities of the associated correspondences. Therefore, we advocate to consider *inclusion correspondences with a probabilistic semantics*.

Finally, the last point that we take into account with in this work is the *scalability* issue. Manually finding correspondences between ontologies is clearly not possible at the Web scale. Such a purpose is a major concern for the future Semantic Web. Therefore, the automatic discovery of semantic correspondences is the bottleneck for scalability purposes. Up to now, such an issue has not been really taken into account. OAEI campaigns gave only some preliminary evidence of the scalability characteristics of the ontology matching technology.

Contributions of the thesis

In this thesis, we propose an approach to discover automatically probabilistic inclusion correspondences between taxonomies, that we denote mappings. First, we investigate and compare two ways of modeling probabilistic mappings which are compatible with the logical constraints declared in each taxonomy. In those two probabilistic models, the probability of a mapping relies on the joint probability distribution of the involved classes. An additional result for characterizing the models that are compatible with the logical constraints under some assumptions is provided. In addition to that, we provide a way to estimate mapping probabilities based on statistics on declared instances.

Second, based on the above probabilistic setting, we have designed, implemented and experimented a generate and test algorithm called ProbaMap for discovering the mappings whose probability is greater than a given threshold. A property of monotony of the probability function is exploited for avoiding the probability estimation of as many mappings as possible, leading to a scalable algorithm.

In the perspective of experiments, we have designed a full generator of taxonomies, instances

and mappings to discover, in order to be able to provide a controlled input to ProbaMap, while providing the corresponding reference of mappings to discover. Experimental results about quality and time efficiency of ProbaMap are thoroughly analysed thanks to this generator whose parameters are crossed with thoses of ProbaMap.

We present two series of experiments that we have conducted on real-worl data. The first series is done on the OAEI directory dataset. The second serie compares ProbaMap with the state-of-the-art integration tool SBI [ITH03] in term of accuracy of the returned alignment on collected directories from Yahoo! and Google. We show that ProbaMap outperforms this existing contribution.

Finally, we sketch two perspectives. The first one is about the reuse of probabilitsitc mappings for probabilistic reasoning in a probabilistic query setting. The second one introduces a way to postprocess the set of confidence coefficients of discovered mappings returned by most of existing matching method to enable them being interpreted as probabilities.

Outline

This thesis is organized as follows:

- The **Chapter II** presents the required formal background and the problem statement
- The **Chapter III** presents the state-of-the-art in the Ontology Matching domain
- The **Part I** provides two models of *probabilistic mappings* and an algorithm to find the most probable.
  - The **Chapter IV** presents and deeply compare the two models of probabilistic mappings that we introduce in this work and a way to estimate these probabilities using a bayesian estimator and classifiers.
  - The **Chapter V** introduces and explains the ProbaMap algorithm and its implementation.
- The **Part II** is devoted to the experiments with the ProbaMap algorithm
  - The **Chapter VI** presents the experiments done on generated synthetic data and details the principles of the generator
  - The **Chapter VII** presents the experiments performed on real data
- The conclusion and some perspectives are given in **Chapter VIII**.

CHAPTER

TWO

PRELIMINARIES AND PROBLEM STATEMENT

We first define taxonomies as a graphical notation and its interpretation in the standard first-order logical semantics, on which the inheritance of instances is grounded. Then, we define *mappings* between taxonomies as inclusion statements between classes of two different taxonomies. We give some useful reminders about probabilities, random variables and bayesian estimation. Finally, we precisely define the problem statement of matching taxonomies that we consider in this work.

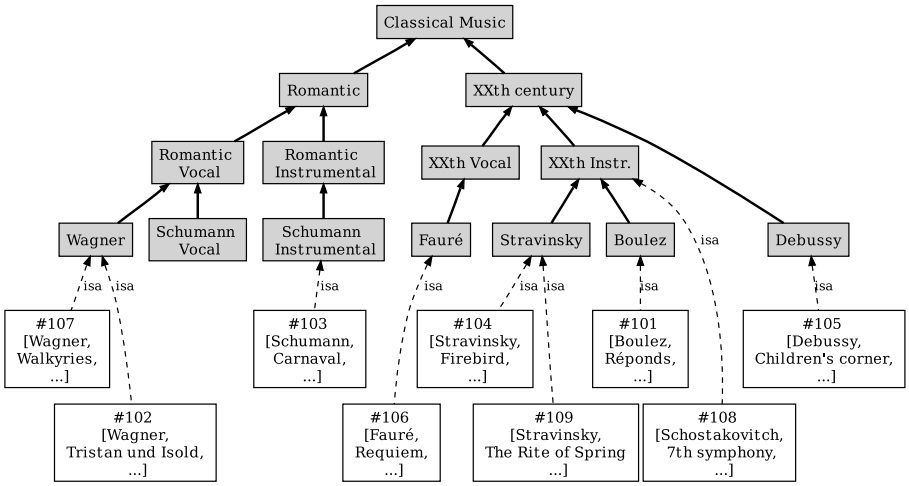
II.1 Taxonomies: classes and instances

Definition II.1 (Directed Acyclic Graph (DAG)):
A directed graph *is a pair* (V, E) *where* V *is a finite collection of vertices and* $E \subseteq V^2$ *a finite collection of directed edges* $(x, y) \in E$ *that connects the vertex* x *to the vertex* y*.*

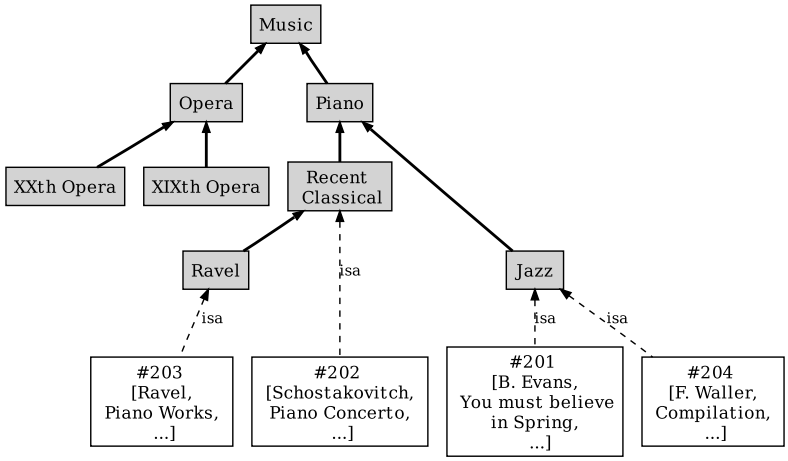
If there is no way to start at some vertex $v \in V$ *and follow a sequence of edges that eventually loops back to* v *again, the directed graph does not contain any cycle, and it is then called a* Directed Acyclic Graph*.*

Definition II.2 (Taxonomy):
Given a vocabulary $\mathcal{V}$ *denoting a set of classes, a taxonomy* $\mathcal{T}_{\mathcal{V}}$ *is a Directed Acyclic Graph (DAG) where each node is labelled with a distinct class name of* $\mathcal{V}$*, and each arc between a node labelled with* C *and a node labelled by* D *represents a specialization/generalization relation between the classes* C *and* D*.*
Each class in a taxonomy can be associated with a set of instances *which have an* identifier *and a content* description *modeled with an attribute-value language.*

By a slight abuse of notation, we speak of the instance i to refer to the instance identified by i.



(a) Taxonomy $\mathcal{T}_1$



(b) Taxonomy $\mathcal{T}_2$

Figure II.1: 2 Taxonomies and associated instances

Figure II.1 shows two samples of taxonomies related to the Music domain (there are already pictured in the previous chapter). Bold arrows are used for representing specialization relations between classes, and dashed arrows for membership relation between instances and classes. In both taxonomies, some instances, with attribute-value description denoted between brackets, are associated to classes. For example, #102 is an instance identifier and [Wagner, Tristan und Isold, ...] its associated description for the attribute sequence [Composer, Title, ...].

The instances that are in the scope of our data model can be web pages (whose content description is a set of words) identified by their URL, RDF resources (whose content description is a set of RDF triples) identified by a URI, or audio or video files identified by a signature and whose content description may be attribute-value metadata that can be extracted from those files.

In this thesis, we consider only boolean attribute-value description. Such a description can be obtained by discretization of attribute-value pairs given in a more complex language, like in Figure 1 where textual values are used. We consider that, possibly after a preprocessing which

is out of the scope of this work, the instances are described in function of a fixed set of boolean attributes $\{At_1, \ldots, At_m\}$. Then, for an instance i, its description, denoted *descr(i)*, is a vector $[a_1, \ldots, a_m]$ of size m such that for every $j \in [1..m]$, $a_j = 1$ if the attribute At_j belongs to the content description of i, and $a_j = 0$ otherwise.

Taxonomies have a logical semantics defined in section II.2 which provides the basis to define formally the extension of a class as the set of instances that are declared or can be *inferred* for that class.

II.2 Logical semantics

In addition of the graphical notation, there are several textual notations for expressing the specialization relation between a class C and a class D in a taxonomy. For example, in RDF(S) [Hay04] which is the first standard of the W3C concerning the Semantic Web, it is denoted by $(C\ rdfs{:}subclassOf\ D)$. It corresponds to the inclusion statement $C \sqsubseteq D$ in the description logics notation.

Similarly, a membership statement denoted by an *isa* arc from an instance i to a class C corresponds in the RDF(S) notation to $(i\ rdf{:}type\ C)$, and to $C(i)$ in the usual notation of description logics.

All those notations have a standard model-theoretic logical semantics based on interpreting classes as sets: an *interpretation* $\mathcal{I}$ consists of a non empty domain of interpretation $\Delta^{\mathcal{I}}$ and a function $.^{\mathcal{I}}$ that interprets each class as a non empty subset of $\Delta^{\mathcal{I}}$, and each instance identifier as an element of $\Delta^{\mathcal{I}}$. According to the *unique name assumption*, two distinct identifiers a and b have a distinct interpretation ($a^{\mathcal{I}} \neq b^{\mathcal{I}}$) in any interpretation I.

Definition II.3 (Model of a taxonomy):
$\mathcal{I}$ is a model of a taxonomy $\mathcal{T}$ if:

- *for every inclusion statement $E \sqsubseteq F$ of $\mathcal{T}$: $E^{\mathcal{I}} \subseteq F^{\mathcal{I}}$,*
- *for every membership statement $C(a)$ of $\mathcal{T}$: $a^{\mathcal{I}} \in C^{\mathcal{I}}$.*

An inclusion $G \sqsubseteq H$ is *inferred* by a taxonomy $\mathcal{T}$ (denoted by $\mathcal{T} \models G \sqsubseteq H$) iff in every model $\mathcal{I}$ of $\mathcal{T}$, $G^{\mathcal{I}} \subseteq H^{\mathcal{I}}$.

A membership $C(e)$ is *inferred* by $\mathcal{T}$ (denoted by $\mathcal{T} \models C(e)$) iff in every model $\mathcal{I}$ of $\mathcal{T}$, $e^{\mathcal{I}} \in C^{\mathcal{I}}$.

Definition II.4 (Extension of a class):
Let $\mathcal{D}$ be the set of the instances associated to a taxonomy $\mathcal{T}$. The extension *of a class C in $\mathcal{T}$, denoted by $Ext(C, \mathcal{T})$, is the set of instances for which it can be inferred from the membership and inclusion statements declared in the taxonomy that they are instances of C:*

$Ext(C, \mathcal{T}) = \{d \in \mathcal{D} / \ \mathcal{T} \models C(d)\}$

II.3 Correspondences, mappings and alignments

For sake of coherence, we adopt the formalism proposed by [ES07] for definition of a correspondence and alignment between taxonomies. To avoid ambiguity and without loss of generality, we consider that each taxonomy has its own vocabulary: by convention we index the names of the classes by the index of the taxonomy to which they belong. For instance, when involved in a mapping, the class $XXth\ Opera$ of the taxonomy $\mathcal{T}_2$ of Figure 1 is denoted by $XXth\ Opera_2$ while the class $XXth\ Vocal$ of the taxonomy $\mathcal{T}_1$ is denoted by $XXth\ Vocal_1$.

Definition II.5 (Correspondence):
Given two taxonomies T_1 and T_2 with their own respective disjoint set of classes V_1 and V_2, a set of possible alignment relations Θ and an ordered set Ξ, a correspondence is a 5-tuple (id, C_1, C_2, r, n) such that:

- *id is a unique identifier of the given correspondence*
- *$C_1 \in V_1$ and $C_2 \in V_2$ are the two classes that are related*
- *$r \in \Theta$ is the kind of relation*
- *$n \in \Xi$ is the confidence value associated to the correspondence, measuring the degree of uncertainty: the higher the confidence value is, the less uncertain the correspondence is.*

For instance, possible alignment relations are binary relations between entities like inclusion, equivalence, disjointness for logical and set point of view. It can also be less strict like "near by", "close to", "related to",... The definition in [ES07] is in fact more general since it does not restrict itself to taxonomies and allows to declare correspondence between expressions of classes, relations and other entities of ontologies. The mappings that we consider are inclusion statements involving classes of two different taxonomies $\mathcal{T}_1$ and $\mathcal{T}_2$.

Definition II.6 (Mapping):
Given two taxonomies T_1 and T_2, a mapping between T_1 and T_2 is a correspondence (id, C_1, C_2, r, n) between T_1 and T_2 such that:

- *$r \in \{\sqsubseteq, \sqsupseteq\}$*
- *$n \in [0, 1]$ represents the confidence value of the mapping*

A mapping between C_1 and C_2 is denoted by $C_1 \sqsubseteq C_2$ or $C_2 \sqsubseteq C_1$, depending on the direction of the entailment relation between them.

Definition II.7 (Mapping alignment):
Given two taxonomies T_1 and T_2, a mapping alignment *is made up of a set of mappings between pairs of classes belonging to V_1 and V_2 respectively.*

From now on, an alignment actually refers to a mapping alignment, exclusively made of mappings of the form $A_1 \sqsubseteq B_2$ or $A_2 \sqsubseteq B_1$ where A_1 and B_1 denote classes of $\mathcal{T}_1$ and A_2 and B_2 denote classes of $\mathcal{T}_2$. For a mapping m of the form $A_i \sqsubseteq B_j$, its left-hand side A_i is denoted $lhs(m)$ and its right-hand side is denoted $rhs(m)$.

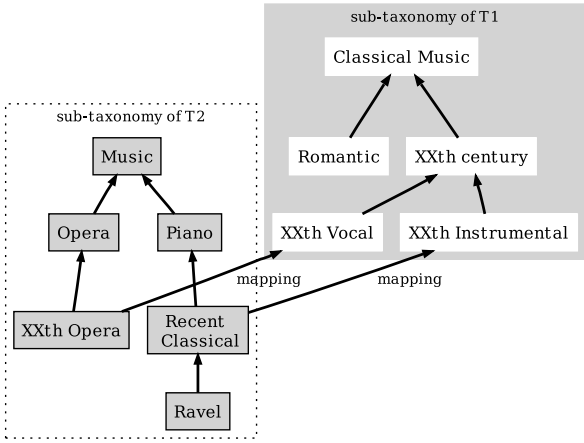


Figure II.2: 2 mappings between $\mathcal{T}_1$ and $\mathcal{T}_2$

The logical entailment between classes extends to logical entailment between mappings as follows.

Definition II.8 (Logical entailment between mappings):
Let $\mathcal{T}_i$ and $\mathcal{T}_j$ be two taxonomies. Let m and m' be two mappings between $\mathcal{T}_i$ and $\mathcal{T}_j$: m entails m' (denoted $m \preceq m'$) iff every model of $\mathcal{T}_i$, $\mathcal{T}_j$ and m is also a model of m', that can be formally written as follows: $\mathcal{T}_i, \mathcal{T}_j, m \models m'$.

It is straightforward to show that $\preceq$ is a (partial) order relation on the set $\mathcal{M}(\mathcal{T}_i, \mathcal{T}_j)$ of mappings between the two taxonomies $\mathcal{T}_i$ and $\mathcal{T}_j$. If $m \preceq m'$, we say that m is more specific than m' (also that m entails m') and that m' is more general than m (also that m' is a consequence of m). Note that we use either "entailment" or "implication" to denote an entailment between two mappings. The following proposition characterizes the logical entailment between mappings in function of the logical entailment between the classes of their left hand sides and right hand sides.

Proposition II.1:
Let $m = E_1 \sqsubseteq F_2$ be a mapping between $\mathcal{T}_1$ and $\mathcal{T}_2$. Let m' be a mapping between $\mathcal{T}_1$ and $\mathcal{T}_2$: $m \preceq m'$ (i. e. $\mathcal{T}_1, \mathcal{T}_2, m \models m'$) iff

(1) m' is of the form $E'_1 \sqsubseteq F'_2$, and

(2) $\mathcal{T}_1 \models E'_1 \sqsubseteq E_1$ and $\mathcal{T}_2 \models F_2 \sqsubseteq F'_2$

Proof :

($\Leftarrow$)
Let us assume (1) and (2), that correspond to:

1. $\mathcal{T}_i, \mathcal{T}_j, m \models E'_1 \sqsubseteq E_1$
2. $\mathcal{T}_i, \mathcal{T}_j, m \models E_1 \sqsubseteq F_2\,(= m)$
3. $\mathcal{T}_i, \mathcal{T}_j, m \models F_2 \sqsubseteq F'_2$

By transitivity of the specialization relation between classes, we obtain:

$$\mathcal{T}_i, \mathcal{T}_j, m \models E'_1 \sqsubseteq F'_2$$

and then

$$\mathcal{T}_i, \mathcal{T}_j, m \models m'$$

($\Rightarrow$)
To prove (1), we show that for every mapping of the form $F'_2 \sqsubseteq E'_1$,
$\mathcal{T}_1, \mathcal{T}_2, E_1 \sqsubseteq E_2 \not\models F'_2 \sqsubseteq E'_1$.

We start with an interpretation I_1 of the classes of $\mathcal{T}_1$ such that all its classes are empty.

I_1 is a model of $\mathcal{T}_1$ (since all the axioms $E_1 \sqsubseteq E'_1$ are satisfied in I_1).

We extend it by an interpretation I_2 of the classes of T_2 which is a model of $\mathcal{T}_2$ such that $F_2'^{I_2} \neq \emptyset$. Such a model I_2 of $\mathcal{T}_2$ exists by Lemma II.α.

Let $I = I_1 \cup I_2$, i.e. the interpretation defined by taking as domain of interpretation $\Delta^I = \Delta^{I_1} \cup \Delta^{I_2}$ and by defining:

- $C_1^I = C_1^{I_1}$ for each class C_1 of $\mathcal{T}_1$
- $C_2^I = C_2^{I_2}$ for each class C_2 of $\mathcal{T}_2$

By construction, I is a model of $\mathcal{T}_1 \cup \mathcal{T}_2 \cup E_1 \sqsubseteq F_2$ but I is *not* a model of $F'_2 \sqsubseteq E'_1$ (since $F_2'^I \neq \emptyset$ and $E_1'^I = \emptyset$).

To prove (2), we now show that if $\mathcal{T}_1 \not\models E'_1 \sqsubseteq E_1$ (resp. if $\mathcal{T}_2 \not\models F_2 \sqsubseteq F'_2$) then $\mathcal{T}_1, \mathcal{T}_2, E_1 \sqsubseteq F_2 \not\models E'_1 \sqsubseteq F'_2$.

By Lemma II.α, there exists a model I_1 of $\mathcal{T}_1$ such that $E_1'^{I_1} \neq \emptyset$.

By Lemma II.β, we can get it from I'_1 such that $E_1'^{I'_1} \neq \emptyset$ and $E_1^{I'_1} = \emptyset$ (because $\mathcal{T}_1 \not\models E'_1 \sqsubseteq E_1$).

Moreover, there exists a model I_2 of $\mathcal{T}_2$ such that $F_2'^{I_2} = \emptyset$.

Let $I = I_1 \cup I_2$. I is a model of $\mathcal{T}_1, \mathcal{T}_2, E_1 \sqsubseteq F_2$ and is not a model of $E'_1 \sqsubseteq F'_2$ because $E_1'^I \neq \emptyset$ and $F_2'^I = \emptyset$.

$\square$

Lemma II.α:
Let $\mathcal{T}$ be a taxonomy. For every class C of $\mathcal{T}$, there exists a model I of $\mathcal{T}$ such that $C^I \neq \emptyset$.

Proof :

We use the fact that the logical semantics of an inclusion axiom A $\sqsubseteq$ B is $\forall x A(x) \Rightarrow B(x)$, which corresponds to the binary clause $\neg A(x) \vee B(x)$.

Let us suppose that there exists a class C such that for every model I of T : $C^I = \emptyset$. This would mean that : $\mathcal{T} \models \forall x \neg C(x)$. By the completeness of the resolution for deriving prime implicates of a set of clauses, this would mean that the clause $\neg C(x)$ would be produced by applying the resolution rule to the set of clauses $\neg A(x) \vee B(x)$ corresponding the clausal form of the inclusion axioms A $\sqsubseteq$ B of the taxonomy.

It can be shown by induction on the number of applications of the resolution rule that from a set of binary clauses of the form $\neg A(x) \vee B(x)$ (i.e., with one postive literal and one negative literal), we can produce by resolution prime implicates that are necessary binary clauses having the same form (i.e., with one postive literal and one negative literal).

Therefore, the clause $\neg C(x)$ cannot be derived, which contradicts our assumption, and thus proves the Lemma.

□

Lemma II.β:
Let E'_1 a class of a taxonomy $\mathcal{T}_1$ and let I_1 a model of $\mathcal{T}_1$ such that $E_1'^{I_1} \neq \emptyset$.
Let I'_1 obtained from I_1 as follows:

- *for each C such that $\mathcal{T}_1 \not\models E'_1 \sqsubseteq C$, $C^{I'_1} = \emptyset$*
- *for each C such that $\mathcal{T}_1 \models E'_1 \sqsubseteq C$, $C^{I'_1} = C^{I_1}$*

I'_1 is a model of $\mathcal{T}_1$

Proof :

For any $C \sqsubseteq D \in \mathcal{T}_1$, let us show that $C^{I'_1} \subseteq D^{I'_1}$

- If $\mathcal{T}_1 \models E'_1 \sqsubseteq C$:
  since $C \sqsubseteq D \in \mathcal{T}_1$, $\mathcal{T}_1 \models E'_1 \sqsubseteq D$,
  and then as $C^{I'_1} = C^{I_1}$ and $D^{I'_1} = D_{I_1}$, we obtain: $C^{I'_1} \subseteq D^{I'_1}$.

- If $\mathcal{T}_1 \not\models E'_1 \sqsubseteq C$,
  then $C^{I'_1} = \emptyset$ and therefore $C^{I'_1} \subseteq D^{I'_1}$.

I'_1 is a model of $\mathcal{T}_1$

□

For example, two mappings between taxonomies $\mathcal{T}_1$ and $\mathcal{T}_2$ of Figure II.1 are illustrated in Figure II.2:

- the mapping $XXth\ Opera_2 \sqsubseteq XXth\ Vocal_1$ is more specific than the mapping $XXth\ Opera_2 \sqsubseteq XXth\ Century_1$,
- and the mapping $RecentClassical_2 \sqsubseteq XXth\ Instrumental_1$ is more specific than the mapping $Ravel_2 \sqsubseteq Classical\ Music_1$.

II.4 Probability measure

As we will provide a probabilistic model for mappings, we first recall some definitions about probability measure based on Kolmogorov axioms (e.g. [Gut05]).

Definition II.9 ($\sigma - algebra$, event):
Given a set Ω called the universe set, a $\sigma - algebra$ on Ω is a non-empty family $\mathcal{F}$ of subsets of Ω which satisfies the following conditions:

1. $\Omega \in \mathcal{F}$
2. $\forall A \in \mathcal{F}, \Omega \setminus A \in \mathcal{F}$
3. *$\mathcal{F}$ is closed under infinite union:*
   for every sequence $(A_n)_{n\in\mathbb{N}}$ of elements of $\mathcal{F}$, $\bigcup_{n\in\mathbb{N}} A_n \in \mathcal{F}$

An elements of $\mathcal{F}$ is called an event.

Definition II.10 (Measurable space):
Given a set Ω and a $\sigma - algebra$ $\mathcal{F}$ on Ω, the couple $(\Omega, \mathcal{F})$ is called a measurable space.

Definition II.11 (Probability measure):
Given a universe Ω, a probability measure *on Ω is a function $P : \mathcal{F} \longrightarrow \mathbb{R}$ which satisfies the Kolmogorov axioms:*

1. $P(\emptyset) = 0$
2. $\forall (A_n)_{n\in\mathbb{N}}$ *sequence of events with A_n pairwise disjoint,* $P(\bigcup_{n\in\mathbb{N}} A_n) = \sum_{n\in\mathbb{N}} P(A_n)$
3. $P(\Omega) = 1$

The triple $(\Omega, \mathcal{F}, P)$ is called a probability space.

In our context the $\sigma - algebra$ on Ω is always the set $\mathcal{P}(\Omega)$ of subsets of Ω. Therefore the domain of introduced probability measures is constituted by all the subsets of a given set Ω.

Definition II.12 (Random variable, distribution, expected value, variance):
Given a probability space $(\Omega, \mathcal{F}, P)$*:*

- a random variable X *is a measurable*[1] *function* $X : \Omega \longrightarrow \mathbb{R}$. *If* $X^{-1}(\mathbb{R})$ *is a countable set,* X *is said to be* discrete, *otherwise it is said to be* continuous.
- *The* discrete distribution μ_X *followed by a discrete random variable* X *is the function which associates to each element* y *of the image* $X(\Omega)$ *the probability of its inverse image* $\forall y \in X(\Omega)\,; \mu_X(y) = P(X^{-1}(y))$ *which is commonly denoted* $P(X = y)$
  The sum of the distribution values is 1: $\sum_{y \in E} \mu_X(y) = 1$
- *The* distribution function F_X *followed by a random variable* X *is the function* $F_X : \mathbb{R} \longrightarrow [0; 1]$ *where* $F_X(y) = P(X^{-1}(]-\infty, y[))$ *commonly denoted* $P(X < y)$.
  When there exists an integrable and positive or null function f_X *such that* $F_X(y) = \int_{-\infty}^{y} f_X(t)dt$, f_X *is called the density function of* X. *Its integral on* $\mathbb{R}$ *equals 1.*
- *The* expected value $E(X)$ *of* X *is defined as follows :*
  - $E[X] = \sum_{i \in \mathbb{R}} y\mu_X(y)$ *if* X *is discrete of distribution* μ_X.
  - $E[X] = \int_{y \in \mathbb{R}} y f_X(y)$ *if* X *is continuous and has a density function*
- *The* variance $Var(X)$ *of* X *is defined as :* $Var(X) = E[(X - E[X])^2] = E[X^2] - E[X]^2$.
  The standard deviation $\sigma(X)$ *is then defined as* $\sigma(X) = \sqrt{Var(X)}$.

Some examples are given below:

Example II.1 (Indicator function of an event)
Given a probability space $(\Omega, \mathcal{F}, P)$ *and an event* A, *the indicator function* $\mathbb{1}_A : \Omega \longrightarrow \{0; 1\}$ *such that* $\mathbb{1}_A(\omega) = 1$ *when* $\omega \in A$ *and 0 otherwise, is a discrete random variable whose the expected value is* $P(A)$.

Example II.2 (Bernouilli distribution)
Let p *a real in* $[0; 1]$. *The Bernouilli distribution* $\mathcal{B}(p)$ *is the discrete probability distribution of the success and failure in a single yes/no experiment, each experiment having a probability success of* p. *Formally, a random variable* X *following* $\mathcal{B}(p)$ *has the following properties:*

- *the universe* Ω *is the set of the two possible issues of the experiment: yes and no.*
- X *associates to the yes experiment issue the image* 1 *and to the no issue the image* 0
- $P(X = 1) = \mu_X(1) = p$ *and* $P(X = 0) = \mu_X(0) = 1 - p$
- *expected value of* X *is the mean value we can expect from such an experiment :* $E[X] = p$.

Example II.3 (Binomial distribution)
Let n *be a natural and* p *a real in* $[0; 1]$. *The binomial distribution* $\mathcal{B}(n, p)$ *is the discrete probability distribution of the number of successes in a sequence of* n *independent yes/no experiments, each of which yields success with probability* p. *Formally, a random variable* X *following* $\mathcal{B}(n, p)$ *has the following properties:*

- *the universe Ω is the set of all possible sequences of n independent yes/no experiments (with a success probability p)*
- *X associates to each such a sequence of Ω the number of the yes results in the sequence, which lies in $\mathbb{N}$ and a fortiori in $\mathbb{R}$. Note that this number is bounded between 0 and n and can only takes integer values. That is why X is called* discrete, *and the binomial distribution is called a discrete distribution.*
- *Given $k \in [0 \dots n]$, $P(X = k) = \mu_X(k) = P(X^{-1}(y)) = C_n^k p^k (1-p)^{(n-k)}$*
- *The expected value of X: $E[X] = np$.*

The *Beta* distribution is a family of continuous distributions extensively used to model the distributions of the parameters p of Bernouilli and binomial distributions in Bayesian estimation (section II.5):

Example II.4 (*Beta* distribution)
The Beta distribution is a family of continuous distribution defined on $[0;1]$ and parameterized by two positive parameters denoted by α and β. The corresponding density function family is the following :

$$f(x;\alpha,\beta) = \frac{x^{\alpha-1}(1-x)^{\beta-1}}{\int_0^1 u^{\alpha-1}(1-u)^{\beta-1}du}$$

The denominator is a normalizing factor ensuring that $\int_0^1 f(x;\alpha,\beta)dx = 1$.

The corresponding expected value is $\frac{\alpha}{\alpha+\beta}$.

Definition II.13 (Conditional probability):
Given a probability space $(\Omega, \mathcal{F}, P)$ and $(A, B) \in \mathcal{F}^2$ the conditional probability *of A given B is denoted $P(A|B)$ and when $P(B) \neq 0$:*

$$P(A|B) = \frac{P(A \cap B)}{P(B)}$$

$P(A|B)$ is undefined when $P(B) = 0$.

Note that when $P(A|B) = P(A)$, the events A and B are said to be *independent*, and $P(A \cap B) = P(A)P(B)$.

Definition II.14 (Bayes rule):

Given a probability space $(\Omega, \mathcal{F}, P)$, $\forall (A, B) \in \mathcal{F}^2,\ /P(B) \neq 0,\ P(A|B) = \frac{P(B|A)P(A)}{P(B)}$

This rule is commonly used to reverse conditional probabilities and stands at the core of the principle of Bayesian estimation in statistics.

II.5 Bayesian estimation

Bayesian estimation [Deg04] aims at providing an estimate value for an unknown scalar θ value (for instance a parameter of a probability distribution) based on linked observations (for instance sample values of data following the corresponding probability distribution) by considering it as a random variable following a prior distribution. A prior distribution expresses the *a priori* knowledge and uncertainty on θ before the observation data. Bayesian estimation represents a popular alternative to the classical maximum likelihood estimation in statistics.

Suppose we want to estimate an unknown scalar θ which follows a prior distribution Π (information we can know about θ before collecting any other data). Let $\mathbf{X}$ be the observations considered as a random variable, for instance taking its values in $\mathbb{R}^m$ with m fixed, and following any distribution. An estimator function of θ $\mathbf{x} \longrightarrow \hat{\delta}(\mathbf{x})$ provides a single estimate value for θ given an observation $\mathbf{x}$.

The Bayesian estimator of θ is the estimator function $\hat{\theta}$ which minimizes the Bayes risk for any given observation $\mathbf{x}$, i.e. :

$$\hat{\theta}(\mathbf{x}) = argmin_{\delta_{\mathbf{x}}} E[L(\theta, \delta_x)|\mathbf{X} = \mathbf{x}]$$

where $L : \mathbb{R}^2 \longrightarrow \mathbb{R}$ is a given loss function. The expected value is taken over the distribution of Π (the prior distribution of θ).

Usually the loss function is the squared error function and then the Bayes risk corresponding to a particular observation $\mathbf{x}$ is the following: $E[(\delta_{\mathbf{x}} - \theta)^2|\mathbf{X} = \mathbf{x}]$. In this case, the Bayesian estimator is given by the expression:

$$\hat{\theta}(\mathbf{x}) = E[\theta|\mathbf{X} = \mathbf{x}] \ \ (\mathbf{BE})$$

The Bayes rule is the main point of the proof of this theorem. Bayesian estimator is often applied to an unknown parameter of a distribution d by making it a random variable following a prior distribution Π. A convenient way to use this estimation, when we can choose it, is to take for Π the *conjugate* distribution d' of d. By doing this, the distributions of θ and $\theta|\mathbf{X} = \mathbf{x}$ are of the same algebraic form, and often of the same family of function with different parameters. That leads to make the calculus of the (**BE**) formula easier. In this thesis, we exploit the fact that the conjugate of a Bernouilli distribution is a *Beta* distribution.

II.6 Problem statement

As mentioned in the introduction, uncertainty is intrinsic to mapping discovery, because of the heterogeneity of taxonomies. We advocate that mappings need to be modeled using the grounded probability theory, in order to associate them with some probability values measuring their reliability. Proposing such models relies on defining how mappings and involved classes are measured as events or conditionals within a measurable space.

Such a probabilistic model cannot be independent of the logical semantics. In particular, it is expected that a mapping logically entailed by another mapping with a high probability (i.e.,

whose probability exceed a threshold) also get a high probability. Therefore, the first problem is to propose some models of probability for mappings, leading to probabilistic mappings. The probabilities associated to a mapping should be provided a practical way to compute them.

The second problem addressed in this work is the design of a scalable algorithm discovering the most probable mappings between two taxonomies.

Note that the problem of aligning n taxonomy can be reduced to a sequence of $n-1$ problems of aligning two taxonomies, each step allowing to group two aligned taxonomies in a single one.

CHAPTER

THREE

STATE-OF-THE-ART

As outlined in the introduction, semantic correspondences are the glue for data integration systems. A wide range of methods of schema/ontology matching have been developed both in the database [RB01] and the semantic web communities as shown in the book [ES07].

We start here by giving an overview of basis techniques, then we describe several methods which combine them. We focus especially on methods close to our work: instance-based methods and works about uncertainty handling for correspondences. Finally, we shall conclude by pointing out the important remaining challenges in the ontology matching domain which are tackled by this work.

Note that in many methods, the term "mapping" is used for denoting a correspondence. Other methods call mapping an alignment.

III.1 Overview of existing methods for ontology matching

Most of the time, ontology matching methods are combinations of atomic techniques that can be split in four categories: lexical or linguistic techniques, structural techniques, instance-based techniques, and reasoning techniques. In the following we give details for each of these categories and provide examples of actual methods using them.

III.1.1 String based and language-based techniques

Roughly speaking, string-based and linguistic-based techniques find correspondences between textual entities descriptions and labels. Many existing methods use them such as TaxoMap [HSNR09], H-MATCH [CFM03].

String-based techniques match entities labels or descriptions in a syntactic way. The underlying idea for matching entities with their names is that the more similar are the names (according to a chosen measure), the more there are likely to denote the same concepts.

Similarities on two strings can be computed according to the size of the largest common prefix or suffix. Edit distances like Levenshtein distance [Lev66] associate to two strings the minimum number of elementar operations (e.g. char insertion, char deletion, char substitution) to change the first string into the second one.

Another techniques based on string labels and entity descriptions use linguistic tools or resources. For instance, before the string-based comparison, words can be normalized (process of digits, punctuation, case, diacritics,...), lemmatized, stemmized using specific algorithms or dictionaries.

Linguistic resources like the WordNet thesaurus [C.98] allow to bridge the gap between a syntactic information and its meaning by giving all the senses (called synsets) of a given word or a phrase, and all the words for one given sense. In addition, it provides a directed relation graph between the synsets that represents the semantic relations between synsets (e.g. "Hyponym" for a subconcept relation, "Hypernym" for a superconcept relation, "Antonym" for an opposite meaning). These graphs represent a language-based way for computing distances or similarities between two given labels. A survey on using WordNet for ontology matching can be consulted in [LS08].

The H-MATCH [CFM03] matching method exploits WordNet for computing similarities between entities of two ontologies. In H-MATCH, the *Linguistic Affinity* (LA) between two labels of entities is computed as the highest-strength path joining two synsets of the two entities in the WordNet graph (restricted on a subset of the more classical semantic relations). The strength of the path is computed by multiplying the weight of each kind of semantic relations, so the best paths are the shortest or those involving the strongest relations.

But H-MATH uses another principle to compute the similarity between two entities: it uses the context of the entities, i.e. the sets of entity labels that are connected to the matched entity by a relation in the ontology. For instance, (c_1, r_1) is an element of the context of e_1 if e_1 is related to c_1 with r_1 in its ontology, and (c_2, r_2) is an element of the context of e_2 for similar reasons. H-MATCH introduces an affinity measure between entities based on their context: the *Contextual Affinity* between two entity contexts combines all *Linguistic Affinity* (LA) between every possible elements pair of the context (e.g. $((c_1, r_1), (c_2, r_2))$ is a pair). LA between c_1 and c_2 contributes to the *Contextual Affinity* of e_1, e_2 with a weight depending on the similarity between the respective r_1 and r_2, and so on for all possible pairs $((c_1, r_1), (c_2, r_2))$ of the contexts of e_1 and e_2. This principle involves the structure of the ontologies for computing a similarity between two entities. This is an example of the techniques based on external structural matching that we are going to see in the next section.

III.1.2 Structural matching techniques

External structure matching is based on the fact that the similarity between two entities on two respective graphs impact the similarities between the respective connected entities in each graph.

The work in [MGR+02] called *Similarity flooding* is based on this principle and uses a fix-point algorithm to affect to each pair of node a similarity, given two DAGs as input. The idea of "two elements of two distinct models (ontologies) are similar when their adjacent elements are similar" is applied by spreading similarities in the spirit of how IP packets flood the network

in broadcast communication. The core of the method relies on a similarity propagation graph: nodes are pairs of entities of the input ontologies, and directed edges between two nodes exist if the two elements of the first pair are connected to the two elements of the second pair with the same relation in the two respective ontologies. For each existing edge, a reverse (backward) edge is added to this graph with the idea that the similarity should be influenced in the two directions when there are relations between matched entities. Then, an initial similar value is affected to each node of the similarity propagation graph, for instance by using a string-based distance between the two labels in each node. After that, the fix-point computation is launched and all similarities on each nodes is updated according to the similarities of their neighbours and the weight fixed in the corresponding edges. A general form of the equation for a given pair (x, y) of entities to match is the following:

$$\sigma^{i+1}(x,y) = \sigma^{i}(x,y) + \sum_{(a,b) \text{ predecessors of } (x,y)} \sigma^{i}(a,b).w((x,y),(a,b))$$

$$+ \sum_{(a,b) \text{ successors of } (x,y)} \sigma^{i}(a,b).w((x,y),(a,b))$$

where $\sigma^i(x, y)$ represents the similarity between x and y at the step i, and $w((t, u), (v, z))$ the influence weight for the edge between the pair (t, u) and the pair (v, z).

When the fix-point is reached, filters are used for determining which are the best correspondences to be returned, given all similarities between pairs of entities and given some enabled constraints of cardinality (e.g. at least one correspondence from each class), and possibly other expressed constraints coming from the specific application of the matching.

The OLA (OWL-Lite Alignment) [EV04] method explicitly fits the grammar of the standardized ontology language OWL for defining similarities between entities. OLA establishes a family of similarity measures, one per node category (class, property, instance). The mutual dependencies between similarities reflects the OWL grammar: classes similarity involves the related properties similarities, the related classes similarities, and the related instances similarities. Dependency in equations of similarities can be approximatively solved by an iterative method, providing for each pair of entities a similarity value. By applying a threshold and/or using some constraints, these similarities can lead to an alignment.

The OMEN [MNJ05] method can be used for post-process the result of ontology matching methods that associate a confidence coefficient to each correspondence. This method is based on a Bayesian network where nodes are all possible 1-1 correspondences and where edges comes from meta-rules induced by the ontologies knowledge. For example, a meta-rule is "if there exist correspondences between all but one siblings of two superclasses that are also matched, then the probability for the correspondence between the two remaning siblings is high". Each node (i.e. each pair of classes) is associated with a distribution over the set of relations $\{=, \subset, \supset, \cap, \text{not related}\}$. Distributions for root nodes are initialized by the output of another matching method (it can be incomplete) and they are updated by reasoning on the Bayes Net. It is underlain by the same idea than for similarity flooding.

At the opposite side of structural matching techniques, the internal structural matching techniques exploit constraints and properties associated to entities like types, cardinalities or multiplicities and keys (for relational schemas). For example, the SemInt method [LC00] which was

originally intended to match relational schema works with an utilization of metadata present in database for each attribute (i.e. each class in an ontology framework): primary and external keys, data type for each attribute, cardinality constraints, data content statistics.

III.1.3 Instance-based matching techniques

Instance-based matching techniques are based on the analysis of statistics or distributions of class extensions. They can also rely on properties or descriptions of instances. Instances analysis can be exploited to compute similarities score between classes or to train classifiers for machine learning methods. As our work is instance-based, we provide a focus on such methods and particularly detail three of them, because they consider the closest probabilistic framework.

We start by presenting the general work on instance-based methods in [IvdMSW07] titled "An empirical study of instance-based ontology matching". It proposes a systematic study of three major dimensions involved when matching two classes:

1. how to measure the overlap
2. which thresholds are to be used
3. if hierarchy should been taken into account for calculating the extensions

A conclusion of this work is that the instance-based methods are reliable especially for practical and critical applications. Authors underline the critical choice of the measure depending on the application and on the kind of correspondences to discover. However they do not focus on oriented correspondences. In particular, they show that using the hierarchy for computing class extensions does not improve the quality of the alignment on the dataset used in this work. This surprisingly negative result comes directly from the structure of the dataset and from the fact that they always include equivalent correspondences in their evaluation.

We now describe representative and real-world instance-based methods. In contrast to the previous sections on other elementary techniques, we directly consider real-world methods that are mainly instance-based here. In particular we detail three of them:

1. the first one called GLUE [DMDH02] uses machine learning and standard similarities
2. the second one [DGGB06] is based on probabilities and logical constraints
3. the third one called SBI [ITH03] is based on probabilities and statistical test

Note that the experimental results of our work is compared to SBI in term of the accuracy of the returned alignments on Web directories (see Section VII.2).

GLUE

The GLUE [DMDH02] system automatically discover 1-1 correspondences between the classes of ontologies according to the assumption that there exists a moderate amount of available instances associated to each class. The classes are modeled as the set of instances of their extension, taken from a finite universe of instance. For two classes C_1 and C_2 of two ontologies $\mathcal{O}_1$ and $\mathcal{O}_2$, the

whole distribution is given by the four probabilities $P(C_1 \cap C_2), P(\overline{C_1} \cap C_2), P(C_1, \overline{C_2}), P(\overline{C_1}, \overline{C_2})$. GLUE is able to work on similarities that come from probabilistic interpretations. For example, consider the two following similarities for each pair of classes C_1, C_2:

- the Jaccard similarity [TSK06] between two classes C_1 and C_2:

$$J(C_1, C_2) = \frac{P(C_1 \cap C_2)}{P(C_1 \cup C_2)}$$

- the Most-specific-parent (MSP) similarity measure:

$$MSP(C_1, C_2) = P(C_1|C_2) \ if \ P(C_2|C_1) = 1 \ and \ 0 \ otherwise$$

The core of the GLUE system is divided into three steps:

1. Estimating the joint distribution of all pairs of classes and complementary classes
2. Computing the similarity matrix for the two ontologies from the joint distributions of classes, i.e. computing the similarity for each pair of classes of the two respective ontologies to be aligned
3. Affecting to the classes of $\mathcal{O}_2$ the class labels of $\mathcal{O}_1$ and conversely.

Now we detail the first step involving classification techniques. As the set of instances of the two ontologies are likely to be disjoint or with a very small overlap, the joint distributions can not be estimated directly by counting instances belonging to the corresponding intersections. Based on the general assumption that all extensions of classes are representative of the respective classes in the covered universe of instances, $P(C_1 \cap C_2)$ is estimated by the frequency-based formula :

$$\frac{N_1^{C_1 \cap C_2} + N_2^{C_1 \cap C_2}}{N_1 + N_2}$$

where N_1 and N_2 are the respective number of instances in $\mathcal{O}_1$ and $\mathcal{O}_2$, and $N_1^{C_1 \cap C_2}$, $N_2^{C_1 \cap C_2}$ are the cardinal of the sets of instances of $\mathcal{O}_1$ that belongs to both C_1 and C_2, in $\mathcal{O}_1$ and $\mathcal{O}_2$.

As C_1 is not a class of $\mathcal{O}_2$ and C_2 is not a class of $\mathcal{O}_1$, there is no instance of $\mathcal{O}_2$ that belongs to C_1 and no instance of $\mathcal{O}_1$ that belongs to C_2.

In order to overcome that, GLUE makes use of classifiers like Naive Bayes that exploit the textual data located in instances as features. A classifier is trained for C_1 with the training example the instances of C_1 in $\mathcal{O}_1$ and as counter-examples all the other instances in $\mathcal{O}_1$. Then, the instances of C_2 are classified in C_1 or in $\overline{C_1}$. The reverse process is done for classifying instances of C_2 into C_1 or $\overline{C}_1$. After that, the joint probabilities can be estimated by the above formula, and then the similarities.

Note that in GLUE, there is not a single classifier used but actually two classifiers combined by a meta-learner for weighting their results. The *content* classifier uses the textual content of instances (textual description considered as a set of normalized and stemmized tokens). The *name* classifier uses the full name of instances, i.e. the concatenation of all the labels of all instances it belongs to (including inferred) plus its proper name. The weights of the combination can be learned automatically by using stacking techniques derived from cross-validation techniques.

The third stage of GLUE is called a "relaxation labeling", which consists to affect the labels of the classes of $\mathcal{O}_1$ to all the classes of $\mathcal{O}_2$, given a set of constraints. This is based on the same idea of similarity flooding, i.e. that the label of a node is influenced by the feature of the node's neighborhood in the graph: labels, percentage of nodes in the neighborhood that satisfy a certain criterion, satisfaction of a constraint, etc. All probabilities that each class can labeled by such label are updated with an iterative way involving complex equations depending on the influence of all neighborhood's features. Example constraints can be the fact that "two nodes match if all their children match", or "if all children of node X match node Y, then X match node Y", or domain-dependent constraints like "There can be at most one node that matches DEPARTMENT CHAIR". Experiments on directories show that the relaxation labeling step improves results.

It should be pointed out that GLUE discovers symmetric correspondences that have no formal semantic. In addition, the associated confidence value for each discovered correspondence are not real probabilities as they are based on similarities like the Jaccard similarity. These two points contrast to our work in which we define a formal semantic for inclusion correspondences (denoted by mappings in this thesis) and we handle probabilities associated to each mappings.

A method based on Implication Intensity

The work presented in [DGGB06] is also based on extensions of classes. For a given set of instances of size n, and two classes a and b of respective extension sizes n_a and n_b, The implication intensity associated to the implication rule $a \rightarrow b$ is defined by:

$$\phi(a \rightarrow b) = 1 - P(N_{a \wedge \bar{b}} \leq n_{a \wedge \bar{b}})$$

where $n_{a \wedge \bar{b}}$ is the number of instances that contradicts the implication rule and $N_{a \wedge \bar{b}}$ denotes the number of instances that contradicts a random implication rule constructed with two random classes A and B of the same size than a and b. In this work, the underlying random variable for $N_{a \wedge \bar{b}}$ is modeled by a Poisson distribution of parameter $\lambda = \frac{n_a(n-n_b)}{n}$.

Actually the set of instances used for the computation of the implication intensity is not the initial set of instances for each class, but the set of relevant terms for each class. Such a transformation from the set of instances to the set of terms is done by using a threshold on the implication intensity of rules of the form $t \rightarrow i$ between instances i and terms t for each document.

Then, the extraction of mappings from the set of all possible rules with their associated implication intensity is done by:

- filtering the intensity of relevant rules by a threshold
- using a filter that transform a rule $a \rightarrow b$ into a mapping $A \sqsubseteq B$ only if all the logical implicant correspondences correspond to a rule having an intensity lower or equal than the intensity of the considered rule.
- using a top-down algorithm that exploits the above monotony-based filter to avoid computing all the possible intensity implications corresponding to each possible mapping (and

then for each possible rule), according to the fact that for each relevant mapping, every mapping consequence of it should be relevant.

This monotony-based filter should be underlined here because such a property of monotony of the strength of the correspondence according to the logical implication is at the core of the work in this thesis: the probabilistic confidence functions that we analyse for measuring the strength of mappings directly respect such a monotony property.

SBI

SBI [ITH03, IHT04] is intended to match internet directories (we see as taxonomies) by statistics on instances. It uses the same probabilistic framework as GLUE for classes and instances, and is based on the kappa-statistic Fleiss coefficient[KCRF+04] that measures the degree of agreement between two raters (that give a score of 0 and 1), each of them corresponding to a matched class. Each rater give the score 1 for each instance that belongs to the class, and 0 otherwise. The more the raters agree, the more the classes can be considered as equivalent.

Given two matched classes C_1 and C_2, the formula for the corresponding Fleiss kappa coefficient is the following:

$$\kappa = \frac{P - P'}{1 - P'}$$

where,

- P is their probability of coincidence of the two matched classes i.e.

  $$P(C_1 \cap C_2) + P(\overline{C_1} \cap \overline{C_2})$$

  estimated by $\frac{N_1^{C_1 \cap C_2} + N_2^{C_1 \cap C_2}}{N}$ with N representing the size of common instances of both taxonomies. The estimation of P (by computing the intersection of C_1 and C_2 inside each of both taxonomies) is processed without classification in the original SBI method.

- P' is the probability of coincidence of the two classes by chance (depending only of their sizes and on the size of the common instances). This should be considered with regard to the random variable $N_{a \wedge \overline{b}}$ in the implication intensity definition (see previous section), which is used as a reference quantity that the probability should exceed for making the correspondence relevant.

For a tested correspondence between A and B, the Fleiss Kappa coefficient is statistically tested to be different from 0, that is interpreted as a success.

Possible correspondences between two classes are tested from the top classes to the leaves in the two taxonomies with a pruning principle. For instance, if the two top classes do not match, the algorithm terminates with no correspondences.

Then, in an integration purpose, for each class of the source taxonomy, only the correspondence with the highest kappa is stored. For classes that are not involved in a discovered correspondence, correspondences involving its direct parent are considered instead, and so on.

One should notice that the discovered correspondences are equivalences and the kappa coefficient is symmetric. But because of the process that enforce a unique correspondence by source classes, the results are not the same if one switches the two taxonomies given as input.

As the confidence of correspondences is measured with the kappa coefficient which is symmetric and can not been interpreted as a probability, this work differs from ours. In addition, correspondences discovered are not formally defined equivalence. Finally, due to the specific application of directory integration, the fact that an unique correspondence is needed for each source class make the returned alignment different if the two input taxonomies are switched, although the considered correspondences are symmetric.

Now we quickly sketch other interesting and relevant methods of this category, respectively using Formal Concept Analysis and Machine Learning on correspondences.

Formal Concept Analysis: FCA-merge [SM01]

FCA-merge considers two taxonomies that annotate two sets of textual documents. The two respective sets of instances come from these documents. This method considers a *formal context* indicating which document is about which class : the set of documents are objects, the set of classes are attributes and the binary relations between objects and attributes is taken from logical declarations in taxonomies or are discovered by linguistic techniques. The core of the process uses a Formal Concept Analysis [GWW05] to create a (pruned) lattice of classes of both taxonomies of the granularity degree of the two ontologies. Concepts that appears in this lattice are at least as specific as one formal concept of both taxonomies. The final step that derives a *merged ontology* from this concept lattice requires human interaction. Authors underline that are documents are to be relevant and respect a good coverage of the ontologies to be merged.

The particular distinguishing points of this work with regard to ours is that it is intended to merge two taxonomies and not to discover correspondences and that it is semi-automatic by requesting a systematic human interaction.

Matching by learning correspondences

The three following methods are based on machine learning requiring a corpus of true correspondences.

LSD [DDL00] is a semi-automatic method that takes as training data the textual features of instances preprocessed by a set of correspondences provided by the user. A set of learners (based on textual similarity measures and on word distribution (Naive Bayes)) are combined with provided domain-constraint knowledge e.g., no more than one correspondence related to this class, or “if a matches b, and a matches c, then b should match c”). This process leads to symmetric correspondences between classes of ontologies. A particularity of the LSD system if that it can learn from a corpus of previous alignment of other ontologies.

The work introduced in [WES08] uses classifiers directly on *correspondences*, by representing each correspondence in a vector space constructed from instances features. A training set of true correspondences should be provided. Then, for a tested correspondence between two classes

A and B, the similarities between (i) the instances of A, (ii) the instances of B, and (iii) the instances of all examples correspondences allow to give to the tested correspondence a position in the correspondence space. Hence the classifier can be used to determine if the tested correspondence is relevant or not, according to its position in the correspondence space and the learned examples.

In a similar way, CSR [SVV08] provides a classification-based learning of inclusion (and then oriented) correspondences between classes of ontologies. CSR is based on the same machine learning methods in the space of correspondences. The features used for classification are the class properties, the terms extracted from labels, the comments, the instances of classes. The training is done using for positive examples the pairs of classes (A, B) such that A entails B in each of the two input ontologies considered in isolation. Then, given a pair of classes representing a possible tested correspondence, the classifier determines if this pair belongs to the category of entailment or in its complementary. Additionally, CSR adopts a pruning strategy of the search space for avoiding to test each possible correspondence: if there is no correspondence between A and B, then there is no correspondence between A and the descendants of B. This heuristic corresponds to the weak property of monotony IV.1 that we introduce in the next chapter page 45. Authors notice that CSR is potentially improvable by exploiting more kinds of features.

III.1.4 Matching based on logical reasoning

A reasoning process can be used to perform ontology matching in several ways. Many methods incorporates a logical reasoning step to check the logical consistency of the candidate correspondences returned by a previous step, or to infer other correspondences from the previously discovered ones.

Ctx-Match [SBMZ03] reduces the problem of ontology matching to the SAT problem. Each class of taxonomy is encoded into a formula of description logic in which the class atoms are WordNet synsets corresponding to the label of the class filtered to be relevant according to the synsets of the labels of the *context* of the class in its taxonomy: i.e. its ancestors and its direct descendants. Furthermore, the logical formula of a class is enriched with the conjunction of all formula of its ancestors. For instance, a class labeled `Arizona` having a superclass labeled `Snake` and no subclass is associated to the formula :

```
[Arizona(, genus Arizona (glossy snake) ⊓ snake, serpent, ophidian (limbless
scaly elongate reptile; some are venomous)]
```

For each class of both taxonomies, a reasoning can be processed according to the relevant part of a background knowledge theory (for example WordNet) translated in the same description logic language. For instance, inferred inclusion and disjointness formulas are respectively translated into inclusion, equivalence or disjointness correspondences.

The work in [SMS02] is originally intended for integrating spatio-temporal databases and uses a logical reasoning process to check the discovered candidate correspondences, by encoding the input schemas and these correspondences in a description logic language. Then, reasoning services are used to check the consistency of the candidate correspondences. Description logic allows to handle more expressive knowledge than propositional logic that is used in Ctx-match (e.g., binary predicates).

The work in [SMS02] outlines an integration methodology for databases schemas applied to spatio-temporal data, in particular by using logical reasoning with a domain ontology dedicated to spatio-temporal features (MADS). Specified constraints about theoretical properties of the returned integrated schema (reversibility, kind of information loss (class/attribute, instance) are at the core of the method and impact the result.

It should be noticed that semantic techniques described here can not find the alignment by themselves, they need to start from an external resource or a predefined similarity. But there are very helpful when the logical completeness of the results is needed, because the provided result can be closed under logical implication.

III.1.5 Properties of matching methods

In addition to the use of (i) lexical or linguistic techniques, (ii) internal or external structural techniques, and (iii) instance-based techniques, we can point out several criteria for distinguishing ontology matching methods.

- Requirement of an external resource: for example, H-MATCH [CFM03], TaxoMap [HSNR09], S-Match [GSY04], Ctx-Match [BSZ03] rely on the WordNet thesaurus. Similarly, among logic-based knowledge descriptions, Cyc [MCWD06] and DOLCE [GGMO03] are two examples of common knowledge ontologies that may be useful for techniques based on semantic reasoning in the spirit of Ctx-Match [BSZ03] with WordNet. Up to our knowledge, there are no existing work using directly such expressive ontologies.

  Generally speaking, techniques using an external ontology (e.g. for a specific domain) may first align the two ontologies to match to the external one, then they can make some reasoning to return correspondences between the two initial ontologies by using the external ontology as an intermediate knowledge.

- Alignment reuse: some methods may need some manually or previously found alignments as a corpus of correspondences (like COMA [DR02], or the work in [MBDH05]), based on the idea that there are patterns in the alignment task. Close to that, LSD [DDL00] and the work in [WES08] are machine learning-based methods that train classifiers directly on a corpus of provided correspondences.

- The language of correspondences to discover, i.e. what are the kinds of entities that are matched (only classes, expression of classes, properties, relations), and what are the available logical connectors (negation, union, intersection, cardinality restriction). In particular, an important point is the possibility to handle directed correspondences: for example, ASMOV [JK07], CSR [SVV08], Ctx-Match [BSZ03] and TaxoMap [HSNR09] handle inclusion correspondences (i.e. mappings).

- Use of machine learning techniques (e.g., GLUE [DMDH02]), SBI using classifiers [IHT04], Enhanced-NB [AS01], LSD [DDL00], SemInt [LC00], the work in [WES08]). In addition, requirement of a corpus for training (e.g., instances, correspondences).

- Ability to provide a confidence coefficient for each returned correspondence. Most of existing methods do that by default, but Ctx-Match [BSZ03], FCA-merge [SM01] which

are respectively based on pure logical reasoning and Formal Concept Analysis [GWW05] can not, for example.

Other classifications of matching methods can be found in [BSZ03, DH05, Ehr07].

III.1.6 Illustrative examples of real methods

Almost of all real methods like those we present in this section combines several elementary techniques we have explained at the beginning of this section.

SemInt [LC00] uses internal structure matching techniques. Features associated to each class are properties of cardinality or any constraints, and string labels. For both ontology A, B to align, a clustering of classes is done by a self-organizing map (neural network algorithm) in the space of features. For each ontology, the returned clusters are characterized by their center and they represent a partition of the set of classes. For ontology A, the classes and their respective cluster centers are provided to a supervised classifier (back-propagation) as training data (clusters centers represent the labels). The classes of B are classified into the clusters of A, i.e. for each class of B, one can output a score for each possible corresponding class of A. After having doing a similar process by switching the two ontologies, this result is post-processed to obtain a set of correspondences between classes.

ASMOV [JK07] stands for "Automated Semantic correspondence of Ontologies with Validation". It combines lexical similarity (using an external thesaurus, UMLS), internal structure (properties, data types) features, external structure (ancestors and descendants), and similarity on instances by a weighted sum. The entities related by the best similarity score are mapped. Then ASMOV uses a process of semantic validation of this alignment, by checking the consistency of discovered correspondences with regard to the ontologies.

Lily [WX09] is a complex tool combining textual and structural techniques. A preprocessing step is firstly run in order to build a semantic description for all entities of the ontology. The matching process itself works on these descriptions with lexical similarity and similarity flooding on the structure. A distinguishing feature of this method is that it makes use of web search engines to address the problem of semantic heterogeneity. Like in ASMOV, a post-processing step is conducted to make the result more robust (detection inconsistency or redundancy, bad or abnormal correspondences based on their features).

Clio [CHKP07] is a semi-automatic schema integration system. It merges different schemas by discovering symmetric correspondences between classes (concepts) extracted from schemas. The process of matching starts with a cast of the schemas into taxonomies of classes. Classes that are matching are identified by correspondences between their attributes in the initial schema. Then all possible ways of aligning the two taxonomies of classes are enumerated, that correspond to an enumeration of all possible merged taxonomy. Each merged taxonomies an be recast-back as an integrated schema. Thus, all induced correspondences between initial schemas and each of enumerated integrated schema can be given. A refinement step and a user feed-back step are then used in order to visualise and select the more relevant alignment according to the user.

Coma++ [ADMR05] is a customizable and generic tool for matching schemas and ontologies. It combines several individual matchers and thresholds for several kind of similarities.

Cupid [MBR01]: a schema-based approach combining linguistic (using a thesaurus) methods with constraints (keys, data types, because it was firstly dedicated to schema matching). A distinguishing point to notice is that Cupid makes a special care on different granularities and nesting of the matched schema.

PRIOR [MP06] is a combination of linguistic and structural techniques by similarity flooding. Each class is represented by its corresponding profile. Then profiles are modified inductively by the profiles of the neighborhood. PRIOR correspondences based on the cosine similarity of profiles. It address the scalability issue: for large ontologies, the PRIOR system makes use of information retrieval techniques, by indexing the profiles of the first ontology, then generating queries from profiles of the second input ontology, and ranking the answers.

TaxoMap [HSNR09]: This method combines linguistic techniques and textual similarity with a complex system of thresholds to select correspondences. This process is based on similarity between labels or multi-labels, inclusion between labels, and reasoning on structure. It discovers oriented and symmetric correspondences between classes of taxonomies. In the case of large taxonomies, TaxoMap uses a partitioning technique based on a specific partitioning algorithm for ontologies. The main principle is to partition ontologies into blocks that are strongly related, and to match only the more related blocks, i.e. those that share enough labels. The classes that share the same labels are called anchors and are used in the process of partitioning.

Yet Another Matcher (YAM) [DCBM09] is a matcher factory that generates a dedicated matcher given application constraints specified by the users. For example, such requirements could be a preference for either Recall or Precision, a set of already matched schemas, or provided correspondences. YAM learns how to apply a pool of tools in the best way, according to these provided data and constraints, by meta-learning with the goal of maximizing the quality of the output. This approach constitutes an extreme and application-driven view of how to combine of a set of techniques.

Other interesting matching methods are presented in [GSY04, ST05, HQ08, LTLL08].

We should mention here the project **Linking Open Data**[1] [Biz09] which aims at providing an unified semantic dataset by publishing various connected open ontologies as RDF on the web. The main point of the project is that the RDF cross links between the different sources are correctly set to enable the connection between ontologies, and then it provides a large "Web of data". Ontologies from popular knowledge databases like FOAF[2], Wikipedia[3], WordNet [C.98] are among the 120 sources datasets. These datasets represent 13.1 billion of triples and are connected by 142 millions of triples (in November 2009). Data reconciliation and ontology matching are at the core of the achievement of this huge project.

III.2 Remaining challenges

Such a large amount of implemented methods proves that the matching task is not easy at all. Each of these methods fits with an application-specific use, a particular user-constraint

[1] `http://esw.w3.org/SweoIG/TaskForces/CommunityProjects/LinkingOpenData`
[2] www.foaf-project.org
[3] wikipedia.com

or requires a property on data. The properties of matching systems listed in Section III.1.5 confirm such an heterogeneity: there is still a lack of a global and generic method for ontology matching. Based on this observation, ten challenges for ontology matching are presented in [SE08] by taking into consideration the results of all the conducted OAEI contests.

III.2.1 Handling uncertainty of correspondences

The uncertainty is intrinsic to correspondences discovery because two classes or properties (for example) independently created are unlikely to exactly match. As stated in the introduction, there is still a need to better understand the foundations of modeling uncertainty that is primary important for improving the detection of correspondences causing inconsistencies, e.g., via probabilistic reasoning, or to identify where the user feedback is maximally useful, and for improving the quality of the interpretation of correspondences. The defined notion of correspondence (see Definition II.5) include a confidence coefficient to measure the uncertainty associated to each correspondence.

For example, it should be useful for a method that verifies the output of matching process in order to make it more robust, like the work presented in [CFL$^+$08]. This work validates correspondences by probabilistic reasoning, by associating to each correspondence its probability. All correspondences that are consistent with the ontologies in the probabilistic logic are considered valid, and the other ones are given up.

The work in [GATM05] introduces a framework for modeling and evaluating automatic semantic reconciliation. It provides a formal model for semantic reconciliation and theoretically analyses the factors that impact the effectiveness of matching algorithms. The proposed formal model borrows from fuzzy set theory for handling uncertainty by combinations of similarity measures and good properties. This work is not connected to probabilities but is complementary to the approach in this thesis.

Another works on uncertainty handling by probabilities can be read in [NS07] .

A complementary work that considers ranking of several possible alignments to make the resulting alignment more robust and to reduce the need of human verification is presented in [Gal06]: this work proposes an extension of existing matching methods by considering the k ranked best alignments between two schemas discovered by a classical matching method, according to a global score that can be computed from the output of many existing methods. The top-k alignment ranking is combined with the schema to match in order to generate the final alignment that globally maximizes its score. The idea behind this work is that there exists a correlation between patterns in the top-k alignments (i.e. set of correspondences) and the correctness of a considered alignment. This method allows to improve correspondence precision at the cost of recall with regard to existing methods, and to make a step further in making the process of matching completely automatic.

III.2.2 Scalability

The scalability of ontology matching represents a major issue for the achievement of the future Semantic Web, where ontologies sizes and numbers are expected to grow exponentially.

But the main context (OAEI) does not primely focus on scalability up to now, except for larges ontologies like anatomy [4] (2006) which involved a magnitude of 10,000 classes per ontologies to match. Methods like PRIOR [MP06] and H-MATCH [CFM06] perform well on such a dataset. Note that such a dataset does not contain any instances.

In the same time, the directory dataset is available from 2005 to now in two versions, one fully connected of 6628 and 2857 classes, and the other one split into 4000 pairs of corresponding branches to map. Since 2005, all participants use the split version that allows to match more than 4,000 times two taxonomies of about ten classes, instead of matching the whole structures. It can be concluded that the scope of scalability issue starts from ontologies of thousands of instances and more.

For instance-based methods, the number of classes is not sufficient enough to judge of the scalability: because the number of instances is likely to play a major role in the complexity of matching algorithms. A similar remark holds for methods that want to be complete, i.e. to consider Recall as important, because there are more likely to consider the entire search space.

For handling large taxonomy, a central idea is to significantly reduce the size of the possible correspondences by partitioning large ontologies. See for example TaxoMap as explained above and in [HSRZ09] by partitioning ontologies into blocks that are strongly related, and to match only the more related blocks.

Synthesis

As a conclusion, our work stands among instance-based matching methods and deals with an important theoretical aspect: modeling uncertainty for inclusion correspondences (i.e., mappings). Among the existing work in this category, and to the best of our knowledge, we can notice there are no methods which is concerned by defining a formal semantic for discovered correspondences. Moreover, the correspondences returned are often symmetric, standing for informal equivalence or high-similarities between two entities. Finally, if most of methods return an associated score measuring the confidence of each correspondence, no one can be interpreted as a probability. Indeed, for that, a correspondence should respect a property of monotony with regard the logical implication. The work in [DGGB06] enforces such a particular property that we further call "strong property of monotony" according to the logical implication: "each discovered correspondences should have a lower score than its logical consequences according to the taxonomies".

Finally our ambition in the setting of ontology matchingis is to bridge the gap between logic and probabilities by providing probabilistic models that are consistent with the logical semantics underlying ontology languages.

[4] `http://oaei.ontologymatching.org/2006/anatomy/`

Part I

Models and algorithm: combining logic and probability

CHAPTER

FOUR

PROBABILISTIC MODELS OF MAPPINGS

The intrinsic uncertainty of mappings due to multiple heterogeneity factors like those seen in introduction makes inevitable to take account into such uncertainty in semantics of mapping and especially during the mapping discovery process.

In the same time, keeping a connection to the logical implications for mapping semantics is useful for logical reasoning with them and for instance for query rewriting. For example, a realistic situation is that after the discovery process, discovered mappings are interpreted as certain mappings with classical logical semantic (i.e. *certain* means that the mapping is considered as fully true).

An uncertainty model of mappings should take into account possible logical implications between mappings, which can be inferred from the inclusion axioms declared between classes within each ontology. A good property should be that a mapping logically entailed by another mapping should be less uncertain that its entailer.

Theories that include uncertainty in logic exist and the most famous are probability logic and fuzzy logic. These domains allow to handle imprecision in logical formulas and reasoning by associating a real number between 0 and 1 to each formula.

Fuzzy logic [Zad65] represent vague and imprecise information, like the formula "this jean is large". The associated number correspond to the degree of which the jean is large and constitutes is degree of truth. 0.7 indicates that the jean is near to be large, but not very large. So the truth of a formula is gradual. In probability logic [FHM90, Nil86], the truth of formulas is binary like in classical logic: it can be only $True$ or $False$. The real number associated to each logical formula measures the probability for this formula to be *True*.

Because of the fact that probability logic extends the classical logic more naturally and that probability is the more grounded and studied way to express uncertainty, we aim at extending the logical semantic via a probabilistic way, by associating to each mapping the probability that it is logically true. Such an approach corresponds thoses used in *probability logic* and *probabilistic reasoning* domains. Probability logic should be distinguished from the *fuzzy logical* approach where the logical values themselves are degrees of truth. Here, we take given values

as they are exact but unknown, so we only consider the two classical logical constants $True$ and $False$.

We also advocate to consider *probabilistic models of mappings*. It means that mappings and their involved classes should be considered as measurable entities (like events for instance) in a probabilistic setting, with regard to a fixed measurable space.

We have considered two probabilistic models for modeling uncertain mappings. Indeed, we introduce here two confidence functions that associate to each mapping a confidence value n (see Definition II.6 page 16) of a mapping. Given two taxonomies $\mathcal{T}_1, \mathcal{T}_2$ and their respective vocabulary V_1 and V_2 we consider the following probabily space (definition II.11), $(\mathcal{D}, \mathcal{P}(\mathcal{D}), P)$ where:

- The universe $\mathcal{D}$ is the set of possible instances of T_1 and T_2
- The set of events $\mathcal{P}(\mathcal{D})$ is the set of all subsets of possible instances of $\mathcal{D}$
- P is a fixed probability measure on all subsets of instances

For every class E, we abusively denote E the event "a randomly chosen instance belongs to E". The Bernouilli (parameter p_E) corresponding random variable is the characteristic function $\mathbb{1}_E$ of E. $P(E)$ is then the probability of the class E, and $P(E) = p_E = E[\mathbb{1}_E]$.

This discrete random variable follows a Bernouilli distribution of parameter p_E such that $p_E = P(E) = E[X_E]$

Logical formulas and events are linked: the negative class $\neg E$ trivially corresponds to the event $\overline{E}$ which is the complementary set of the event E in the universe $\mathcal{D}$. Disjunction and intersection of class corresponds to disjunction and intersection of events. The Venn diagram picture a

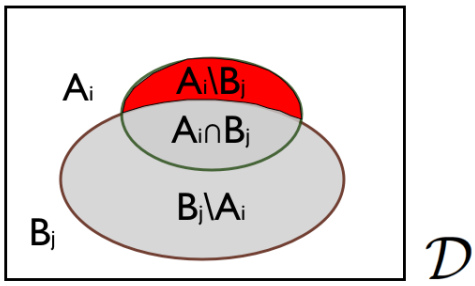


Figure IV.1: Venn diagrams of two class extensions

set interpretation of classes, or equivalently the events associated to them. In this work, we constantly work with an underlying set interpretation of classes, which is coherent with both logical and probabilistic theories.

IV.1 Conditional probabilistic model

The first model defines the probability of a mapping $A_i \sqsubseteq B_j$ as the conditional probability for an instance of the sample space to be an instance of B_j knowing that it is an instance of A_i. It

is the natural way to extend the logical semantics of entailment to probabilities.

Definition IV.1 (Conditional probability for a mapping):
Given a mapping $m = A_i \sqsubseteq B_j$, the conditional probability *of m is defined as:*

$$P_c(m) = P(B_j|A_i) = \frac{P(A_i \cap B_j)}{P(A_i)}$$

In the Venn diagram of Figure IV.1, that corresponds to restrict the $\mathcal{D}$ universe set to the A_i set. The function measure $P_{A_i}(X) = P(X|A_i)$ is a probability measure associating for each subset of $\mathcal{D}$ (each subset of instances) its probability given the knowledge that these instances all belongs to B_j. The general case pictured in Figure IV.1 illustrates the conditional probability by a Venn diagram, as the proportion of $A_i \cap B_j$ in A_i. In that Venn diagram, there is drawn a darkened region corresponding to $A_i \setminus B_j = \overline{B_j} \cap A_i$, in other words the region which lies in B_j but not in A_i. This region is the set of instances that contradict the mapping $A_i \sqsubseteq B_j$.

As $P(B_j|A_i) = 1 - \frac{P(\overline{B_j} \cap A_i)}{P(A_i)}$: the probability of this dark region make P_c decrease when it increases, but its influence is weighted by the probability of A_i.

IV.2 Implication probabilistic model

The second model comes directly from viewing classes as subsets of the sample space and the inclusion formula as the so-called "material" implication (i.e. denoting the classical logical implication): the implication probability of $A_i \sqsubseteq B_j$ is the probability for an element of the sample space to belong to the set $\overline{A_i} \cup B_j$.

Definition IV.2 (Implication probability for a mapping):
Given a mapping $m = A_i \sqsubseteq B_j$, the implication probability *of m is defined as:*

$$P_i(m) = P(\overline{A_i} \cup B_j)$$

Note that $P_i(A_i \sqsubseteq B_j) = P(\overline{A_i} \cup B_j) = 1 - P(A_i \cap \overline{B_j})$.

For this model, the probability of the dark region representing the set of instances that contradict the mapping in Figure IV.1 is a negative linear term in the probability $P_i(A_i \sqsubseteq B_j)$. Therefore in this model, the probability of $B_j \setminus A_i$ is considered in an absolute respect.

A particular point to emphasize is that the implication is truth-functional, i.e. its truth value is fully determined by the truth value of its operands. That is not the case of the value of $P(A_i|B_j)$ which is not depending of $P(A_i)$ and $P(B_j)$ only.

As both P_i and P_c provide *probabilities* as confidence values, there are called *probabilistic confidence functions* in the following.

IV.3 Comparative study of the two probabilistic models

We start by providing the main properties of the two probabilistic confidence functions in order to compare them, especially about a property of monotony with regard to the logical implication. Then, we introduce a result that characterizes the confidences functions which respect one particular form of this monotony property.

Main properties of the two probabilistic confidence functions

The following proposition states the main (comparative) properties of those two probabilistic models. In particular, they both meet the logical semantics for mappings that are certain.

They can both be equivalently expressed using the joint probabilities of the two mapped classes and the probability of the class at the left of the mapping.

To give some statistical properties on both mapping probabilities we consider P_i and P_c as random variable functions of two random class extensions (subsets of Δ).

Proposition IV.1 (Properties on confidence probabilistic functions):
Let m be a mapping between two taxonomies $\mathcal{T}_i$ and $\mathcal{T}_j$. The following properties hold:

1. $P_i(m) \geq P_c(m)$.
2. m *is a* certain mapping *(i.e., $\mathcal{T}_i\,\mathcal{T}_j \models m$):* $\Leftrightarrow P_c(m) \Leftrightarrow P_i(m) = 1$.
3. $P_i(m) = 1 + P(lhs(m) \cap rhs(m)) - P(lhs(m))$
4. $P(lhs(m)) = 0$ *or* $P(rhs(m)) = 1 \;\Rightarrow\; P_i(m) = 1$

The point (3) and the definition of the conditional probability for a mapping show that both probabilities can be expressed with only $P(lhs(m))$ and the joint probability $P(lhs(m) \cap rhs(m))$.

Proof Proof of (3):

By definition of the implication probability:

$$P_i(m) = P(\overline{lhs(m)} \cup rhs(m))$$

$$P_i(m) = 1 - P(lhs(m) \cap \overline{rhs(m)})$$

By disjunction we have $P(lhs(m)) = P(lhs(m) \cap rhs(m)) + P(lhs(m) \cap \overline{rhs(m)})$. Therefore, by combining this and the above expression of $P_i(m)$:

$$P_i(m) = 1 - [P(lhs(m)) - P(lhs(m) \cap rhs(m))]$$

$$P_i(m) = 1 - P(lhs(m)) + P(lhs(m) \cap rhs(m))$$

□

Proof Proof of (1) and (2):

Let $m = A_i \sqsubseteq B_j$.

$$P_c(m) - P_i(m) = \frac{P(A_i \cap B_j)}{P(A_i)} - 1 + P(A_i) - P(A_i \cap B_j)$$

$$= \frac{P(A_i \cap B_j)(1 - P(A_i)) - P(A_i)(1 - P(A_i)))}{P(A_i)}$$

$$= \underbrace{\frac{1 - P(A_i)}{P(A_i)}}_{>0} \underbrace{(P(A_i \cap B_j) - P(A_i))}_{\leq 0}$$

The above formula points out that $P_c(m) = 1$ if and only if $P_i(m) = 1$ and it arises if and only if $A_i \subseteq B_j$ (when $P(A_i) > 0$), or equivalently when $A_i \sqsubseteq B_j$ is certain.

□

Proof Proof of (4):

By application of the definition of $P_i(m) = P(\overline{lhs(m)} \cup rhs(m))$ therefore $P_i(m) \geq P(rhs(m))$ and $P_i(m) \geq 1 - P(lhs(m)))$.

So, $P(rhs(m)) = 1$ or $P(lhs(m)) = 0$ leads to $P_i(m) = 1$

□

The two probabilities differ on the monotony property with regard to the (partial) order $\preceq$ corresponding to logical implication (cf. Definition II.8): P_i verifies a property of monotony whereas P_c verifies a property of *weak* monotony as stated in the following theorem.

Theorem IV.1 (Property of monotony):
Let m and m' two mappings.

1. *If $m \preceq m'$ then $P_i(m) \leq P_i(m')$ (strong monotony)*
2. *If $m \preceq m'$ and $lhs(m) = lhs(m')$ then $P_c(m) \leq P_c(m')$ (weak monotony)*

Proof Proof of the theorem IV.1:

Let E_1, E'_1, F_2, F'_2 be classes of two taxonomies $\mathcal{T}_1$ and $\mathcal{T}_2$. Let $m = E_1 \sqsubseteq F_2$, $m' = E'_1 \sqsubseteq F'_2$ be two mappings such that $m \preceq m'$.

By the proposition II.1 page 17, we have that E_1 and E'_1 belong to $\mathcal{T}_1$ and F_2 and F'_2 belong to $\mathcal{T}_2$ and that $\mathcal{T}_1 \models E'_1 \sqsubseteq E_1$ and $\mathcal{T}_2 \models F_2 \sqsubseteq F'_2$. Specialization relations are certain and

correspond to proper set inclusions in the set interpretation of classes, therefore we have:

$$\begin{cases} P(E_1' \cap \overline{E_1}) = 0 \\ P(F_2 \cap \overline{F_2'}) = 0 \end{cases}$$

By disjunction of E_1 and $\overline{E_1}$:

$$P(E_1' \cap \overline{F_2}) = \underbrace{P(E_1' \cap \overline{F_2} \cap E_1)}_{\leq P(E_1 \cap \overline{F_2})} + \underbrace{P(E_1' \cap \overline{F_2} \cap \overline{E_1}}_{=P(E_1' \cap \overline{E_1})=0}$$

Therefore $P(E_1' \cap \overline{F_2}) \leq P(E_1 \cap \overline{F_2})$.

In the same way:

$$P(E_1' \cap \overline{F_2'}) = \underbrace{P(E_1' \cap \overline{F_2'} \cap F_2)}_{=P(\overline{F_2'} \cap F_2)=0} + \underbrace{P(E_1' \cap \overline{F_2'} \cap \overline{F_2})}_{\leq P(E_1' \cap \overline{F_2})}$$

Then $P(E_1' \cap \overline{F_2'}) \leq P(E_1' \cap \overline{F_2})$ and finally, if we combine it with the previous inequality:

$$P(E_1' \cap \overline{F_2'}) \leq P(E_1 \cap \overline{F_2})$$

As $P(A \cap B) = 1 - P(\overline{A} \cup \overline{B})$, it leads to :

$$P(\overline{E_1'} \cup F_2') \geq P(\overline{E_1} \cup F_2)$$

and

$$P_i(E_1 \sqsubseteq F_2) \leq P_i(E_1' \sqsubseteq F_2')$$

i.e.,

$$P_i(m) \leq P_i(m')$$

Proof for monotony of P_c where $lhs(m) = lhs(m')$:

Let $m = E_1 \sqsubseteq F_2$ and $m' = E_1' \sqsubseteq F_2'$ be two mappings such that $m \preceq m'$. By the proposition II.1 again, we have $\mathcal{T}_2 \models F_2 \sqsubseteq F_2'$.

$P_c(E_1 \sqsubseteq F_2) = \frac{P(E_1 \cap F_2)}{P(E_1)} \leq \frac{P(E_1) \cap F_2')}{P(E_1)} = P_c(E_1 \sqsubseteq F_2')$ because $F_2 \sqsubseteq F_2'$.
Therefore $P_c(m) \leq P_c(m')$.

□

Example IV.1 (Counter-example for strong monotony of P_c)
The Figure IV.2 shows a representative case for the non-monotony of P_c.

Here, $E_1 \sqsubseteq F_2$ entails $E_1' \sqsubseteq F_2$ but $P_c(E_1 \sqsubseteq F_2) > P_c(E_1' \sqsubseteq F_2) = 0$.

We can see here that P_c is not coherent with regard to the transitivity of the inclusion: $P_c(E_1' \sqsubseteq E_1)$ and $P_c(E_1 \sqsubseteq F_2)$ can grow up to infinity whereas $P_c(E_1' \sqsubseteq F_2)$ remains to 0.

A comparison of properties of the two probabilities is given below in order to provide an *a priori*

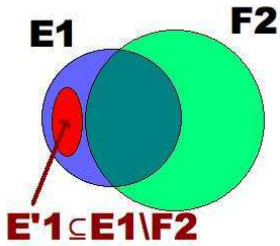


Figure IV.2: Case where $P_c(E_1' \sqsubseteq F_2) = 0$ whereas $P_c(E_1 \sqsubseteq F_2) > 0$ and $E_1' \sqsubseteq E_1$

behaviour and show that they are complementary to distinguish the "best" mappings from the other ones.

As we give here some statistical properties on both models, we consider P_i and P_c as random variable functions of pair of subsets of $\mathcal{D}$, assuming that for each random set S, each instance of $\mathcal{D}$ belongs to S with a probability of $\frac{1}{2}$.By symmetry, it leads to $P(A) = P(\overline{A})$ and to $P(A \cap B) = P(\overline{A} \cap B) = P(A \cap \overline{B}) = P(\overline{A} \cap \overline{B})$.
Both these constraints conduct to:

1. $E[P_i] = \frac{3}{4}$
2. $E[P_c] = \frac{1}{2}$

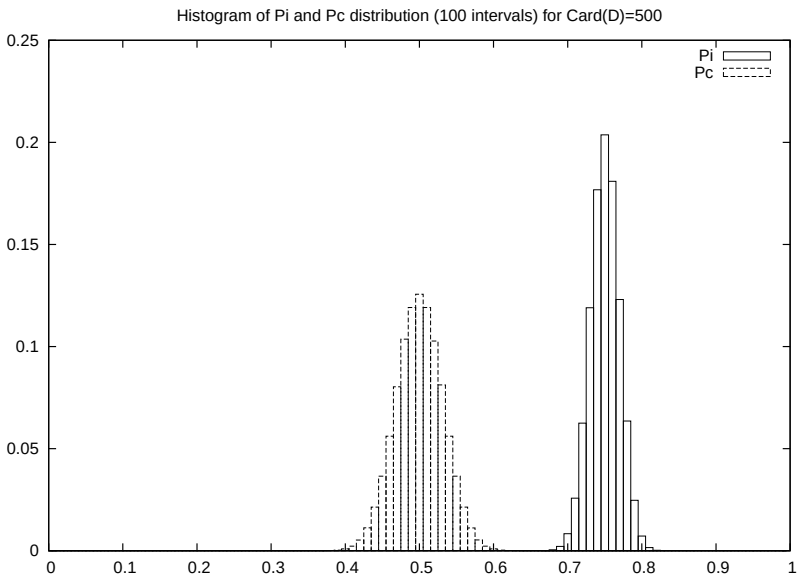


Figure IV.3: Distribution histograms for P_i and P_c

Thresholds for P_i and P_c can be used to distinguish between "good" mappings and "bad" mappings in the discovery process. It is the main approach considered here, rather than ranking or other method.

The Figure IV.3 is generated by using a huge sample of randomly chosen set pairs (A, B) in

a sample space of 500 instances, according to the assumption that a random set is constituted by potentially all instances with a probability of $\frac{1}{2}$ for each one. The distribution of obtained values for P_c and P_i are pictured in the histogram of this figure.

The first conclusion is that the threshold for P_i should be larger than the threshold for P_c. This is coherent with the property $P_c(m) \leq P_i(m)$ for every mapping m.

Secondly, as $P_c(m)$ appears to be more distributed than P_i, the conditional probability may be less sensitive to its corresponding threshold than the implication probability.

Thirdly, P_i is the model that fits the logic in the best way. It has also the same drawbacks as the logical implication itself: a class A_i with a very small probability or a class B_j with a probability close to 1 leads automatically to a implication probability P_i close to 1. In logic, this corresponds to the following cases which are true in propositional logic, although their respective truths are not sure actually in the common sense:

- "If Christophe Collomb did not discover America, I like him." : false antecedent and undetermined conclusion.
- "If I won the quizz , Christophe Collomb discovered America." : undetermined antecedent and true conclusion

Conversely, P_c has been defined as $P_c(A_i \sqsubseteq B_j) = 1 - \frac{P(A_i \setminus B_j)}{P(A_i)}$ which better handles such cases because the probability $P(A_i \setminus B_j)$ of the set of instances that contradict the mapping is taken into account with respect to the size of A_i. Conditional probability measure the B_j probability inside the world in which A_i is true. So no matter that B_j is very probable and no matter the size of A_i for $P_c(A_i \sqsubseteq B_j)$. Making a parallel with logical sentences of implication, it is equivalent to consider their truth only in the cases where the antecedent is true.

But implication probability may be useful anyway. Figure IV.4 shows the P_i distribution values histogram on all mappings m for which $P_c(m) > 0.7$ (with the same generating condition as the Figure IV.3).

Note that 0.7 is an arbitrary but potentially realistic threshold for P_c that we use in our experiment in Part II, and this figure might be on a similar global form for other thresholds. Even if the distribution is not very wide, one can hope that setting a threshold for P_i like 0.86 may allow to discard some bad mappings, especially in this case where A_i is very probable: $P(B_j) = 0.75,\ P(A_i) = 0.75,\ P(A_i \cap B_j) = 0.6$
This leads to : $P_c(A_i \sqsubseteq B_j) = \frac{0.6}{0.75} = 0.8$ and $P_i(A_i \sqsubseteq B_j) = 1 - P(A_i \cap \overline{B_j} = 1 - (0.75 - 0.6) = 0.85$.

A threshold of 0.7 for P_c does not exclude this mapping, but a threshold of 0.9 for P_i does. Here, the ratio of instances in the darkened region that contradict the mapping $A_i \setminus B_j$ can be fairly considered too large, and so only the P_i model takes that into account.

A noticeable property for P_i is that the set of all mappings for which P_i exceeds a threshold is a closed set under logical implication according to the knowledge of taxonomies. This property is equivalent to the strong property of monotony (see Proposition IV.1), and therefore it is not the case for P_c.

Finally, another difference between the two models should be pointed out considering that both can be applied to all dual clauses.

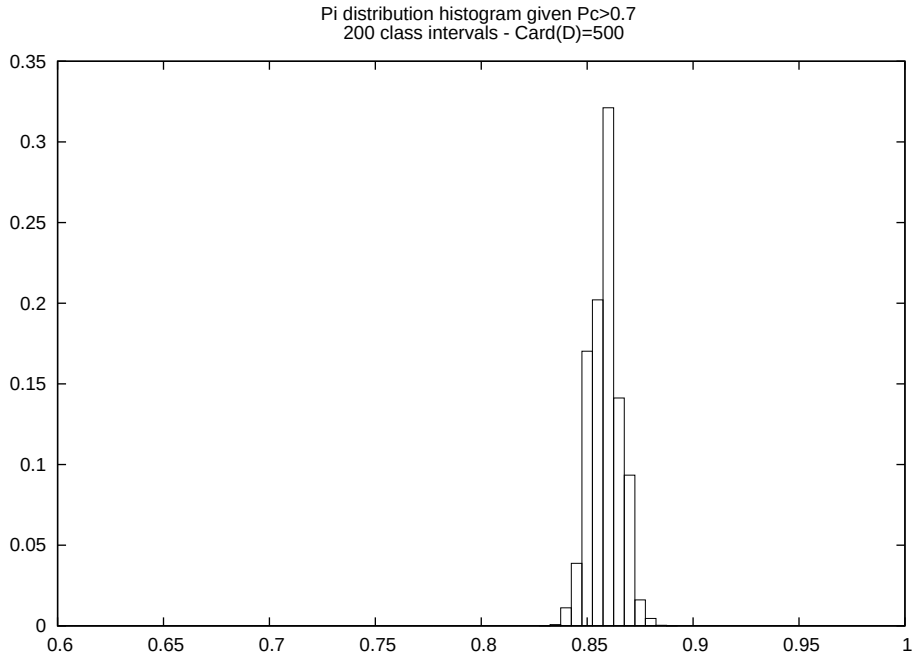


Figure IV.4: Distribution histogram for P_i given $P_c > 0.7$

In particular, P_i give the same value for $A_i \sqsubseteq B_j$ as for its contrapositive $\neg B_j \sqsubseteq \neg A_i$, because they have both the same set interpretation. Conversely, P_c does not respect this property: as shown in the following counter-example:
Continuing the previous example, we have:
$P(B_j) = 0.75,\ P(A_i) = 0.75,\ P(A_i \cap B_j) = 0.6$
$P(B_j|A_i) = 0.6$ and

$$P(\overline{A_i}|\overline{B_j}) = \frac{P(\overline{A_i} \cap \overline{B_j})}{P(\overline{B_j})} = \frac{1 - P(A_i) - P(B_j) + P(A_i \cap B_j)}{1 - P(B_j)} = \frac{0.1}{0.25} = 0.4$$

Characterization of monotonous confidence functions w.r.t. logical implication

In this section we want to characterize the functions $F_P(A, B)$ for measuring the confidence of any inclusion formula $A \sqsubseteq B$ that are compatible with the logical entailments of the taxonomies to be matched. More precisely, we show that F_P respect the strong property of monotony only if for each mapping $A \sqsubseteq B$, $F_P(A, B) = f(P_i(A \sqsubseteq B))$ where f is a monotonous increasing function. Thus, this theorem brings an additional justification to the relevance of the model P_i.

The general approach for doing this characterization consists in finding some necessary property for F_P according to the fact that F_P respects the strong property of monotony w.r.t. the logical implication. Given a probability space $(\mathcal{D}, \mathcal{P}(\mathcal{D}), P)$ of instances, taxonomies correspond to sets of annotated subsets of $\mathcal{D}$. Given two taxonomies $\mathcal{T}_1, \mathcal{T}_2$, we denote $F_P^{\mathcal{T}_1,\mathcal{T}_2}$ the restriction of F_P to mappings between $\mathcal{T}_1$ and $\mathcal{T}_2$. This restriction is well defined by considering that F_P

is independent of the taxonomies to be matched. Then for every A, B subsets of $\mathcal{D}$, for every taxonomies $\mathcal{T}_1, \mathcal{T}_2$, $F_P(A \sqsubseteq B) = F_P^{\mathcal{T}_1,\mathcal{T}_2}(A \sqsubseteq B)$.

The Lemma IV.α points out an intermediate result: a necessary condition for making F_P respecting the strong property of monotony. The proof is given in Appendix A in Section A.1.1.

Lemma IV.α:
If for every pair $\mathcal{T}_1, \mathcal{T}_2$ of taxonomies over $(\mathcal{D}, \mathcal{P}(\mathcal{D}), P)$, $F_P^{\mathcal{T}_1,\mathcal{T}_2}$ is monotonous w.r.t. set inclusion (i.e. to logical implication since sets correspond to classes in logic), then
for every subsets A_1, A_2, B_1, B_2 of Ω (i.e. classes),
if $A' \subseteq A$ and $B \sqsubseteq B'$ then $F_P(A, B) \leq F_P(A', B')$

The following theorem starts by assuming that F_P respect the necessary property of the Lemma IV.α (that is implied by monotony property w.r.t. logical implication.), and characterizes monotonous confidence functions F_P, under some additional assumptions:

1. the probability space should be infinite and no countable
2. for every pair of mappings sharing one common class and for which the not shared classes are related by an implication relation, the confidence function provides the same value for both mappings if they have the same probability P_i. (assumption (ii))
3. $F_P(A, B)$ is a continuous compound function of the probability of A and B and of their joint probability.

Theorem IV.2 (Characterization of probabilistic confidence functions):
Let $(\Omega, \mathcal{P}(\Omega), P)$ be a probability space such that Ω is infinite and no countable.

Let F_P be a function $\mathcal{P}(\Omega) \times \mathcal{P}(\Omega) \to [0; 1]$ such that:

(i) $\forall (A_1, A_2, B_1, B_2) \in \mathcal{P}(\Omega)^4$, *if* $B_1 \subseteq A_1$ *and* $A_2 \subseteq B_2$ *then* $F_P(A_1, A_2) \leq F(B_1, B_2)$

(ii) $\forall (A_1, A_2, B_1, B_2) \in \mathcal{P}(\Omega)^4$,
if ($A_1 \subseteq B_1$ *and* $A_2 = B_2$ *or* $A_2 \subseteq B_2$ *and* $A_1 = B_1$)
and $P_i(A_1 \sqsubseteq A_2) = P_i(B_1 \sqsubseteq B_2)$,
then $F_P(A_1, A_2) = F_P(B_1, B_2)$

(iii) $\exists G : [0; 1]^3 \to [0; 1]$ *continuous such that:*
$\forall (A, B) \in \mathcal{P}(\Omega)^2$, $F_P(A, B) = G(P(A), P(B), P(A \cap B))$

Then:

(1) $\forall (A_1, A_2, B_1, B_2) \in \mathcal{P}(\Omega)^4$ *such that* $P_i(A_1 \sqsubseteq A_2) \leq P_i(B_1 \sqsubseteq B_2)$,
$F_P(A_1, A_2) \leq F_P(B_1, B_2)$

(2) $\forall (A, B) \in \mathcal{P}(\Omega)^2, F_P(A, B) = f(P_i(A \sqsubseteq B))$ *where f is a monotonous increasing function w.r.t. its argument*

The proof of this theorem is given in Appendix A.1.2.

The probabilistic confidence function P_i uses the identity function for f. Therefore, Theorem IV.2 shows that P_i is the simplest confidence function which is well connected with logic.

Synthesis

The two probabilistic models that we propose are derived from the conditional probability theory and the material implication in logic. Both provide a probability measure for modeling the uncertainty of any mapping and rely to the join distribution of the two class extensions. They can be used to distinguish between good and bad mappings via respective thresholds, and they appear to be complementary with regard to their respective special extreme cases.

In addition, we have provided a grounded justification for P_i as the simplest probabilistic function which well connects to logic.

The initial underlying goal is to discover the best mappings with an efficient robustness. Therefore we aim at combining both models in the algorithm to discover the most probable mappings by thresholding the two probabilities $P_i(m)$ and $P_c(m)$ associated to each mapping m.

IV.4 Estimation of probabilities

As shown in point (4) of Proposition IV.1 page 44, the computation of $P_i(m)$ and $P_c(m)$ relies on computing the set probability $P(lhs(m))$ and the joint set probability $P(lhs(m) \cap rhs(m))$. Those values are unknown and must be estimated. They are the unknown parameters of the underlying Bernoulli distributions modeling the membership function to a set as a random variable taking only two possible values: 0 or 1.

Following the Bayesian approach to statistics [Deg04], we model those unknown parameters as continuous random variables, and we use *observations* to infer their *posterior* distribution from their *prior* distribution.

Usual prior distribution for the parameter p of a Bernouilli distribution is the Beta distribution, because Beta and Bernouilli distribution are conjugate: it means that the distribution of p given the observations has the same algebraic form than the distribution of p. Hence the computation of the Bayesian estimation is easier.

Therefore we assume that for a class E, the associated Bernouilli variable X_E has a parameter p_E which follows a Beta distribution $Beta(\alpha_E, \beta_E)$. As the expected value of this prior distribution should correspond to the probability of a random instance to be in a class, we assume that $E[Beta(\alpha_E, \beta_E)] = \frac{\alpha}{\alpha+\beta} = \frac{1}{2}$ so that $\beta_E = 2\alpha_E$.

In a similar way, for an intersection of class denoted $E = F \cap G$, the expected value of the corresponding Bernouilli variable is $\frac{1}{4}$ (by the above symmetry argument when stating that $E[P_u] = \frac{1}{4}$). Then we assume in this case that $\beta_E = 4\alpha_E$

From now on, we set the prior probability for any class E to $\frac{1}{2}$. It corresponds to the uniform distribution for the parameter p_E ($Beta(1,1)$ is equivalent to the uniform distribution). In a similar way, we set the prior probability for any conjunction of class to $\frac{1}{4}$.

Let $\widehat{Ext}(E, \mathcal{O})$ be the set of observed instances of $\mathcal{O}$ that are recognized to be instances of E. According to a basic theorem in probability theory (Theorem 1, page 160, [Deg04]), if the prior distribution of the random variable p_E modeling $P(E)$ is a *Beta distribution* of parameters α_E and β_E, then its posterior distribution is also a Beta distribution the parameters of which are: $\alpha_E + |\widehat{Ext}(E, \mathcal{O})|$ and $\beta_E + |\mathcal{O}|$.

According to the definition of the Bayesian estimator (in Section II.5, page 23), the estimator that minimizes the mean-square error is then the expected value of the posterior distribution

$$\widehat{p_E} = E[p_E|Ob] = \frac{\alpha_E + |\widehat{Ext}(E, \mathcal{O})|}{\alpha_E + \beta_E + |\mathcal{O}|}$$

It should be noticed that there is a choice in α_E and β_E although they are linked variables. The principle is the following : the higher the chosen value for α_E (and so β_E) is, the stronger the weight of the *a priori* is in the estimation. Note that for $\alpha_E = \beta_E = 1$, the corresponding *Beta* distribution is equal to the uniform one of expected value $\frac{1}{2}$. α_E and β_E can be seen as cardinals of positive and negative instance of a phantom observations set added to the real one. This principle is known as the Laplacian way to smooth the estimator.

Finally, for a mapping we can compute its conditional and its implication probabilities by combining their definition formula and estimations of their operands:

Theorem IV.3 (Estimation of mapping probabilities):
Let $m : C_i \sqsubseteq D_j$ be a mapping between two taxonomies $\mathcal{T}_i$ and $\mathcal{T}_j$. Let $\mathcal{O}$ be the union of instances observed in $\mathcal{T}_i$ and $\mathcal{T}_j$. Let $N = |\mathcal{O}|$, $N_i = |\widehat{Ext}(C_i, \mathcal{O})|$, $N_j = |\widehat{Ext}(D_j, \mathcal{O})|$ and $N_{ij} = |\widehat{Ext}(C_i \cap D_j, \mathcal{O})| = |\widehat{Ext}(C_i, \mathcal{O}) \cap \widehat{Ext}(D_j, \mathcal{O})|$

- *$P(C_i)$ is estimated by $\frac{1+\widehat{Ext}(C,\mathcal{O})}{2+N_i} = \frac{1+N_i}{2+N}$*
- *$P(\widehat{C_i \cap D_j})$ is estimated by : $\frac{1+|\widehat{Ext}(C_i \cap D_j,\mathcal{O})|}{4+N} = \frac{1+N_{ij}}{4+N}$*

that leads to:

- $\widehat{P_i}(m) = 1 + \frac{1+N_{ij}}{4+N} - \frac{1+N_i}{2+N}$
- $\widehat{P_c}(m) = \frac{1+N_{ij}}{4+N} \times \frac{2+N}{1+N_i}$

It is worth comparing the (Bayesian) ratios $\frac{1+N_i}{2+N}$ and $\frac{1+N_{ij}}{4+N}$ appearing in the formulas for computing $\widehat{P_i}(m)$ and $\widehat{P_c}(m)$ in Theorem IV.3 with the corresponding ratios $\frac{N_i}{N}$ and $\frac{N_{ij}}{N}$ that would have been obtained by following the standard (frequency-based) approach of statistics (as it is the case for instance in [DMDH02]). The corresponding ratios converge to the same expected value when there are many instances, but the Bayesian ratios are more robust to a small number of instances. In contrast with the frequency-based approach, they are defined even in the case where no instance is observed: their respective values (i.e., $\frac{1}{2}$ and $\frac{1}{4}$) in this particular case correspond to the probability of random sets and the joint probability of two random sets respectively for a uniform distribution of instances in the sample set.

Exploiting classifiers to merge instances of different taxonomies

In our setting, the set $\mathcal{O}$ is the union of the two (possibly disjoint) sets $\mathcal{O}_i$ and $\mathcal{O}_j$ of instances observed in two distinct taxonomies $\mathcal{T}_i$ and $\mathcal{T}_j$. This raises the issue of computing the set $\widehat{Ext}(E, \mathcal{O}_i \cup \mathcal{O}_j)$, especially when E is the conjunction of a class C_i of the taxonomy $\mathcal{T}_i$ and a class D_j of the other taxonomy $\mathcal{T}_j$. The computation of the extension of the intersection can be done as follows:
$\widehat{Ext}(C_i \cap D_j, \mathcal{O}_i \cup \mathcal{O}_j) = \widehat{Ext}(C_i, \mathcal{O}_i) \cap \widehat{Ext}(D_j, \mathcal{O}_j)$

Since the two taxonomies have been created and populated independently by different users, the only information that can be extracted from those two taxonomies are the extensions of each class *within* each taxonomy: $Ext(C_i, \mathcal{T}_i)$ and $Ext(D_j, \mathcal{T}_j)$.

By construction, it is likely that their intersection contains very few instances or even no instance at all, that makes the extension of an intersection between two classes of a mapping to be empty or almost empty, whatever the considered mapping.

Therefore, we use *automatic classifiers* to compute $\widehat{Ext}(E, \mathcal{O})$ inside both taxonomies. The machine learning community has produced several types of classifiers that have been extensively (theoretically and experimentally) studied (see [Mit97] for a survey) and have been made available through open source platforms like Weka [WF05]. They are based on different approaches (e.g., Naive Bayes learning, decision trees, Support Vector Machines) but they all need a training phase on two sets of positive and negative examples.

The general principle of a binary classifier learning a class C is to produce a function $Classifier_C$ for recognizing instances descriptions of C: $Classifier_C$ associates each instance description to a boolean target value. The allowed format for description depends on classifiers, but the time the descriptions are numeric vectors.

Such a function $Classifier_C$ is learned by a training step with examples and counter-examples of the class C. According to minimizing a cost criterion on the prediction error, the training phase provides a classifier function $Classifier_C$ that can predict a boolean value to each possible description, in particular on descriptions not already seen, in order to indicate if this instance with such a description is likely to belong to the learned class or not.

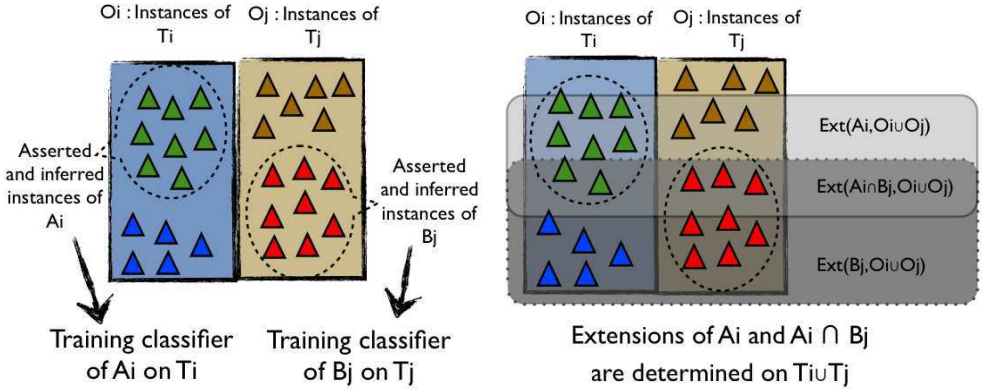
Let $Classifier$ be a classifier. Let C_i be a class of one of the two taxonomies that we denote by $\mathcal{T}_i$, the other one being denoted $\mathcal{T}_j$. For computing $\widehat{Ext}(C_i, \mathcal{O})$, we follow the same approach as the GLUE system [DMDH02], described in Section III.1.3 page 28:

1. $Classifier$ is trained on the descriptions of the elements of $\widehat{Ext}(C_i, \mathcal{O}_i)$ as positive examples, and of its complement set with regard to $\mathcal{O}_i$ as the negative examples.

2. $Classifier$ is then applied to each instance of $\mathcal{T}_j$ to recognize whether it belongs to C_i or not.

When an instance i is classified into a class C_i, the extensions of C_i and of all its ancestors are updated in order to contain i.

After the process of classification for the classes A_i, an extension on $\mathcal{T}_i$ and on $\mathcal{T}_j$ is known, and it is the same for B_j, as illustrated in Figure IV.5. Hence, the intersection extension of the two classes can be computed.



(a) Before classifications (b) After classifications

Figure IV.5: Classifications of instances

All the extensions of each class and their intersection obtained by classification can then be used to replace the initial extensions in the estimation formulas for P_i and P_c.

More details on the integration of this classification technique are given in the following chapter in section V.4 page V.4.

Finally, note that in the case for which there are enough common instances to statistically represent each class of both taxonomies, the full set of considered instances $\mathcal{O}$ can be restricted to the set of instances in common.

CHAPTER

FIVE

PROBAMAP: AN ALGORITHM FOR DISCOVERING MAPPINGS

We have proposed two probabistic models for mappings and a way to estimate these probabilities. The practical purpose of these probabilities considered in this work is the discovery of reliable mappings . An automatic method for discovering mapping have to differentiate mappings that are likely to be valid from an user perspective from unlikely mappings.

Therefore we introduce the notion of *valid mapping* based on the probabilistic semantics seen in Chapter I.

Definition V.1 (Valid mapping):
Given two taxonomies $\mathcal{T}_i, \mathcal{T}_j$ and two thresholds $S_i, S_c \in [0;1]$, a mapping m between $\mathcal{T}_i$ and $\mathcal{T}_j$ is valid *if:*

- $P_i(m) \geq S_i$
- $P_c(m) \geq S_c$

Both probabilities P_i and P_c are combined in that definition. As seen in Chapter I, P_i and P_c are complentary when selecting mappings, in the two perspectives of robustness and tuning. Note that by setting S_i or S_c respectively to 0, the corresponding criterion P_i or P_c can be bypassed.

Given two taxonomies $\mathcal{T}_i$ and $\mathcal{T}_j$ and their associated instances, the purpose of the discovery algorithm is to determine all possible mappings m between $\mathcal{T}_i$ and $\mathcal{T}_j$ that are valid.

Given two taxonomies $\mathcal{T}_i$ and $\mathcal{T}_j$ of respectively n_i and n_j classes, each pair of the Cartesian product of the two sets of classes corresponds to two mappings, one from $\mathcal{T}_i$ to $\mathcal{T}_j$, one from $\mathcal{T}_j$ to $\mathcal{T}_i$. Therfore there exists $n_i n_j$ mappings in each direction, so $2n_i n_j$ mappings in total. In the following we call these $2n_i n_j$ mappings *candidate mappings*.

A naive strategy to find the mappings whose probabilities exceeds their respective thresholds is a brute force algorithm: enumerate all candidate mappings and compute P_i and P_c for all of them. But the distributions of P_c and P_i seen in Chapter I lead to a large amount of candidate

mappings that are likely not to be valid. Indeed if for a mapping m we have $P_i(m) < S_i$, then none of its logical implicants has a corresponding P_i exceeding S_i, thanks to the property IV.1 of monotony. Hence a lot of mappings can be directly seen as invalid, instead of being tested.

The estimation of probabilities P_c and P_i for each tested mapping is done by counting the size of extensions of classes and computing intersection between the classes involved in the mapping. As it is time-consuming and that it represents a large part of the full complexity of the discovery process, we aims at minimizing the number of probability estimations.

That is why we introduce an optimized algorithm generating candidate mappings in a special order to maximize such applications of monotony, and then minimizing the number of computation of probabilities.

We start by introducing the ProbaMap core algorithm, without concern about implementation issue. Then we give the detailed algorithms and implementations for the preprocessing, the core algorithm and the optional classification phase. After that we give the algorithm for the classification phase. Some complexity elements are given at the end of this chapter.

V.1 Candidate mapping generation

The principle of the ProbaMap algorithm is to generate mappings from the two sets of classes in the two taxonomies according to a *topological order* [CLRS01] according to the entailment relation $\preceq$.

We denote $\mathcal{M}(\mathcal{T}_1, \mathcal{T}_2)$ the set of all possible mappings from $\mathcal{T}_1$ to $\mathcal{T}_2$: they constitute all the candidate mappings.

Proposition V.1 (DAG of entailment relation for mappings from $\mathcal{T}_i$ to $\mathcal{T}_j$):
The entailment relation $m \preceq m' \quad \Leftrightarrow \mathcal{T}_1, \mathcal{T}_2, m \models m'$ defines a DAG denoted $\mathcal{D}(\mathcal{T}_1, \mathcal{T}_2) = (\mathcal{M}(\mathcal{T}_i, \mathcal{T}_j), \mathcal{E})$ in which:

- *the vertices set is $\mathcal{M}(\mathcal{T}_i, \mathcal{T}_j)$*
- *the edges set $\mathcal{E}$ is constituted by all pairs $(m, m') \in \mathcal{M}(\mathcal{T}_i, \mathcal{T}_j)$ for which $m \preceq m'$ and there exists no $m'' \in \mathcal{M}(\mathcal{T}_i, \mathcal{T}_j)$ such that $m \preceq m'' \preceq m'$.*

In this DAG, the ancestors of a mapping correspond to its implicants, and the descendants of a mapping correspond to its consequences.

Proof :

Such a graph exists because it is a transitive reduction of the graph of the existing transitive closure graph of the relation $\preceq$.

It does not contain any cycle:
Let $m = A_i \sqsubseteq B_j$, $m' = C_i \sqsubseteq D_j$ be two mappings of $\mathcal{M}(\mathcal{T}_i, \mathcal{T}_j)$.
If $m \preceq m'$ and $m' \preceq m$:
By Proposition II.1,
$A_i \sqsubseteq B_j \preceq C_i \sqsubseteq D_j$ and $C_i \sqsubseteq D_j \preceq A_i \sqsubseteq B_j$ leads to:

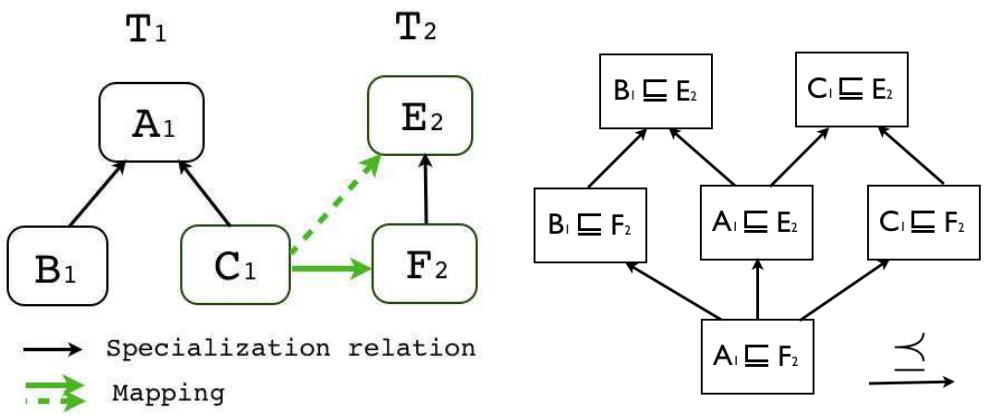
$\mathcal{T}_i \models A_i \sqsubseteq C_i, C_i \sqsubseteq A_i$, and : $\mathcal{T}_j \models B_j \sqsubseteq D_J, D_j \sqsubseteq B_j$.

As $\mathcal{T}_i$ and $\mathcal{T}_j$ are acyclic, $C_i = A_i$ and $D_j = B_j$, then $A_i \sqsubseteq B_j = C_i \sqsubseteq D_j$.

The correspondences between ancestors and implicants, and descendants and consequences are obvious because this DAG is a just a particular graph among all directed graphs representing the relation $\preceq$.

□

A topological order on a given DAG is a total order on the set of node compatible with the partial order relation of the DAG. In other words, a topological order on the set of mappings according to the entailment relation is a total order indexing mappings from 1 to n_m such that $m_i \preceq m_j \Rightarrow i \leq j$. In an enumeration following such an order, a mapping is seen before all its consequences. Conversely, if the enumeration follows the reverse order, a mapping is seen before all its implicants.



(a) Taxonomies (b) DAG of mappings

Figure V.1: Two taxonomies and associated DAG of mapping

Example V.1 (Topological order of mappings)
The Figure V.1(a) shows an example of two pictured taxonomies, with a mapping $C_1 \sqsubseteq F_2$ and a consequence mappings $C_1 \sqsubseteq E_2$. Figure V.1(b) shows the induced DAG of all mappings from the left taxonomy to the right one and the entailments between them. For instance, there is an edge that means an entailment from $C_1 \sqsubseteq F_2$ to $C_1 \sqsubseteq E_2$, and $A_i \sqsubseteq F_2$ is the most specific mapping among them and it entails all of them.

The two following propositions shows that a topological order for mapping according to the

entailment relation can be obtained directly from the two preprocessed topological orders of classes within each taxonomies.

The following proposition is a corollary of Proposition II.1.

Proposition V.2:
Let $\mathcal{T}_i$ and $\mathcal{T}_j$ two taxonomies.
Let $Topo(\mathcal{T}_i)$ be the sequence of classes of $\mathcal{T}_i$ resulting from a topological sort of $\mathcal{T}_i$. Let $Topo(\mathcal{T}_j)$ be the sequence of classes of $\mathcal{T}_j$ resulting from a topological ordering of $\mathcal{T}_j$. Let $m : C_i \sqsubseteq D_j$ and $m' : C'_i \sqsubseteq D'_j$ two different mappings from $\mathcal{T}_i$ to $\mathcal{T}_j$. If m' is an implicant of m then C_i is after C'_i in $Topo(\mathcal{T}_i)$ or $C_i = C'_i$ and D_j is before D'_j in $Topo(\mathcal{T}_j)$.

Proof :

Let $m = C_i \sqsubseteq D_j$ and $m' = C'_i \sqsubseteq D'_j$ be two different mappings like those introduced above, such that $m' \preceq m$, and that respect the following assumption (H), which denies the Property V.2:

- C'_i is before C_i (H_1)
  OR
- $C'_i = C_i \wedge D_j$ is after D'_j (H_2).

The application of Proposition V.2 leads to:
$\mathcal{T}_i \models C_i \sqsubseteq C'_i$ (a) and
$\mathcal{T}_j \models D'_j \sqsubseteq D_j$ (b).

H_1 is contradictory to (a) and the topological order in $Topo(\mathcal{T}_i)$.

H_2 is contradictory to (b) and the topological order in $Topo(\mathcal{T}_j)$

As a consequence, assuming that there exists a pair of mapping that respects (H) leads to a contradiction. Therefore, for all pairs of mappings (H) is false, and then the Property V.2 is true.

□

Proposition V.3:
Let $\mathcal{T}_i$ and $\mathcal{T}_j$ two taxonomies.
Let $Topo(\mathcal{T}_i)$ be the sequence of classes of $\mathcal{T}_i$ resulting from a topological sort of $\mathcal{T}_i$. Let $Topo(\mathcal{T}_j)$ be the sequence of classes of $\mathcal{T}_j$ resulting from a topological sort of $\mathcal{T}_i$.
Consider the sequence of mappings $C_i \sqsubseteq D_j$ constructed in the lexicographic order of the reverse sequence $Topo(\mathcal{T}_i)$ and the sequence $Topo(\mathcal{T}_j)$: $C_i \sqsubseteq D_j$ is before $C'_i \sqsubseteq D'_j$ in the sequence if C_i is after C'_i in $Topo(\mathcal{T}_i)$ or $C_i = C'_i$ and D_j is before D'_j in $Topo(\mathcal{T}_j)$ This sequence is a topological order of the DAG of mappings directed from $\mathcal{T}_i$ to $\mathcal{T}_j$.

Proof :

From Proposition V.2, all pairs of mappings in the sequence respect the topological order of the DAG of mappings.
As all candidate mappings from $\mathcal{T}_i$ to $\mathcal{T}_j$ are in the sequence, it constitutes a topological order of the DAG of mappings.

□

Example V.2 (Application of the Propositions V.2 and V.3)
The goal of the property V.2 is to show that a topological order of Figure V.1 can be computed from two topological orders of the taxonomies of Figure V.1(a). For instance, B_1, C_1, A_1 *is a topological order for* $\mathcal{T}_1$*, and* F_2, E_2 *for* $\mathcal{T}_2$*.*
The proposition V.2 leads to the following order for mappings from $\mathcal{T}_1$ *to* $\mathcal{T}_2$*:*
$A_1 \sqsubseteq F_2, A_1 \sqsubseteq E_2$,
$B_1 \sqsubseteq F_2, B_1 \sqsubseteq E_2$,
$C_1 \sqsubseteq F_2, C_1 \sqsubseteq E_2$
This order fits the DAG of entailments between mappings, it is a topological order according to mapping entailment.

V.2 Pruning the candidate mappings to test

Based on the monotony property of the probabilistic confidence function P_i (Theorem IV.1), every mapping m' that entails a mapping m such that $P_i(m) < S_u$ verifies $P_i(m') < S_u$. Therefore, in the algorithm we prune the probability estimation of all mappings that entails any mapping m such that $\widehat{P}_i(m) < S_i$. We shall use the notation $Implicants(m)$ to denote the set of all mappings that entails m.

Similarly, based on the property of weak monotony of the probabilistic confidence function P_c (Theorem IV.1), when a tested candidate mapping m is such that $\widehat{P_c}(m) < S_c$ we prune the probability estimation of all mappings that entail m having the same left-hand side as m.

Proposition V.2 has a convenient consequence:

Proposition V.4 (Topological order enumeration maximizes pruning):
Testing the validity of mappings following the reverse topological order according to the $\preceq$ *relation maximizes the pruning.*

Proof Proof of the proposition V.4:

Let assume that there exists a currently enumerated mapping m that is not pruned but that should have been so, according to the strong monotony property of P_i (resp. the weak monotony property of P_c).

Then, there exists a mapping m' which should have triggered the pruning, so such that $P_i(m') < S_i$ and $m \preceq m'$ (resp. $P_c(m') < S_c$, $m \preceq m'$ and $lhs(m) = lhs(m')$). If such a

pruning has not be done, then m' has not been examined yet. It means that m' will be seen after m.

This is contradictory with the fact that the mappings are enumerated according to the reverse topological order according to $\preceq$ and that $m \preceq m'$.

□

Based on the order in which the mappings are generated, Proposition V.2 shows that the order of generation of mappings maximizes the number of pruning.

V.3 The ProbaMap algorithm

We have designed an algorithm which combines the generation and the pruning seen above for discovering valid mappings. We firstly describe a simple generic version of it in Algorithm 1. $inst_i$ and $inst_j$ are two maps that store the declared instances for each class of respectively $\mathcal{T}_i$ and $\mathcal{T}_j$. For instance, $inst_i(A_i)$ is the set of declared instances for the class A_i. $Nodes(\mathcal{T})$ represent the set of classes for a taxonomy DAG $\mathcal{T}$.

Algorithm 1 ProbaMap-Simple

Require: Taxonomies (DAG) $\mathcal{T}_i, \mathcal{T}_j$, thresholds S_c, S_i, Instances maps $inst_i, inst_j$
Ensure: returns $\{m \in \mathcal{M}(\mathcal{T}_i, \mathcal{T}_j)$ such that $\widehat{P_i}(m) \geq S_i$ *and* $\widehat{P_c}(m) \geq S_c\}$

1: $M_{Val} \leftarrow \emptyset$
2: $M_{NVal} \leftarrow \emptyset$
3: **for all** $C_i \in$ reverse topological order of $\mathcal{T}_i$ **do**
4: **for all** $D_j \in$ in direct topological order of $\mathcal{T}_j$ **do**
5: let $m = C_i \sqsubseteq D_j$
6: **if** $m \notin M_{NVal}$ **then**
7: **if** $\widehat{P_i}(m, inst_i, inst_j, \mathcal{T}_i, \mathcal{T}_j) \geq S_i$ **then**
8: **if** $\widehat{P_c}(m, inst_i, inst_j, \mathcal{T}_i, \mathcal{T}_j) \geq S_c$ **then**
9: $M_{Val} \leftarrow M_{Val} \cup \{m\}$
10: **else**
11: $M_{NVal} \leftarrow M_{NVal} \cup$ IMPLICANTS_LHS$(m, \mathcal{T}_j)$ // Pruning using the weak monotony
12: **end if**
13: **else**
14: $M_{NVal} \leftarrow M_{NVal} \cup$ IMPLICANTS$(m, \mathcal{T}_i, \mathcal{T}_j)$ // Pruning using the strong monotony
15: **end if**
16: **end if**
17: **end for**
18: **end for**
19: return M_{Val}

Algorithm 1 returns mappings directed from $\mathcal{T}_i$ to $\mathcal{T}_j$. In order to obtain *all* valid mappings, it must be applied again by swapping its inputs $\mathcal{T}_i$ and $\mathcal{T}_j$.

The discovered valid mappings are stored in the set M_{NVal}. Mappings that are pruned are stored

in the set M_{NVal}. The nested loops in Line 4 in Algorithm 1 generate all the mappings $C_i \sqsubseteq D_j$ by enumerating the classes C_i of $\mathcal{T}_i$ following a *reverse* topological order and the classes D_j of $\mathcal{T}_j$ following a *direct* topological order according to the class specialization relation. According to the property V.2, it leads to generate mappings in the reverse topological order with regard to the logical entailment, form the most general to the most specific mappings.

When a mapping is generated and if it is not already pruned, it is firstly tested with regard to $P_i(m) \geq S_i$ in Line 7, then if it is positive, it is tested with $P_c(m) > S_c$ in Line 8. In the case $P_i(m) < S_i$, all the implicants of m are marked as pruned (Line 14), thanks to the strong monotony property of P_i. This is done by the function IMPLICANTS $(m, \mathcal{T}_i, \mathcal{T}_j)$ which returns all the implicants of m according to $\mathcal{T}_i$ and $\mathcal{T}_j$

If $P_i(m) \geq S_i$ but $P_c(m) < S_c$, then the property of weak monotony conducts to prune all the implicants of m with the same left-hand side. They are returned by the function call IMPLICANTS_LHS$(m, \mathcal{T}_j)$ in Line 11 and added to M_{NVal}.

This ordered generation of mappings make the DAG of mappings with the entailment relation $\preceq$ be traversed without been constructed. Such a DAG is likely to be huge ($NiNj$ vertices and at most $(NiNj)^2$ edges), and to be the largest structure in the algorithm. We save a lot of memory space when avoiding constructing it.

In the case the generated mapping to consider is already pruned, it is skipped. In order to know if a mapping is pruned or no, the M_{NVal} set is kept up to date by containing all the pruning mappings from the beginning. Storing pruned mappings in a single set is a simple and efficient way to traverse this DAG in the required order while handling an implicit pruning on it in the same time.

Proposition V.5 (Correctness and completeness of ProbaMap):
Given two taxonomies $\mathcal{T}_i, \mathcal{T}_j$ and two thresholds S_i, S_c, under the assumptions that $\widehat{P_u} = P_u$ and $\widehat{P_c} = P_c$, and that IMPLICANTS *and* IMPLICANTS_LHS *are correct,*
ProbaMap returns all the possible mappings m from $\mathcal{T}_i$ to $\mathcal{T}_j$ for which $P_i(m) \geq S_i$ and $P_c(m) \geq S_c$.

Proof :

Correctness
Each mapping m added in M_{Val} respects $\widehat{P_i}(m) \geq S_i$ and $\widehat{P_c}(m) \geq S_c$ thanks to the two tests in Lines 7 and 8 in Algorithm 1.
With the assumed confusion between $\widehat{P_c}$ and P_c, and $\widehat{P_i}$ and P_i, each mapping m that constitutes the returned set M_{Val} verify $P_i(m) \geq S_i$ and $P_c(m) \geq S_c$.

Completeness
The nested loops in Line 4 generates all the candidate mappings from $\mathcal{T}_i$ to $\mathcal{T}_j$. The test of a mapping m is skipped only if m is pruned (it belongs to M_{NVal}). Assuming that the IMPLICANTS and IMPLICANTS_LHS are correct, such a mapping m has been pruned from an consequence mapping m' for which $\widehat{P_i}(m') < S_i$ or $\widehat{P_c}(m') < S_c$. Either $P_i(m) \leq P_i(m') = \widehat{P_i}(m) < S_i$ or $P_c(m) \leq P_c(m') = \widehat{P_c}(m') < S_i$. Then m' is invalid.

All of skipped mappings are invalid. Therefore, all of the others are tested and so no valid mapping is skipped. M_{Val} finally contains all possible valid mappings.

□

V.3.1 Implementation of ProbaMap

An overview of the actual mapping discovery sequence is the following:

1. Preprocessing phase (mandatory) $\longrightarrow$ factors the computation of some data structure used several times in the ProbaMap algorithm
2. Classification phase (optional) $\longrightarrow$ classifies the instances of $\mathcal{T}_i$ in classes of $\mathcal{T}_j$ and vice-versa
3. The ProbaMap core algorithm $\longrightarrow$ computes the mappings whose probabilities P_i and P_c exceed their respective thresholds, with data from 1 and 2.

The implementation of ProbaMap is detailed in the following algorithm 2. The input differs slightly from the input of the simple Algorithm 1: in addition it requires the two sets of instances I_i and I_j of both taxonomies, and three parameters for the classification purpose:

1. $desc_i$ is the map that associates its description to each instance of $\mathcal{T}_i$
2. $desc_j$ is the map that associates its description to each instance of $\mathcal{T}_j$
3. $ClassEnabled$ is the boolean variable used to switch on/off the classification step

The two taxonomies DAG $\mathcal{T}_i$, $\mathcal{T}_j$ are basically assumed transitively reduced: if there exist an edge between A_i and C_i in $\mathcal{T}_i$, there is no another longer path from A_i to C_i.

Here are detailed the functions and variables used in the algorithm during the three parts:
Preprocessing

- $Topo_i$ and $Topo_j$ denote the respective sequences $Topo(\mathcal{T}_i)$ and $Topo(\mathcal{T}_j)$, that are computed by the function Order_Topo in Line 2 taking a DAG as input.
- $TClo_i, TClo_j$ are the respective transitive closure of the DAG of $\mathcal{T}_i$ and $\mathcal{T}_j$ that are computed by the function Closure in Line 4.
- ext_i, ext_j appearing in Lines 5,6 are the two maps (one for each taxonomy) associating to each class its extension : $ext(C_i) = Ext(C_I, \mathcal{T}_i)$. They are computed from the instances declared maps $inst_i$ and $inst_j$ and the transitive closure of the taxonomies DAG, by the Infer function:
  Infer$(inst_i, TClo_i)$ associates to each class C_i the set of instances:
  Infer$(C_i, TClo_i) = inst_i(C_i) \cup \bigcup_{C \in \text{Predecessors}(C_i, TClo_i)} inst_i(C)$

The extension of each classes C_i is used several times, for example when estimating the probabilities of mappings of the form $C_i \sqsubseteq D_j$, and when training the classifier for C_i. That is why all extensions are computed in the preprocessing phase of the algorithm.

The best way to compute them is to perform a traverse of the DAG in the topological order, as for computing the transitive closure of the taxonomies.

Classification
In Line 9 CLASSIFICATION($\mathcal{T}_i, ext_i, desc_i, I_i, I_j, desc_j$) is used to complete the extension of each classes of $\mathcal{T}_i$ with instances of $\mathcal{T}_j$ by exploitation of classifiers trained on the descriptions of instances of $\mathcal{T}_i$, denoted in the map $desc_i$. A similar call of CLASSIFICATION happens in the next line for classifying instances of $\mathcal{T}_i$ into classes of $\mathcal{T}_j$.

Core algorithm

- ESTIMATION_PCLASS(C_i, ext_i), ESTIMATION_PJOINT(A_i, B_j, ext_i, ext_j) in Lines 23,24,20,21 are the two functions that compute the Bayesian estimator probability for a given class C_i or an intersection of two given classes A_i and B_j. When the classification phase is enabled, the only difference is that the estimation of the joint and class probabilities aez done on the classified extensions of classes (Lines 20, 21).

- IMPLICANTS_LHS($m, \mathcal{T}_i$) is the implemented function (described below) that returns the implicants of the mapping m in $\mathcal{T}_i$ with the same left-hand side.

- IMPLICANTS_OPTIMIZED($m, \mathcal{T}_i, \mathcal{T}_j$) is the implemented function (described below) that returns the implicants of the mapping in argument but without a part of them already pruned by that procedure.

V.3.2 Implementation of the computation of probabilities estimations

As the two probabilities $P_i(m)$ and $P_c(m)$ can be computed from $P(lhs(m))$ and $P(lhs(m) \cap rhs(m))))$, the estimation of these probabilities are computed from the estimations of $P(lhs(m))$ and $P(lhs(m) \cap rhs(m))$. Following formulas given in Theorem IV.3, and by considering that the observed instances $\mathcal{O}$ are represented in the argument maps ext_i, ext_j,

- the set of observed instances for A $\widehat{Ext}(A_i, \mathcal{O})$ is replaced by $ext(A_i)$

- the set of observed instances for $A \cap B$ is replaced by $\widehat{Ext}(A_i \cap B_j, \prime)$ by $ext(A_i) \cap ext(B_j)$

It leads to the two functions:

- ESTIMATION_PCLASS(A_i, ext_i, N_i)= $\frac{1+|ext_i(A_i)|}{2+N_i}$

- ESTIMATION_PCLASS($A_i, B_j, ext_i, ext_j, N$)= $\frac{1+|ext_i(A_i) \cap ext_j(B_j)|}{4+N}$

V.3.3 Implementation of the pruning functions

A special attention should be given for the two pruning functions IMPLICANTS_LHS and IMPLICANTS_OPTIMIZED. They rely on classical primitives on DAG, providing predecessors, successors, ancestors and descendants of a class node.

As the DAG are stored directly by adjacency lists, we assume that the primitive PREDECESSORS$(C, \mathcal{T})$, SUCCESSORS$(C, \mathcal{T})$ that returns the predecessors and the successors of a class C in a taxonomy $\mathcal{T}$ are executed in $O(1)$.

The transitive closure of each taxonomies $TClo_i, TClo_j$ are computed in the preprocessing of ProbaMap, therefore the ancestors and descendants of a class C_i can be computed in $O(1)$:

- PREDECESSORS$(C_i, TClo_i)$ provides the ancestors of any class C_i in $\mathcal{T}_i$
- SUCCESSORS$(C_i, TClo_i)$ provides the descendants of any class C_i in $\mathcal{T}_i$

If $P_c(A_i \sqsubseteq B_j) < S_c$, all implicants mappings of $A_i \sqsubseteq B_j$ with the same left-hand side have to be pruned. Due to the Proposition II.1, they are all the mappings of the form $A_i \sqsubseteq D_j$ such that $\mathcal{T}_j \models D_j \sqsubseteq B_j$.

The naive approach for implementing IMPLICANTS_OPTIMIZED is to compute the Cartesian product of all ancestors of $lhs(m)$ and all descendants of $rhs(m)$, but it leads to a lot of redundant computation because many pruned subsets overlap during the execution of ProbaMap. In other words, many mappings may be pruned twice or more times, and the fact that a lot of invalid mapping are found contributes to redundant pruning of the same mappings, as illustrated in Figure V.2.

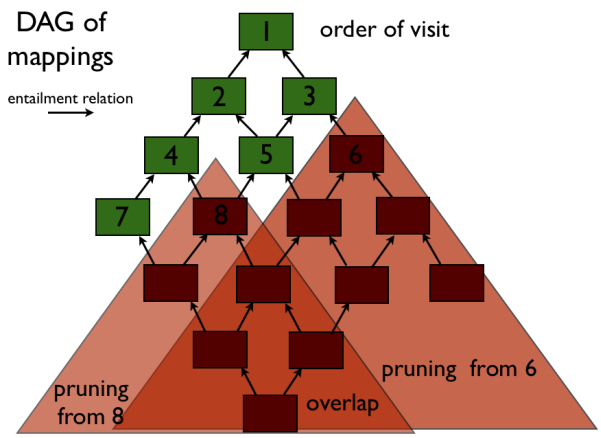


Figure V.2: Example of redundant pruning

The goal is to avoid generating mappings already pruned. In addition, it should be avoided to store in memory sets of mappings that are already pruned. Hence we are forbidden to explicitly compute the whole set of implicants or consequences of a particular mapping in order to do any intermediate optimization or computation.

Therefore, we adopt another approach based on an optimized partial traverse of the DAG of mappings $\mathcal{D}(\mathcal{T}_i, \mathcal{T}_j)$ to prune all implicant of one mapping that are not already pruned. This DAG is traversed without been explicitly constructed in order to limit memory usage, and because its whole construction would compensate negatively any effort to avoid to consider and store mappings twice.

The generation of the implicit DAG of mappings is based on the following property, which give a way to generates predecessors of a mapping m on the fly in the DAG of mappings, using the DAGs of the two taxonomies.

Proposition V.6 (Predecessors in $\mathcal{D}(\mathcal{T}_i, \mathcal{T}_j)$):
Let two taxonomies $\mathcal{T}_i, \mathcal{T}_j$. Let $\mathcal{D}(\mathcal{T}_i, \mathcal{T}_j)$ be the DAG as defined in proposition V.1. reduced. Let $m = C_i \sqsubseteq D_j$ be a mapping. Under the assumption that $\mathcal{T}_1, \mathcal{T}_2$ are transitively reduced, the predecessors of $m = C_i \sqsubseteq D_j$ in D are all and only all the mappings of the form:

- *$A_i \sqsubseteq D_j$ with an edge from C_i to A_i in $\mathcal{T}_i$*
- *$C_i \sqsubseteq B_j$ with an edge from B_j to D_j in $\mathcal{T}_j$*

Proof :

Let $m = C_i \sqsubseteq D_j$ be a mapping between $\mathcal{T}_i$ and $\mathcal{T}_j$. Let $m' = C_i \sqsubseteq B_j$ be a mapping such that B_j is connected to D_j with an edge in $\mathcal{T}_j$.
It is clear that $m' \preceq m$ because $\mathcal{T}_j, m' \models m$. If there exists a mapping m'' such that $m' \preceq m'' \preceq m$, then $m'' = E_i \sqsubseteq F_j$ is such that:
$\mathcal{T}_i \models C_i \sqsubseteq E_i \wedge E_i \sqsubseteq C_i$
$\mathcal{T}_j \models B_j \sqsubseteq F_j \wedge F_j \sqsubseteq D_j$

As there is no cycle in $\mathcal{T}_i$, $C_i = E_i$. As B_j and D_j are connected by an edge and no other path, either $F_j = B_j$ or $F_j = D_j$.

Then the two possible mappings for m'' are $C_i \sqsubseteq B_j$ and $C_i \sqsubseteq D_j$, so they are m and m'. There is no mapping m'' distinct from m and m' such that $m' \preceq m'' \preceq m$.

Therefore, m' is a predecessor of m. Then, each mapping $m' = C_i \sqsubseteq B_j$ is a mapping such that B_j is connected to D_j with an edge in $\mathcal{T}_j$ is a predecessor of m in $\mathcal{D}(\mathcal{T}_i, \mathcal{T}_j)$.

A similar reasoning can be done for each mapping $m' = A_i \sqsubseteq D_j$ such that there exists an edge between C_i and A_i in $\mathcal{T}_i$.

Let us assume that $m' = A_i \sqsubseteq B_j$ is a predecessor of $m = C_i \sqsubseteq D_j$. A fortiori, m' entails m and by Proposition II.1, we have $\mathcal{T}_i \models C_i \sqsubseteq A_i$ and $\mathcal{T}_j \models B_j \sqsubseteq D_j$.

There exist no class E_i in the path from C_i to A_i in $\mathcal{T}_i$, and no class F_j in the path from B_j to D_j in $\mathcal{T}_j$.

Otherwise, if there exist a class E_i in the path between C_i and A_i, we have :
$E_i \sqsubseteq B_j \;\; \preceq \;\; C_i \sqsubseteq B_j$, so $E_i \sqsubseteq B_j \;\; \preceq \;\; C_i \sqsubseteq D_j$
$A_i \sqsubseteq B_j \;\; \preceq \;\; E_i \sqsubseteq B_j$

Then $m'' = E_i \sqsubseteq B_j$ is such that $m' \preceq m'' \preceq m$ and that contradicts the fact that m' is a predecessor of m.

A similar reasoning can be done with assuming that a class F_j is in the path between B_j and D_j.

□

The principle of the IMPLICANTS_OPTIMIZED function is that the DAG D is traversed from m by generating successively predecessors of m, and then their own predecessors, etc. It leads to an implicit depth-first traverse of $\mathcal{D}(\mathcal{T}_i, \mathcal{T}_j)$ from m. This traverse is optimized by avoiding traversing mappings that have already been traversed.

The algorithm 4 presents the implementation of the function IMPLICANTS_OPTIMIZED:

Thanks to the test Line 14, when a mapping has been generated but is already pruned thanks by the function IMPLICANTS_OPTIMIZED (use of the static variable $StrongPruned$), its implicants are not generated from it. This is done by not pushing its predecessors to the stack. That is the point of the optimization.

But some of its implicants can be generated later because they are implicants of another mapping not already pruned.

Proposition V.7 (Termination, correctness and completeness of Algorithm 4): *Given two taxonomies $\mathcal{T}_i, \mathcal{T}_j$ transitively reduced, a mapping m and a set of mapping already pruned,* IMPLICANTS_OPTIMIZED *returns at least all mappings that implies m that are not already pruned by this function.*

Proof :

Let D be the DAG of mappings from $\mathcal{T}_i$ to $\mathcal{T}_j$.

Correctness
Let $m = A_i \sqsubseteq B_j$ be the mapping from which IMPLICANTS_OPTIMIZED is called. According to the fact that there are predecessors of m or transitive predecessors of m that are pushed in the stack, each mapping added to $ToPrune$ is of the form $C_i \sqsubseteq D_j$ with $\mathcal{T}_i \models C_i \sqsubseteq A_i$ and $\mathcal{T}_j \models B_j \sqsubseteq D_j$. By application of Proposition II.1, each mapping added to $ToPrune$ is an implicant of m.

Therefore the returned set of mapping is only constituted of implicants of m.

Termination
All mappings that are pushed in the stack S are either predecessors of m (according to Proposition V.6), or predecessor of mappings that are already in the stack. Then, by a recurrence reasoning it is clear that S contains only ancestors of m in D, that are all implicants of m.

A mapping m can not appear in S more than its number of successors $Succ(m)$ in $\mathcal{D}(\mathcal{T}_i, \mathcal{T}_j)$, because each of successors used to generate m are added to $StrongPruned$ and will not be used again to generate m (because of the test in Line 14). $|Succ(m)|$ is majored for all m of D by the maximum outdegree Deg_{out} of D.

At the ith iteration of the while loop in Line 8, the size of S is majored by $Deg_{out} - i$ that corresponds to a strictly decreasing sequence minored by the size 0. Therefore, the number of iteration is bounded because the stack becomes empty.

Completeness
We denote $StrongPruned_0$ the set $StrongPruned$ as its initial state when IMPLI-

CANTS_OPTIMIZED is called. Therefore, $StrongPruned_0$ contains all mappings already pruned by a previous call of IMPLICANTS_OPTIMIZED.

Let $Level(m, k)$ be the set of implicants m' of m for which the minimal path from m' to m has a length of k in $\mathcal{D}(\mathcal{T}_i, \mathcal{T}_j)$. If we denote K the longest path in $\mathcal{D}(\mathcal{T}_i, \mathcal{T}_j)$, the set $Level(m, k)$ for $0 \leq k \leq K$ constitutes a partition of the implicants of m.

Let $H(k)$ be the predicate that the set of mappings returned by IMPLICANTS_OPTIMIZED contains all mappings of $Level(m, k) \setminus StrongPruned_0$.

$H(0)$ is trivially true: $Level(m, 0) = \{m\}$. In the algorithm 4, the returned set contains m is contained if it is not already in $StrongPruned_0$.

Let us assume that $H(k)$ is verified. Let us assume that there exist a mapping m' of $Level(m, k+1)$ which is neither in $StrongPruned_0$ nor in $ToPrune$ at the end of the execution.

As m' belongs to $Level(m, k+1)$, there exist a shortest path of length $k+1$ from m' to m in $\mathcal{D}(\mathcal{T}_i, \mathcal{T}_j)$. It can be decomposed into a path of length 1 between m' and a mapping m_k, and a path of length k between m_k and m. We have $m_k \in Level(m, k)$, else the path between m' and m is not the shortest path.

As $H(k)$ is verified, all mappings of $Level(m, k)$ are either in $StrongPruned_0$, or pushed then popped, and added to $ToPrune$ during the algorithm. In addition, all of their predecessors are pushed to the stack. This happens in Line 12 according to the Proposition V.6 and the fact that taxonomies are transitively reduced.

If m_k belongs to $StrongPruned_0$, it means that it has been pushed in the stack then popped in a previous call of IMPLICANTS_OPTIMIZED, then that all its predecessors are pruned, including m'. That is not the case because m' is assumed not to belong to $StrongPruned_0$. Then $m_k \notin StrongPruned_0$.

As m' is a predecessor of m_k which is not already pruned, m' is pushed to the stack at a time. (Note that it can be pushed multiple times from multiple successors m_k).

As m' does not belong to $StrongPruned_0$, the first time it is pushed to the stack, it is added to $ToPrune$ and so is returned by the algorithm.

In conclusion, $H(k+1)$ is true because all mappings of $Level(m, k+1)$ that do not belong to $StrongPruned_0$ are returned by IMPLICANTS_OPTIMIZED.

By recurrence on the levels $Level(m, k)$, we have proved that each mapping of each $Level(m, k)$ for $k \in [0; K]$ is contained in the returned set to prune or in $StrongPruned_0$.

As $Level(m, k)$ constitutes a partition of the implicants of m, all implicants are either returned by the algorithm or already pruned by it.

□

Note that we can only avoid to prune mappings that are already pruned by IMPLICANTS_OPTIMIZED, because a mapping pruned by IMPLICANTS_LHS has only a subset of its implicants that are already pruned. That is why we distinguish the static variable

StrongPruned from the variable M_{NVal} of the algorithm.

Proposition V.8 (Complexity of IMPLICANTS_OPTIMIZED):
The overall complexity of all the calls of IMPLICANTS_OPTIMIZED *during the whole ProbaMap execution can be asymptotically bounded by* $O(m_i + m_j)$.

Proof :

Indeed, IMPLICANTS_OPTIMIZED has a linear complexity with respect to the number of mapping considered in all the *Pred* sets (Line 12 of algorithm 4).

As a predecessor is only generated when one of its seed successor is popped from the stack without belonging to *StrongPruned*, the same mapping can not be used to generate its predecessor more than one time. Therefore, a mapping can be generated and considered at most the number of successor it has. As a consequence, each edge in $\mathcal{T}_i$ and in $\mathcal{T}_j$ is only used one time for each generated mappings to consider in *Pred*.

The number of iterations considering the elements of *Pred* over all the calls of IMPLICANTS_OPTIMIZED is therefore majored by the total number of edges in both $\mathcal{T}_i$ and $\mathcal{T}_j$.

□

V.4 Classification phase

The set of observed instances that belongs to A_i and B_j is the intersection of the two set of observed instances for A_i and B_j:

$$\widehat{Ext}(A_i \cap B_j, \mathcal{O}) = \widehat{Ext}(A_i, \mathcal{O}_i) \cap \widehat{Ext}(B_j, \mathcal{O}_j)$$

When taxonomies are independently populated from disjoint and independent pool of instances, classification on instances can be exploited to merge the instances of both taxonomies in order to avoid to obtain empty intersections of extensions, as explained in Section IV.4.

The general principle of the classification phase of ProbaMap is to construct one classifier for each class of both taxonomies, each classifier learning its corresponding class C provided the descriptions of instances of C as examples, and all descriptions of other instances in the taxonomy of C as counter-examples.

The classification phase is separated into two parts:

1. classification of instances of $\mathcal{T}_j$ in classes of $\mathcal{T}_i$
2. classification of instances of $\mathcal{T}_i$ in classes of $\mathcal{T}_j$.

Due to their symmetry we only describe the first part.

In the algorithm 2 we have given the articulation of such a processing with regard to the ProbaMap algorithm. We now give the detail of the CLASSIFICATION function.

We denote $desc_i, desc_j$ the maps giving the description for every instance I of both $\mathcal{T}_i, \mathcal{T}_j$. These descriptions are preprocessed by the PREPROCESS function in order to obtain a vector of numerical values for each instances. The raw format of description as the detailed preprocessing are detailed in the Part II. The function $Train$ creates a classifier in function of some provided examples and counter-examples descriptions, which are preprocessed according to the learned class. This classifier make a decision according to the description in argument. It returns `true` if it predicts that the corresponding instance belongs to the class and `false` otherwise.

The implementation of classifiers that we use are those of Weka 3.6.4, i.e. NaiveBayes, J48 for C4.5 and SMO for the SVM. The formats of their input and their output are the same, but J48 and SMO have some different parameters to be tuned. A more detailed usage of classifiers is exposed in part II.

V.5 Complexity

The practical implementation is done in Java 1.6 using the `jdt-graph 1.6` library for representing DAG structures of taxonomies, computing the closure and the topological sort of DAGs.

We give here some complexity elements for the whole ProbaMap algorithm. We denote n_i, n_j the number of classes of $\mathcal{T}_i, \mathcal{T}_j$, and m_i, m_j the number of edges in each of them, corresponding to the specialization relation between classes. N_i, N_j denotes the number of instances in respectively $\mathcal{T}_i, \mathcal{T}_j$, and N the total number of instances in both taxonomies. We measure the complexity according to the classical elementary operations like accessing to/adding/removing an element of a set or a list, algebraic operations or comparisons and affectations.

- The complexity of the preprocessing phase is majored by the complexity of a depth-first traverse of a DAG which is in $O(|vertices| + |edges|)$, because the transitive closure, the computation of classes extensions and the topological order can all be computed in this way. Indeed, the transitive closure can be done by aggregation of pushed nodes and sets of instance during a depth-first traverse. Moreover, a topological order is provided by the reverse sequence of nodes traversed by a postfix depth-first traverse of a DAG. Therefore the preprocessing phase is in $O(n_i + n_j + m_i + m_j)$.
- The classification phase is separated into the training step and the proper classification. We assume that $TClass$ the complexity of the classification of one instance by a trained classifier is O(1). We denote $TTraining(N)$ the complexity of the training step with a set of examples and counter-examples of size N.
  - As there is a training step for each class on N examples and counter-examples in the worst case, the training is in
    $O((n_i + n_j) \times TTraining(N))$
  - All instances of $\mathcal{T}_i$ which do not belongs to $\mathcal{T}_j$ are classified in all classes of $\mathcal{T}_j$ and vice-versa, so the proper classification has a complexity of

$O((N - N_i)n_i + (N - N_j)n_j)) = O(N(n_i + n_j))$

The classification phase has a whole complexity of
$O((n_i + n_j) \times TTraining(N) + N(n_i + n_j))$.

- The core algorithm is constituted by the nested loops of $n_i \times n_j$ steps. Inside these loops, all the operation are in $O(1)$ except the pruning functions:
  1. IMPLICANTS_LHS has a complexity of $O(n_j)$ in the worst case in which $\mathcal{T}_j$ is a linear graph and the function is called from the top class.
  2. IMPLICANTS_OPTIMIZED has an overall complexity of $O(m_i + m_j)$ as seen in Proposition V.8

  Therefore the complexity of the core algorithm is in
  $O(n_i n_j(1 + n_j) + m_i + m_j) = O(n_i n_j^2 + m_i + m_j)$.

The overall complexity of the ProbaMap algorithm is then :

- $O(n_i n_j^2 + m_i + m_j)$ without classification
- $O(n_i n_j^2 + m_i + m_j + (n_i + n_j)TTraining(N) + N.(n_i + n_j))$ with classification

Algorithm 2 ProbaMap

Require: Taxonomies (DAG) $\mathcal{T}_i, \mathcal{T}_j$, thresholds S_c, S_i, Instances maps $inst_i, inst_j$, instances sets I_i, I_j, instances features maps $desc_i, desc_j$, boolean $ClassEnabled$

Ensure: return $\{m = A_i \sqsubseteq B_j, \mathcal{T}_j)/A_i \in Nodes(\mathcal{T}_i), B_j \in Nodes(\mathcal{T}_j)$, and $\widehat{P_i}(m) \geq S_i$ *and* $\widehat{P_c}(m) \geq S_c\}$

1: // ——Preprocessing——
2: Sequence $Topo_i \leftarrow$ ORDER_TOPO($\mathcal{T}_i$)
3: Sequence $Topo_j \leftarrow$ ORDER_TOPO($\mathcal{T}_j$)
4: DAG $TClo_j \leftarrow$ CLOSURE($\mathcal{T}_j$)
5: Map $ext_i \leftarrow$ INFER($inst_i, TClo_i$)
6: Map $ext_j \leftarrow$ INFER($inst_j, TClo_j$)
7: // ——Classification——
8: **if** $ClassEnabled$ **then**
9: $\quad extCL_i \leftarrow$ CLASSIFICATION($\mathcal{T}_i, ext_i, desc_i, I_i, I_j, desc_j$)
10: $\quad extCL_j \leftarrow$ CLASSIFICATION($\mathcal{T}_j, ext_j, desc_j, I_j, I_i, desc_i$)
11: **end if**
12: // ———Core———-
13: Set $M_{Val} \leftarrow \emptyset$
14: Set $M_{NVal} \leftarrow \emptyset$
15: **for all** $C_i \in$ reverse order of $Topo_i$ **do**
16: $\quad$ **for all** $D_j \in$ direct order of $Topo_j$ **do**
17: $\quad\quad$ let $m = C_i \sqsubseteq D_j$
18: $\quad\quad$ **if** $m \notin M_{NVal}$ **then**
19: $\quad\quad\quad$ **if** $ClassEnabled$ **then**
20: $\quad\quad\quad\quad Plhs \leftarrow$ESTIMATION_PCLASS($lhs(m), extCL_i, |I_i \cup I_j|$)
21: $\quad\quad\quad\quad Pjoint \leftarrow$ ESTIMATION_PJOINT($lhs(m), rhs(m), extCL_i, extCL_j, |I_i \cup I_j|$)
22: $\quad\quad\quad$ **else**
23: $\quad\quad\quad\quad Plhs \leftarrow$ESTIMATION_PCLASS($lhs(m), ext_i, |I_i|$)
24: $\quad\quad\quad\quad Pjoint \leftarrow$ ESTIMATION_PJOINT($lhs(m), rhs(m), ext_i, ext_j, |I_i \cap I_j|$)
25: $\quad\quad\quad$ **end if**
26: $\quad\quad\quad p_i \leftarrow 1 - Plhs + Pjoint$
27: $\quad\quad\quad$ **if** $p_i \geq S_i$ **then**
28: $\quad\quad\quad\quad p_c \leftarrow \frac{PJoint}{Plhs}$
29: $\quad\quad\quad\quad$ **if** $p_c \geq S_c$ **then**
30: $\quad\quad\quad\quad\quad M_{Val} \leftarrow M_{Val} \cup \{m\}$
31: $\quad\quad\quad\quad$ **else**
32: $\quad\quad\quad\quad\quad M_{NVal} \leftarrow M_{NVal} \cup$IMPLICANTS_LHS($m, TClo_j$)
33: $\quad\quad\quad\quad$ **end if**
34: $\quad\quad\quad$ **else**
35: $\quad\quad\quad\quad M_{NVal} \leftarrow M_{NVal} \cup$IMPLICANTS_OPTIMIZED($m, \mathcal{T}_i, \mathcal{T}_j$)
36: $\quad\quad\quad$ **end if**
37: $\quad\quad$ **end if**
38: $\quad$ **end for**
39: **end for**
40: return M_{Val}

Algorithm 3 IMPLICANT_LHS

Require: Mapping $m = A_i \sqsubseteq B_j, TClo_j$
Ensure: return implicants of m with the same left-hand side

```
Set ToPrune ← {m}
// For all C_j descendants of B_j
for all Class C_j in Successors(B_j, TClo_j) do
    ToPrune ← ToPrune ∪ {A_i ⊑ C_j}
end for
Return ToPrune
```

Algorithm 4 IMPLICANTS_OPTIMIZED

Require: Mapping $m = A_i \sqsubseteq B_j$ Taxonomy (DAG) $\mathcal{T}_i, \mathcal{T}_j$
Ensure: return mappings to be pruned from m

```
static variable Set StrongPruned: mappings that are pruned in this procedure
Set ToPrune ← ∅ // the result variable returned by this function
Stack S ← ∅
if m ∈ StrongPruned then
    Return ∅
end if
Push(S, m)
while Size(S) ≠ 0 do
    Mapping mc = C_i ⊑ C_j ←Pop(S)
    StrongPruned ← StrongPruned ∪ {mc}
    ToPrune ← Pruned ∪ {mc}
    Set Pred ← (⋃_{A'_i ∈ Predecessors(C_i, T_i)} A'_i ⊑ C_j) ∪ (⋃_{B'_j ∈ Successors(C_j, T_j)} C_i ⊑ B'_j)
    for all Mapping m' ∈ Pred do
        if m' ∉ StrongPruned and m' ∉ S then
            Push(S, m')
        end if
    end for
end while
return ToPrune
```

Algorithm 5 CLASSIFICATION

Require: $\mathcal{T}_i, \mathcal{T}_j, TClo_i, ext_i, I_i, I_j, desc_i, desc_j$ (set of instances)
Ensure: returns classified extensions for class of $\mathcal{T}_i$ with instances of $\mathcal{T}_j$

```
extCL_i ← ext_i
for all C_i ∈ classes of T_i do
    PreprocessedDesc ←PREPROCESS(desc_i, desc_j, C_i)
    Exemples ← ⋃_{I∈ext(C_i)} descriptions(I)
    CounterEx ← Instances_i \ Examples
    Classifier ←TRAIN(Examples, CounterEx, PreprocessedDesc)
    Set Instances_ToClass_j ← I_j \ I_i
    for all Instance J ∈ Instances_ToClass_j do
        if CLASSIFIER(PreprocessedDesc(J)) = True then
            for all D_i ∈PREDECESSORS(C_i, TClo_i) ∪ C_i do
                extCL_i(D_i) ← extCL_i(D_i) ∪ {J} // For all ancestors of C_i
            end for
        end if
    end for
end for
return extCL_i
```

Part II

Experiments

CHAPTER

SIX

EXPERIMENTS ON CONTROLLED SYNTHETIC DATA

This part on experiments aims at evaluating the ProbaMap algorithm and the interest of the probabilistic confidence functions P_i and P_c. Firstly, this chapter introduces a thorough analysis of quantitative and qualitative results of ProbaMap conducted on synthetic data. Secondly, the Chapter VII reports results on real-world datasets, allowing to evaluate the scalability and the quality of the output in a real-world setting.

For evaluating the quality of our results, we use the standard criteria of Precision and Recall [VR75]:

- Recall is the ratio of returned results that are expected w.r.t. all expected results.
- Precision is the ratio of returned results that are expected w.r.t. all returned results.

The measured features of ProbaMap during this analysis are mainly:

- the Precision and Recall of the returned results as qualitative measures
- the ratio of mappings that are pruned during the algorithm execution and the computation time as quantitative measure.

For the purpose of systematic testing in various conditions, we have evaluated ProbaMap on synthetic data on which important parameters can be tuned to guarantee structural or distributional properties: for example, we can control the sizes of the taxonomies and so the size of the search space, the number of mappings to discover, the number of instances per class, the noise in the annotated data.

We have crossed these input parameters with the ProbaMap paramaters in order to measure the whole impact these parameters whatever the features of the input data. The ProbaMap parameters are the thresholds S_i and S_c and the kind of classifier used for the instance merging phase.

We first describe the principles and the process of the data generator on which we have conducted the different experiments. Then we describe the experimental protocol that we have followed, before presenting all the corresponding results.

VI.1 Synthetic data generation

The synthetic data generation is a central issue for many scientific challenges. In our case it allows to control the input data and expected results in order to evaluate ProbaMap in a systematic way that can be considered independent of the application domain or bias in data.

Data generation in our context represents quite challenging issue when compared to other synthetic data generation, such as functional Armstrong relation for functional dependencies[1] [BDFS84], or transactional databases for frequent itemsets [RMZ03].

We are faced with the following hard challenges: (1) generating both the *structure* and the instances of taxonomies, and (2) generating the mappings to be discovered. Roughly speaking, we borrow the same principles than those developed for Armstrong relations [Fag82]. Each generated mapping should be satisfied by the generated data, and each mapping that is not generated should be contradicted by the generated data.

With regard to the Armstrong relations, our generation tackles additional issues. Firstly, the structure may be generated whereas Armstrong relations suppose the structure (schemas) given. Secondly, for relational databases, it is enough to generates two tuples that contradict one dependency for ensuring that the dependency is not satisfied. In our case where mapping probabilities are estimated from statistics on class extensions, the amount of instances that contradict a mapping has a strong impact on its validity, then we can not only generate one instance for each mapping to be contradicted.

Synthetic data generation is divided into three steps: generation of taxonomies with fixed sizes, generation of the expected mappings to discover, and population of each class by generating a fixed number of instances and associated description.

For a fair evaluation of performance, it is important to control the distribution of the taxonomies that are provided as inputs to ProbaMap. For doing so, we rely on an existing model for the random generation of combinatorial structures which is mathematically grounded ([DFLS04]).

For evaluating Precision and Recall of the mappings returned by the ProbaMap, we have to compare them to a reference, so the generation step should include the generation of mappings to be discovered. Thus, we generate the expected mappings and we force the extensions of the classes to respect all the consequences of the generated mappings according to the taxonomies, and only them. In other words, the instance descriptions should be coherent with the generated taxonomies and mappings by respecting these two properties:

(PC) - Positive Coherence: the instances and their descriptions respect all the logical entailments between classes which can be logically inferred from the generated mappings and taxonomies (so-called *expected entailments*)

(NC) - Negative Coherence: the instances and their descriptions contradict all the logical entailments between classes which can not be inferred from the generated mappings and taxonomies (so-called *unexpected entailments*)

Therefore, below are the required specifications for our synthetic generator:

[1] An Armstrong relation for a set of functional dependencies is a relation that satisfies each dependency implied by the set and does not satisfy any dependency that is not implied by it.

Specification VI.1 (Generation of populated taxonomies and mappings):
Given $(n_1, n_2, m_S, n_p) \in \mathbb{N}^{*4}$*, the generation process should provide:*

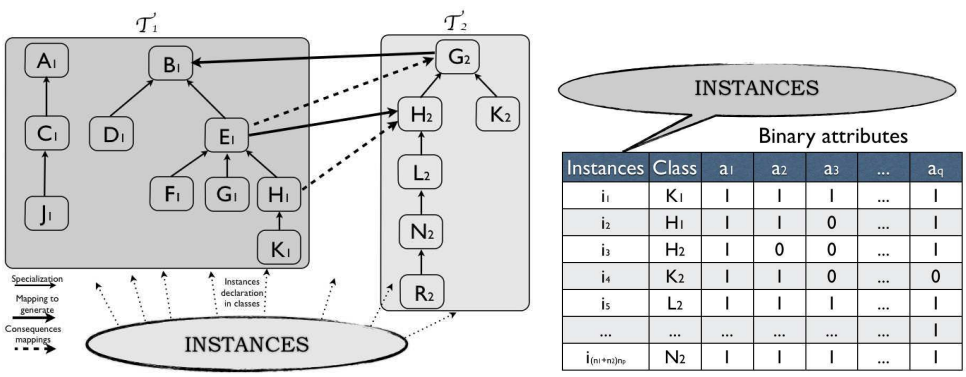
- *Two taxonomies* $\mathcal{T}_1, \mathcal{T}_2$ *of respective size* n_1 *and* n_2
- $\mathcal{M}_S$*: a set of* m_S *distinct most specific mappings between* $\mathcal{T}_1$ *and* $\mathcal{T}_2$ *(in both directions)*
- *For each class* C *of* $\mathcal{T}_1$ *and* $\mathcal{T}_2$*, a set* $inst(C)$ *of* n_p *distinct instances only declared as instances of* C*.*
- *For each instance* I*, a binary description vector* $desc(I)$*. All description vector have the same length* q *(not fixed* a priori*).*

such that the descriptions of instances respect:

- *the property of Positive Coherence (PC)*
- *the property of Negative Coherence (NC) except for a statistically small and controlled set of entailments*

We have designed a method for generating the attribute descriptions of the instances in the taxonomies which guarantees that the only relations that can be inferred by those descriptions are those entailed by the generated class inclusions and the generated mappings. This is achieved by the generation of enough distinct attributes to characterize each class inclusion stated in the generated taxonomies and mappings.

We now describe the different steps of the data generation before proving that this generation respect the two properties of coherence. Generation is a three steps process: taxonomies generation, mapping generation, and instances generation with their descriptions.



Instances	Class	a_1	a_2	a_3	...	a_q
i_1	K_1	1	1	1	...	1
i_2	H_1	1	1	0	...	1
i_3	H_2	1	0	0	...	1
i_4	K_2	1	1	0	...	0
i_5	L_2	1	1	1	...	1
...	...	...	...	...	...	1
$i_{(n_1+n_2)n_p}$	N_2	1	1	1	...	1

(a) Taxonomies and mappings (b) Instances and attributes

Figure VI.1: Example of result provided by the generation process

Generation of taxonomies

Given constraints on number of classes n_1 and n_2, the structure of the two taxonomies $\mathcal{T}_1$ and $\mathcal{T}_2$ is generated as a forest of general trees (unconstrained in-degrees) by using a Boltzmann sampler for unlabelled trees as explained in [DFLS04]. We have instanciated the proposition made in [DFLS04], from which the following algorithms have been designed. Algorithm 6 generates random general trees with a random size n, with the particularity that each tree of size n has the same probability to occur than other of the same size. In Algorithm 7, a reject method is used to get random rooted trees with a given number of node. Each new node provided by NEW_NODE of generated tree is automatically labelled by a new label by the function NEW_LABEL that generates a novel string label. The parameter x used in the generation is set to the particular value $\frac{1}{4}$ in order to better distribute the sizes of all possible generated trees, and optimize the reject method. Hence for instance, in order to obtain a random general tree of size n, the following call should be done: GENTREEOFSIZE($n, \frac{1}{4}$). A general tree T is represented by a pair $(Root, SubtreesSet)$ where $Root$ is the root node of T and $SubtreesSet$ the set of all subtrees of T. As there is no labelling in this step, the order of subtrees does not matter.

Algorithm 6 GENTREEREC

Require: Generation parameter $x \in [0; \frac{1}{4}]$, max size m
Ensure: returns a random general tree of size $\leq m$ or interrupts itself

```
global integer count
real A_x ← (1−√(1−4x))/2
if count + 1 > m then
    INTERRUPT;
end if
Node NewRoot ←NEW_NODE(NEW_LABEL()))
List SubTrees ← ∅
while RANDOMUNIFORM([0; 1]) ≤ A_x do
    SubTrees ← SubTrees∪GENTREEREC(x, m) // add a new subtree to the set of subtrees of the generated tree
end while
Return (NewRoot, SubTrees)
```

Algorithm 7 GENTREEOFSIZE

Require: Size S, Real $x \in [0; \frac{1}{4}]$
Ensure: Returns a random general tree of size S with an uniform probability among all general trees of size S

```
global integer count
repeat
    TreeGenTree ← EmptyTree
    count ← 0
    GenTree ← GENTREEREC(x, S) // GenTreeRec can be interrupted, in this case GenTree remains empty.
until SIZE(GenTree) = S
Return GenTree
```

To obtain taxonomies as forests with n_1 and n_2 nodes, we call GenTreeOfSize($n_1 + 1, \frac{1}{4}$) and GenTreeOfSize($n_2 + 1, \frac{1}{4}$) and we remove the root node in the two provided trees then we label each node by a distinct class name. This method is simple and bounded in $O(n^2)$ where n is the required size, while guaranteeing an uniform distribution among the trees with the same number of nodes. In our experiments, we generally set $n_1 = n_2$ so the two taxonomies have the same size, which is the unique parameter of the taxonomies generation.

Mapping generation

We initialize the generation of mappings to be discovered $\mathcal{M}_G$ with a set $\mathcal{M}_S$ of seed mappings with a size of m_S.

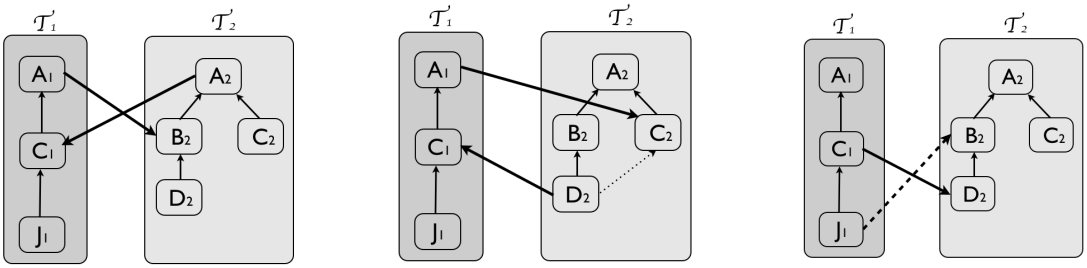
Each mapping $m \in \mathcal{M}_S$ is generated by a random choice for the two classes $lhs(m)$ and $rhs(m)$ in $\mathcal{T}_1$ and $\mathcal{T}_2$, or in $\mathcal{T}_2$ and $\mathcal{T}_1$, depending on the mapping direction which is randomly chosen too.

There are three constraints on generated mappings:

1. Seed mappings should not introduce any cycle in the graph constituted by the two taxonomies plus the mappings. Figure VI.2(a) illustrates such a situation.

2. Seed mappings which logically entails class inclusions that are not entailed within each taxonomy are rejected. In other words, we forbid generated mappings to modify the knowledge of each taxonomy. Figure VI.2(b) illustrates this situation: the dotted arrow in $\mathcal{T}_2$ represents a specialization relation which is a consequence of initial taxonomies and generated mappings.

3. Seed mappings do not entail each other according to the taxonomies. On Figure VI.2(c) , the mapping $J_1 \sqsubseteq B_2$ (dashed arrow) is entailed by the mapping $C_1 \sqsubseteq D_2$ and can not been generated as a seed mapping if $C_1 \sqsubseteq D_2$ is already a seed mapping.



(a) Cycle (b) Internal consequence (c) Mappings entailment

Figure VI.2: Counter-examples on constraints applied to mappings generation

The set $\mathcal{M}_G$ of all mappings to discover will then be the set of mappings that can be logically entailed by $\mathcal{M}_S$ and the two taxonomies. Note that mappings of $\mathcal{M}_G$ and by construction $\mathcal{M}_S$ do not introduce any cycle nor modify the knowledge of taxonomies.

We added a logical specificity parameter for the generation of mappings, that indicates a requested number of consequences for seed mappings:

- a *high* level of specificity forces each seed mappings to have at least h consequences mappings, where h depends on the taxonomies depths. For generating each seed mapping mapping $A \sqsubseteq B$, it is achieved by taking A among the highest classes in the first taxonomy, and by taking B among the deepest classes in the second taxonomy.

- a *middle* level of specificity forces each seed mappings to have at least l consequences mappings and at most h consequences mappings, where l, h depend on the taxonomies depths. For generating each seed mapping mapping $A \sqsubseteq B$, it is achieved by taking A among the classes that have a middle-height in the first taxonomy, and by taking B among the classes that have a middle-depth in the second taxonomy.

Again, the specialization generated in taxonomies, the generated mappings and all their consequences are those called *expected* ones. All the possible entailment between two classes of all the classes of the taxonomies that are not expected are those called *unexpected* ones.

Instances and description generation

For this step, we consider the two taxonomies and the mappings between them as a DAG of classes. The absence of cycle in that graph is guaranteed by the constraints imposed in the production of the set $\mathcal{M}_G$ of generated mappings described above.

We first generate a set of boolean attributes sufficient to associate a minimal intentional description of each class C for which generated instances will conform to.

The core of the principle is that the intentional descriptions should respect the semantic partial order $\preceq$ conveyed by the above DAG structure.

Then, we use this intentional knowledge to generate a given number of instances for each class C according to the description C.

Generation of the intentional description of classes:
We traverse the DAG of classes according to a reverse topological order [CLRS01] starting from the most general classes that constitute the level 0, and we iterate the following process for generating the intention of classes as sets of attributes:

- For each class C_i^0 of level 0, we generate a disjoint set of distinct attributes $\mathcal{A}t_i^0$ and we set the intention of C_i^0, denoted $Int(C_i^0)$, to be At_i^0.

- For each class C_i^j of level j (according to the reverse topogical order), we generate a set $\mathcal{A}t_i^j$ of novel attributes (disjoint from the set of existing attributes) with a size fixed to the out degree of C_i^j in the DAG of classes, and we set $Int(C_i^j)$ to be $\mathcal{A}t_i^j \cup \bigcup Int(C_{i_k}^{j-1})$, where the $C_{i_k}^{j-1}$ are the successors of C_i^j in the DAG.

  A result for two taxonomies $\mathcal{T}_1$ and $\mathcal{T}_2$ is shown in Figure VI.3. The classes that have no parents (A_1 and B_1) constitute the level 0 and their intentional description are initialized respectively by the attributes a and b. Then the intentional description of C_1 can be generated because it has only A_1 as parent, and A_1 has just been processed. The intentional description for C_1 constitutes of the union of the attributes of its parents in the DAG of

classes, that is A_1 (attribute $\{a\}$) plus a number of novel attributes equal to its outgoing edges, so one, we will call c. Therefore, the intentional description of C_1 is $\{a, c\}$. Further in the process, the intentional description of E_1 should contain the intentional description of its two parents B_1 and H_2, plus two novel attributes e, e' because there are two outgoing edges. It leads to the description $\{b, e, e', g, h\}$.

Intuitively, a novel attribute for an entailment relation in the DAG should be considered as a specialization: the more the intentional description is large, the lower the number of possible total descriptions is, and so the less probable are these descriptions to happen.

As we adopt the idea that one class with two parents is specialized twice and so more specific that one class with only one of both parents, the number of possible descriptions for instances of a class should be reduced twice. This is achieved by adding two novel attributes in its intentional description instead of one. This principle is generalized to n parents (outgoing degree) in our generation process.

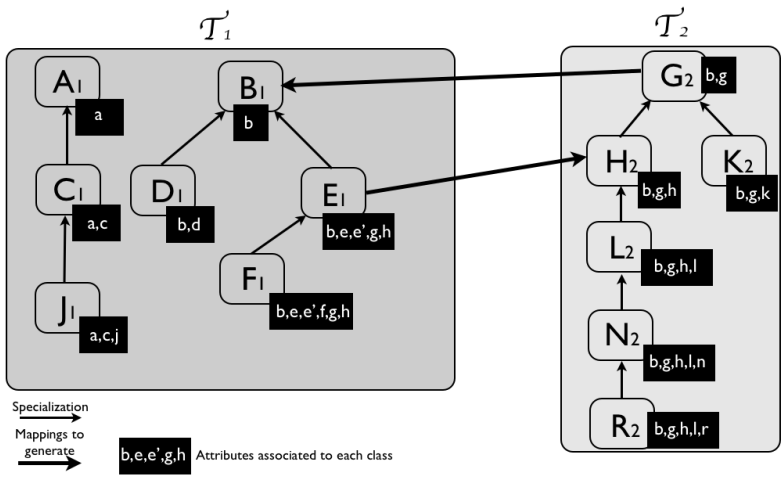


Figure VI.3: Example of generation of intentional descriptions for each class

Population of classes:
Let $\{At_1, \ldots, At_q\}$ be the set of attributes generated at the previous step. We populate each class with n_p instances, and we associate to them descriptions that respect the corresponding intentional description, as follows: For each class C, each of its instances is described by a boolean vector $[a_1, \ldots, a_q]$ obtained by:

- setting to 1 each a_i such that the corresponding attribute At_i is in the intention of the class C,
- randomly setting the other values a_j to 0 or 1.

This way, by construction, all the instances in the extension of a class have in common that all the attributes in $Int(C)$ are present in their description.

The results of the data generation can be summarized into a table T_{data} with $n_A + n_1 + n_2$ attributes, where each tuple $[a_1, \ldots a_q, c_1, \ldots, c_{n_c}]$ concatenates the description $[a_1, \ldots a_q]$ of an instance in terms of attributes, and its categorization $[c_1, \ldots, c_{n_c}]$ with respect to the classes: for each $i \in [1..n_C]$ c_i is equal to 1 if $i \in Ext(C)$ and to 0 if it is not the case.

Correctness of the generation w.r.t. the required Specification VI.1

We need a reference to characterize a correct generation of instances according to the Specification VI.1. In particular the (PC) and (NC) constraints should be instantiated with a precise meaning for the fact that instances and their descriptions should *respect* all the logical entailment between classes which can be inferred from the generated DAG.

For doing that for (PC), we will say that the instances and their descriptions respect such a constraint if some particular reference estimations for P_c and P_i are very close to 1 for each entailment in the DAG. This reference estimation is based on an oracle classifier that we introduce here:

Definition VI.1 (Oracle classifier):
Given two generated taxonomies (with the set of all classes denoted $\mathcal{C}$), generated mappings, and intentional descriptions of classes and their common length q, the oracle classifier *is a function $O : [0;1]^q, \mathcal{C}\ rightarrow[0;1]$ for which:*

- $O([v_1, \ldots, v_q], C) = 1$ *if* $[v_1, \ldots, v_q]$ *fits the intentional description of* C.
- $O([v_1, \ldots, v_q], C) = 0$ *otherwise*

For each entailment formula between classes $A \sqsubseteq B$, the oracle classifier can be used to obtain the respective extensions of A, B and then $A \cap B$ on the two taxonomies. We denote $\widehat{P_c}^O(A \sqsubseteq)$ and $\widehat{P_i}^O(A \sqsubseteq B)$ the estimations of $P_c(A \sqsubseteq B)$ and $P_i(A \sqsubseteq B)$ using such classified extensions.

The oracle classifier will be the reference used to characterize a correct generation of instances according to the specifications VI.1. Therefore, Positive and Negative coherence properties (PC) and (NC) based on the oracle classifier become the following constraints:

(PO') - the oracle estimations $\widehat{P_c}^O$ and $\widehat{P_i}^O$ should return a value very close to 1 when applied to an expected entailment relation according to the generated taxonomies and mappings

(NO') - Conversely, $\widehat{P_c}^O$ and $\widehat{P_i}^O$ should return a value distant from 1 for all entailments between classes that are unexpected

More formally these constraints are expressed as follows:
There exist $\epsilon \in [0;1]$ and a function $n_p \longrightarrow f(n_p)$ with $\lim_{n_p \to \infty} f(n_p) = 1$ (n_p denote the number of generated :instances by class) such that:

(PO) - $\widehat{P_c}^O(A \sqsubseteq B) \geq 1 - \epsilon$ for each $A \sqsubseteq B$ expected entailment,

(NO) - $P(\widehat{P_i}^O(A \sqsubseteq B) \leq 1 - 2\epsilon) \geq f(n_p)$ for each unexpected entailment

The Theorem VI.1 asserts that the generator respects the Specification VI.1 according to the oracle, in particular that the generator respects the two above constraints (PO) and (NO).

Theorem VI.1 (Correctness of the generation w.r.t. its specifications):
The output of the generation process is correct with regard to the Specification VI.1 and the oracle classifier.

The proof can be found in Appendix A in Section A.2.2. It uses the following property which extends the relation $P_c(m) \leq P_i(m)$ to their estimations with the oracle classifier. The proof for this property is written in Appendix A, Section A.2.1.

Proposition VI.1:
Given any formula of the form $A \sqsubseteq B$ where A, B are two classes, the following inequality holds:

$$\widehat{P_c}^O(A \sqsubseteq B) \leq \widehat{P_i}^O(A \sqsubseteq B)$$

VI.2 Experimental protocol

We first recall the goals of the successive experiments we have performed on the synthetic data: the first goal is to analyze the impact of the thresholds S_c, S_i on the quality of the result. This experiment permits to fix appropriate thresholds for the next experiments. The second goal is to analyze the impact of multiple parameters on the pruning ratio of ProbaMap, in order to determine what causes ProbaMap to be more or less efficient. In particular we study the influence of taking only one probability P_c or P_i versus both of them for the validity criterion. We will consider generation parameters as the balance between the two taxonomies in term of their size, and the specificity level of the generated seed mappings (measuring their respective amount of consequences).

The third goal is to analyse and compare the impact both on Precision/Recall and on total running time of three real classifiers (Naive Bayes, C4.5 and SVM) for estimating the probabilities. The purpose is to determine the classifier offering the best trade-off between quality of results and running time. Note that we do not take the learning time of classifiers into account because we consider that this task can be precomputed for each taxononomy.

The last goal with synthetic data is to analyse the robustness of the approach to noisy data.

For all the experiments on synthetic data presented in this section, each point is obtained by averaging the results of 100 runs of ProbaMap, each involving two runs of Algorithm 2 (page 71), one for each direction of mappings: for two taxonomies $\mathcal{T}_i, \mathcal{T}_j$ the search space is first $\mathcal{M}(\mathcal{T}_i, \mathcal{T}_j)$ then $\mathcal{M}(\mathcal{T}_j, \mathcal{T}_i)$. For each of these 100 runs, a new synthetic dataset is generated with the parameters set. Note that in our experiments we generate taxonomies with few dozens of classes. The number of random taxonomies of such sizes can be counted in billions. Thus, averaging over 100 runs for a point does not prevent from local variations, leading to curves that are not smooth.

Our algorithm is written in Java and compiled using Sun Java version 1.6. We run all the tests on a quad-core Intel Q6700 Xeon at 2.66 GHz with 4 GB of memory. The OS is Ubuntu Linux

8.10. For all the experiments measuring run times, only one instance of our program and the OS are running on the machine, to avoid memory contention effects with other programs that would affect the results.

Reference used for computing Precision and Recall

Following [Euz07], the computation of precision and Recall will be based on $\mathcal{M}_G$, the set of all consequences of the generated mappings $\mathcal{M}_S$ according to $\mathcal{T}_1, \mathcal{T}_2$. Let R be the result of ProbaMap. The Recall is the proportion of mappings of $\mathcal{M}_G$ actually returned by ProbaMap:

$$Recall = \frac{\mathcal{M}_G \cap R}{\mathcal{M}_G}$$

The Precision is the proportion of returned mappings that are actually in $\mathcal{M}_G$:

$$Precision = \frac{\mathcal{M}_G \cap R}{R}$$

Summary of parameters

There are two kinds of parameters for these experiments: the parameters for the generation, and the parameters for ProbaMap. There are summarized in tables below.

Parameter for generation	description	range	default value
Size of the search space	candidate mappings: $2n_1n_2$	100 - 8000	-
Balance	$n_1 = n_2$ or $n_1 = 10n_2$	balanced/unbalanced	balanced
Specificity of seed mappings	amount of consequences	middle/high	middle
Number of instances per class		10 - 200	50

Parameter for ProbaMap	description	range	default value
S_i	threshold for P_i	[0;1] (0 for bypass)	0.90
S_c	threshold for P_c	[0;1] (0 for bypass)	0.85
Classification	Name of the classifier	Oracle/NB/C4.5/SVM	Oracle

Except for a few experiments, the main parameter in the x axis will be the size of the search space, i.e. the number of candidate mappings. Therefore, all parameters for which we measure the impact are crossed with the size of the search space which is a major point in our approach.

VI.3 Experimental results

Impact of thresholds on Precision

We compare the influence of the thresholds S_c and S_i associated to probabilities $\widehat{P_c}$ and $\widehat{P_i}$ on the quality of the results returned by ProbaMap.

In this experiment, the computation of probabilities is performed using the oracle classifier. The parameters in the synthetic generator are defined such that $|\mathcal{M}(\mathcal{T}_1, \mathcal{T}_2)| = 320$. We set the

number of seed mappings $|\mathcal{M}_S| = 4$. Note that by logical entailment the total number $|\mathcal{M}_G|$ of mappings to be discover may be much greater. For each pair of threshold $S_c, S_i \in [0.78; 0.995]$, we compute the Precision and the Recall of the results of ProbaMap. We observed that the Recall remains constant at 1.0 independently of values of S_c and S_i. This is because thanks to the oracle classifier, estimated probabilities for the mappings of $\mathcal{M}_G$ are very close to 1, and superior to all experimented thresholds, leading to a perfect Recall. Thus, we only show the results for Precision in Figure 3.

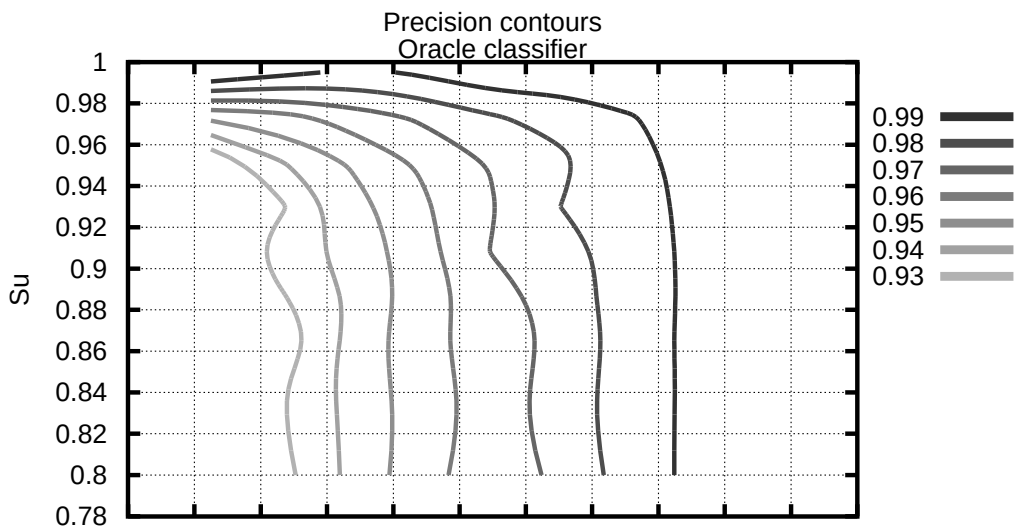


Figure VI.4: Precision w.r.t. S_c and S_i thresholds

Figure VI.4 shows the contours of the different Precision levels, from a precision of 0.93 to a precision of 0.99. From the shape of these contours, it is clear that both $\widehat{S_c}$ and $\widehat{S_i}$ and then the used probabilities have an influence on Precision. As the relation $\widehat{P_i} \geq \widehat{P_c}$ holds (Proposition VI.1), under the diagonal $\widehat{P_i}$ has no influence on Precision.

The probability $\widehat{P_c}$ is more discriminant than $\widehat{P_i}$. The figure shows that $\widehat{P_c}$ influences the Precision for a large range of values of the threshold S_c, while $\widehat{P_i}$ only has an influence for very high values of S_i. We have observed that the distribution of values of $\widehat{P_c}$ for valid mappings overlap less the distribution of values of $\widehat{P_c}$ for invalid mappings that those for $\widehat{P_i}$. The statistic gap between valid and invalid mappings is larger for P_c than for P_i.

$\widehat{P_i}$ gives higher probability values to invalid mappings, this explains why it can only have an influence on Precision at very high S_i values. In addition, it should be noted that S_i should be higher than S_c to have an influence on testing the candidate mappings, because $P_c(m) \leq P_i(m)$ for any mapping m.

Based on all these observations, we can conclude that it is interesting for the quality of results (Precision, here) to use a combination of both probabilistic confidence functions P_c and P_i.

Based on the curve of Figure VI.4, we set the thresholds at $(S_c = 0.83, S_i = 0.96)$ for experiments where the classifier used to estimate the probabilities is the oracle. This gives a good Precision

of 0.95, and maps to a region where $\widehat{P_i}$ has an influence on the quality of results.

For the experiments in which a real classifier is used to estimate the probabilities, we relax the thresholds at $(S_c = 0.85, S_i = 0.90)$ to be tolerant to classification errors.

Parameters influencing the pruning factor

Here we expose an analysis of the behaviour of the ProbaMap optimization part according to different parameters. Except for the influence of the classifier, all experiments here are done with the oracle classifier in order to consider unbiased situations.

Impact of the used probability functions on pruning factor

We now study the impact of the choice of the probability functions that are involved for testing the validity on the pruning factor. We denote by *pruning factor* the ratio of number of calls to the estimation functions ESTIMATION_PJOINT and ESTIMATION_PCLASS used for P_c and P_i estimations, with regard to the number of calls with a disabled pruning. That corresponds to consider all candidate mappings, i.e. $2|\mathcal{M}(\mathcal{T}_i, \mathcal{T}_j)|$ calls.

Figure VI.5 shows the pruning factor obtained using either only one of each probability model by setting the threshold of the other to 0, or both of them. The reference is the naive approach that does no pruning by testing all candidate mappings.

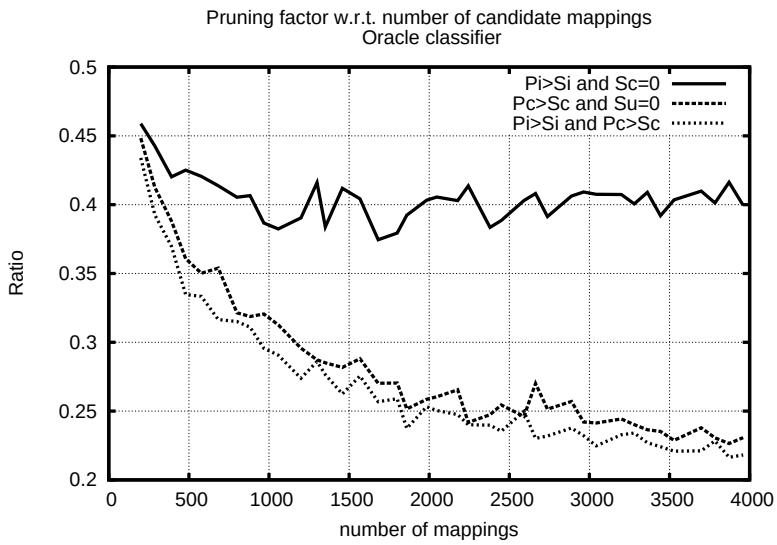


Figure VI.5: Pruning factor with only $\widehat{P_c}$, only $\widehat{P_i}$ and $\widehat{P_c}$ *and* $\widehat{P_i}$

ProbaMap version using only $\widehat{P_i}$ (by setting S_c to 0) is the one that prunes the least mappings, computing probabilities for about 40% of all candidate mappings. Both versions using $\widehat{P_c}$ and ($\widehat{P_c}$ *and* $\widehat{P_i}$) does more pruning and obtain a significant reduction of the search space. Combining $\widehat{P_i}$ and $\widehat{P_c}$ obtains slightly better results than using $\widehat{P_c}$ alone, so for the remainder of this chapter, we set S_c and S_i to non-zero real numbers. This is a very positive result because the combination

of P_i and P_c allows to prune more than 75% of the search space (for search space larger than 2500 mappings), that is very time-saving, while improving the Precision (as seen in the previous experiment).

Impact of balance of taxonomies on pruning factor

Figure VI.6 shows the pruning factor with regard to the size of the search space, in a balanced situation where the sizes of the two taxonomies are equal, and in an unbalanced situation where the size of $\mathcal{T}_1$ is ten times the size of $\mathcal{T}_2$. In the unbalanced situation ProbaMap slightly

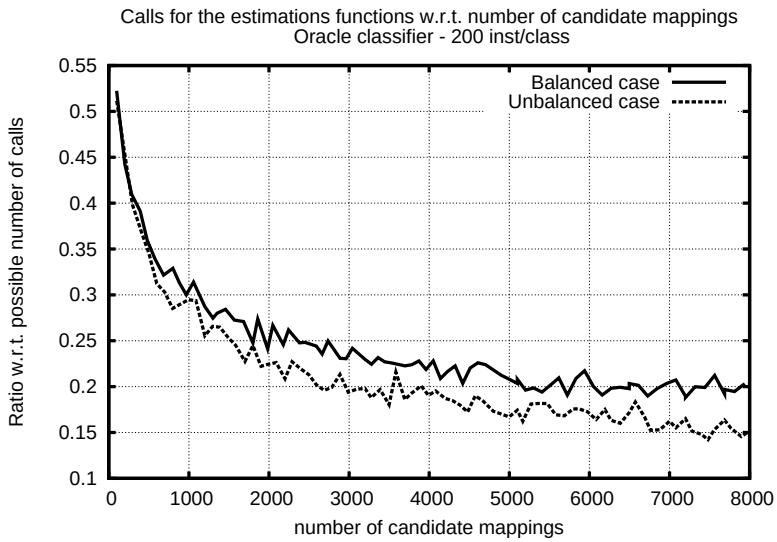


Figure VI.6: Pruning factor for unbalanced and balanced taxonomies

outperforms itself in the balanced situation. It is useful to know that the balance does not affect the optimization of ProbaMap.

Impact of specificity of seed mappings on pruning factor

Comparative results for high and middle specificity of mappings are provided in Figure VI.7. When the mappings to find are more specific, the pruning is less important. It can be easily seen while considering that ProbaMap generates mappings from the most general to the most specific, and can only prune implicants of mappings evaluated mappings. If the mappings to find are more specific, there are more valid consequences mappings, and so less pruning to do before finding the most specific that are valid.

Impact of the classifiers on time and quality of the results

In this subsection, we replace the oracle classifier with a real classifier. We compare the results given by three well-known classifiers: Naive Bayes [Mit97], C4.5 [Qui93] and SVM [FL02]. We use the Weka [WF05] implementation of these classifiers and have interfaced it with our code.

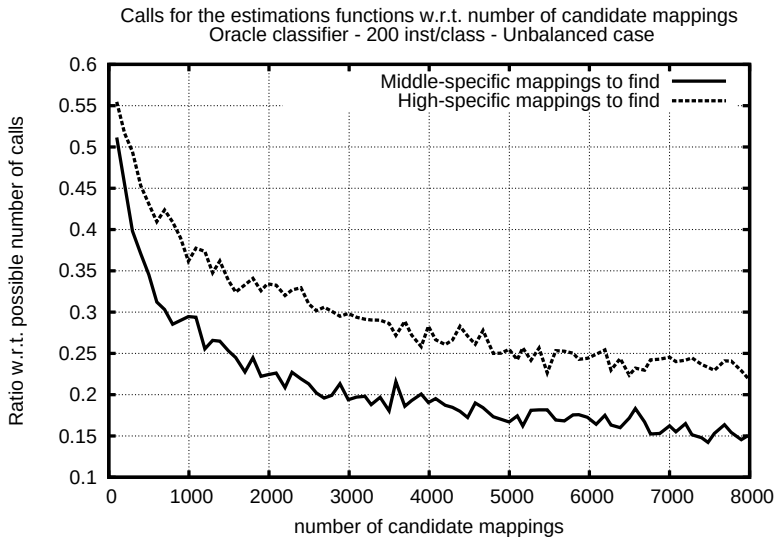


Figure VI.7: Pruning factor w.r.t. specificity of seed mappings

Impact of the choice of real classifiers on pruning factor

Figure VI.8 shows that the used real classifier does not affect significantly the pruning factor, whatever the size of search space. It is a good point related to the robustness of our optimization.

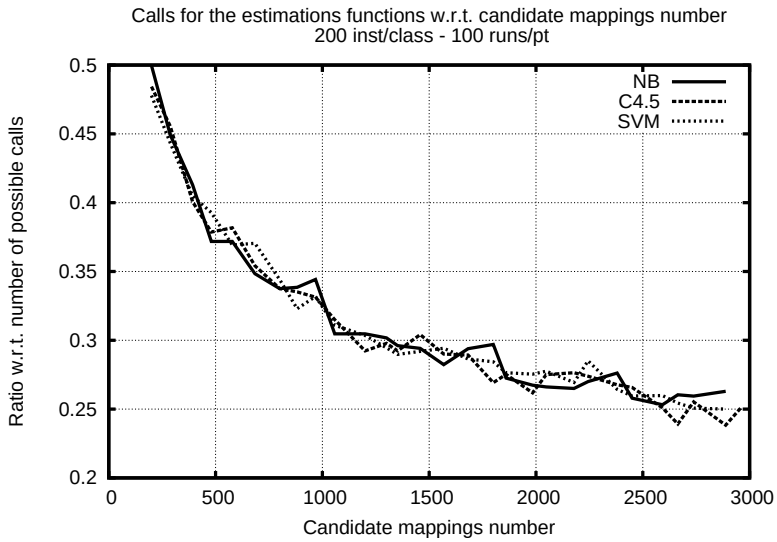


Figure VI.8: Pruning factor for different classifiers

Impact of the choice of real classifiers on time

The comparisons of running times are shown in Figure VI.9 and in log scale in Figure VI.10.

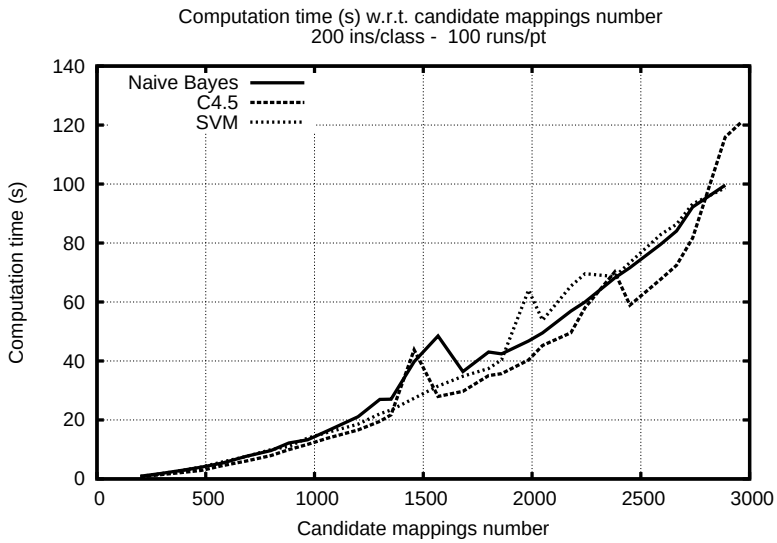


Figure VI.9: Computation time (s) for different classifiers

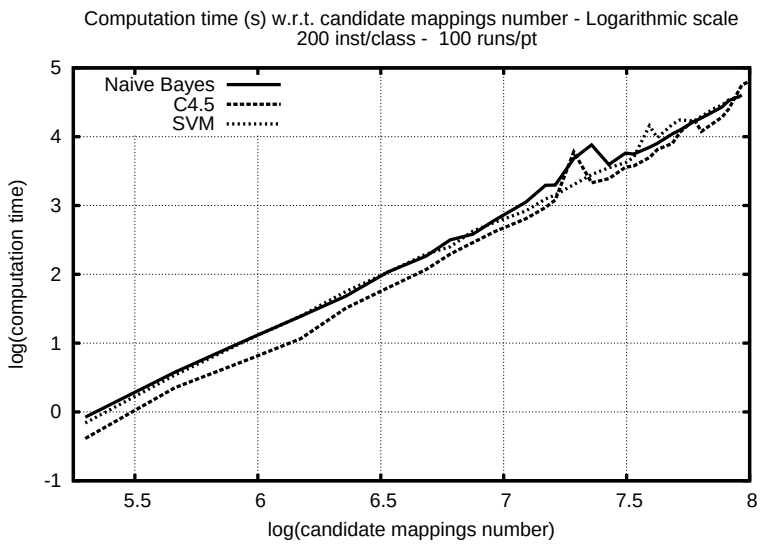


Figure VI.10: Computation time in log scale for different classifiers

A first conclusion is that the running times are polynomial in the number of mappings. But in our generator, the size of taxonomies is correlated to both total number of instances and length of the attribute description q. Thus by increasing the size of the search space and so of the taxonomies, there are three parameters that are changing in Figures VI.9 and VI.10. Thus, we can not derive from them a complexity formula which would be only dependent of the search space.

A second conclusion is that changing classifier does not have a significant effect. Naive Bayes is slightly slower than C4.5 and SVM.

Impact of the choice of real classifiers on quality

Comparisons for Precision and Recall are respectively shown in Figures VI.11 and VI.12. Whatever the classifier, Precision is not impacted by the number of candidate mappings, i.e. the size of search space. This is also the case for the Recall, except when the classifier is Naive Bayes.

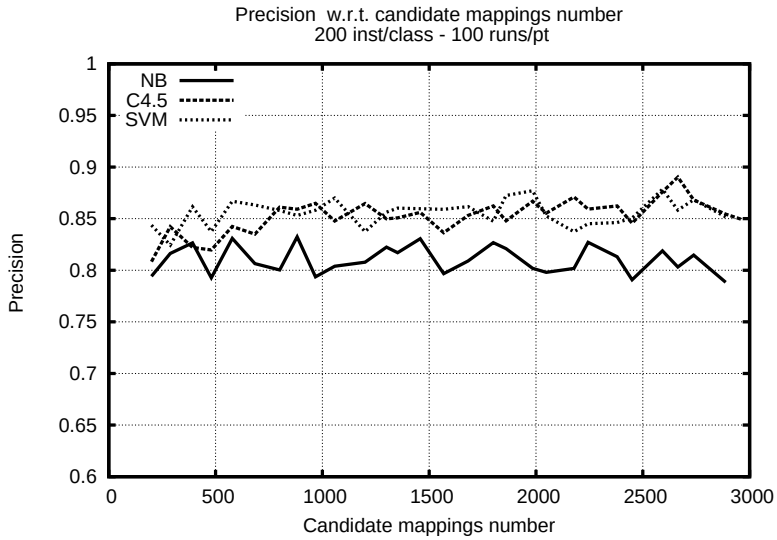


Figure VI.11: Precision for different classifiers

Naive Bayes has both the worst Recall and the worst Precision, the choice is thus between C4.5 and SVM. They seem to have similar results. We thus choose C4.5 for further experiments because it has a training time shorter than SVM (not taken into account in Figure VI.9).

We vary the number of instances per class n_p between 10 and 450. The results for computation time, Precision and Recall are shown in Figures VI.13, VI.14 and VI.15.

In this experiment, the number of classes and of mappings is constant, hence the number of classifications to perform is linear in the number of instances. The C4.5 algorithm takes linear time in the number of instances. As expected, this is also the case for Algorithm 1, as shown by Figure VI.13. Increasing the number of instances per class only increases slightly Precision, whereas it strongly improves Recall. The most important point to note is that excellent values of Precision and Recall are obtained with as few as 50 instances per class, as expected, with a use of a Bayesian approach of statistics.

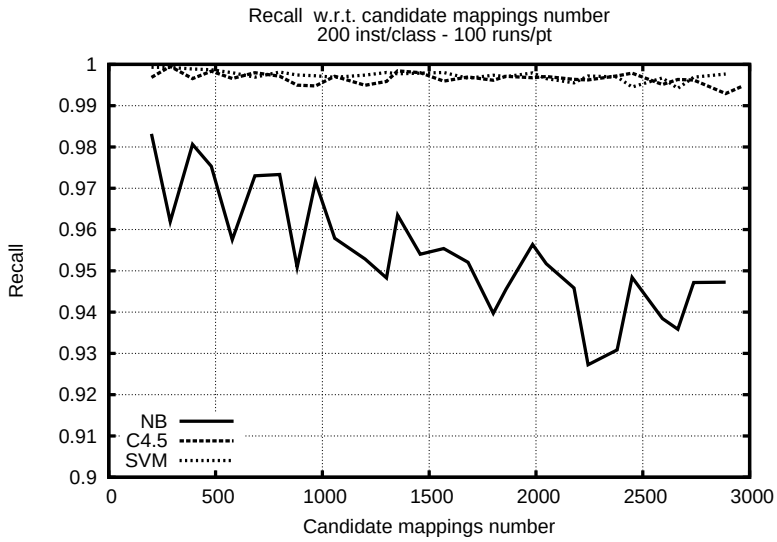


Figure VI.12: Recall for different classifiers

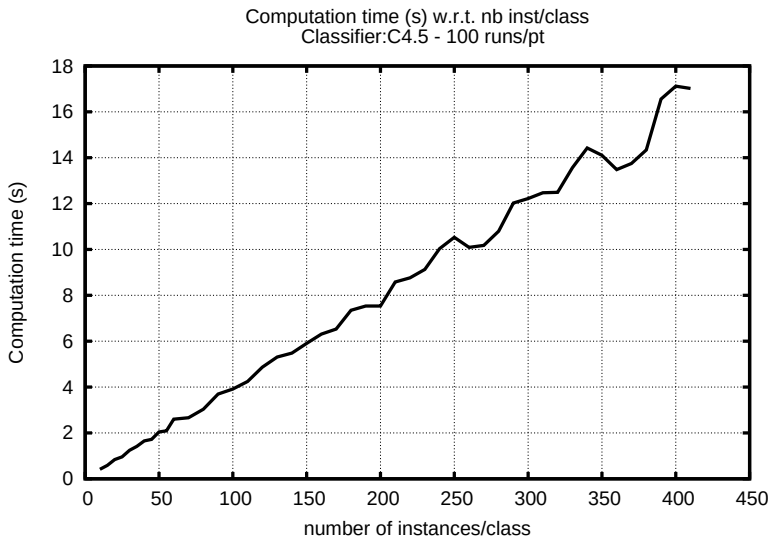


Figure VI.13: C4.5 - Computation time (s) - impact of number of inst/class

Robustness to noisy data

In order to test the robustness to noise of our algorithm, we define a new parameter θ corresponding to the quantity of noise to inject in the synthetic data. Each dataset produced by the synthetic data generator goes through a step of noise application in attributes. Each boolean corresponding to the value of an attribute for an instance can be reversed with a probability θ. This is applied for all attributes of all instances. The new dataset is then processed as usual by ProbaMap.

The variations of Precision and Recall for values of $\theta \in [0; 0.3]$ are show in Figure VI.16.

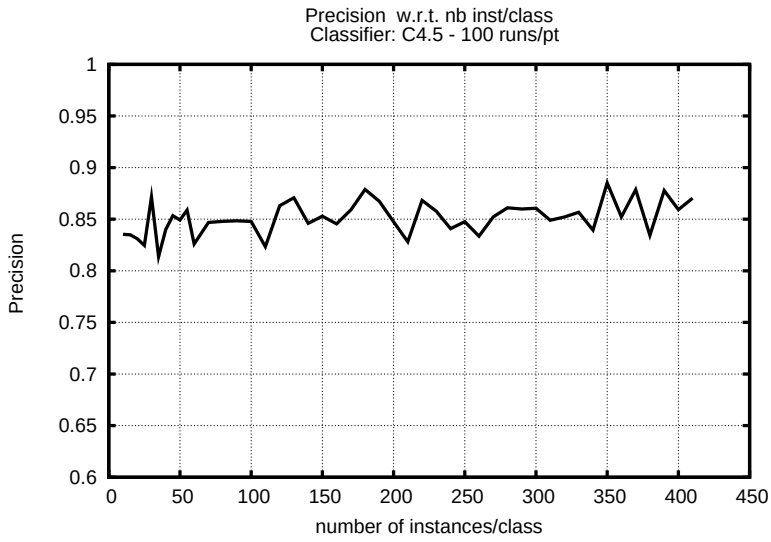


Figure VI.14: C4.5 - Precision - impact of number of inst/class

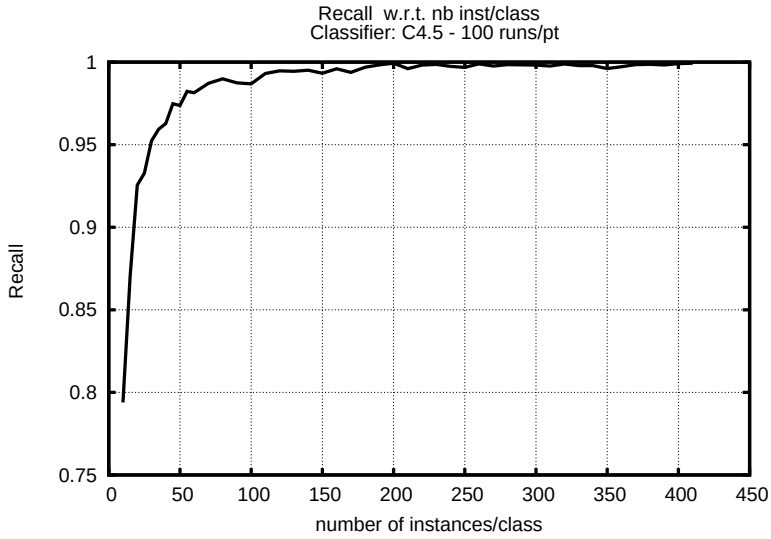


Figure VI.15: C4.5 - Recall - impact of number of inst/class

The figure shows that Recall gracefully degrades when noise increases. At 10% noise, the Recall is nearly unaffected, at a value of 0.95. Values of noise superior to 15% have a more significant impact and lead to poor Recall.

Precision, however, exhibits a different behavior. It first *increases* with noise, before abruptly decreasing for more than 24% of noise.

In order to understand this phenomenon, we have investigated in details the classifier results and the values of probabilities given to mappings. We found that for 0% noise, there are invalid mappings that are incorrectly given too high probabilities, and that appear as valid. This explains the non-perfect 0.88 Precision value. The probability values for these mappings

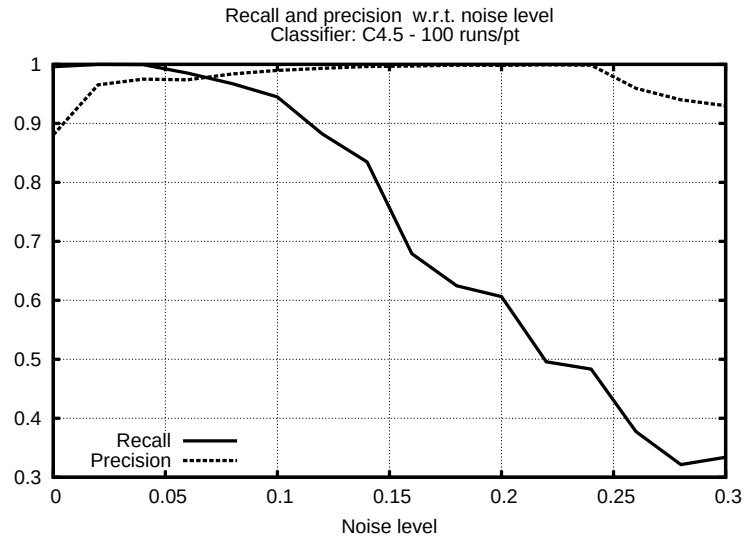


Figure VI.16: C4.5 - Precision and Recall - impact of noise level

are close to the threshold. Increasing noise makes the classifiers more selective, and tends to decrease the values of all probabilities. So the probabilities of these invalid mappings go below the threshold for a moderate amount of noise, whereas the probabilities of valid mappings remain above the threshold. Thus the Precision increases. This is an interesting result, that we might study further in a future work in order to better discriminate valid and invalid mappings. For example we could compare the mappings obtained by the original dataset and these obtained on the same dataset but with an additional noise, in order to determine which mappings resist to the noise.

In this chapter, we have introduced a synthetic data generator with good properties according to the distribution of generated taxonomies and the coherence between instances descriptions and the produced logical knowledge. We have exploited this generator for analysing the behaviour of ProbaMap with regard to the parameters available in generation: size and balance of taxonomies, degree of specificity of generated mappings, number of instances declared in each class, attribute noise. We have shown that combining the two probability models is useful to improve qualitative result, and allowing to do more pruning. The more the mappings to find are general, the more the optimization of ProbaMap is efficient. The choice of classifier does not influence the Precision of results, but C4.5 and SVM have significantly better Recall than Naive Bayes. Finally, ProbaMap is robust w.r.t. a large proportion of noise in instance description, and is not influenced by size variation of population in classes. Note that other matching systems could be used over our synthetic data.

CHAPTER

SEVEN

EXPERIMENTS ON REAL-WORLD DATA

After the detailed experimental analysis on synthetic data of Chapter VI, we confront ProbaMap with real-world data in this chapter.

For doing this, we Recall the vision about the future Semantic Web where each user or organization annotates their documents with taxonomies. This have led us to apply ProbaMap on taxonomies about large and popular domains, instead of attempting to align some expert-designed or specific domain taxonomies. Thus the two series of experiment that we present here are based on aligning parts of the Yahoo! and Google web directories.

Firstly we describe an experiment and some preliminar results on the OAEI Directory benchmark. As this benchmark containing no instances, we have made use of an external resource (WordNet) to compensate the lack of instances that ProbaMap needs.

Secondly we present a comparative experiment with the SBI [ITH03] method on aligning Yahoo! and Google subdirectories.

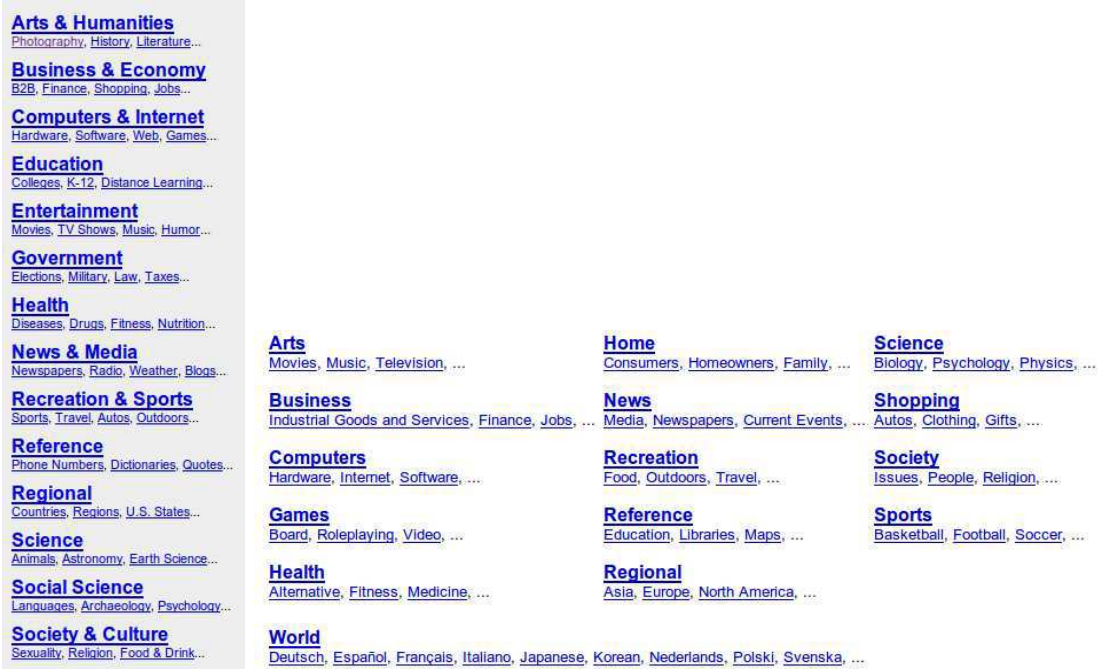
VII.1 OAEI directory benchmark

VII.1.1 Dataset and experimental settings

We have made experiments on the directory set of the Ontology Alignment Evaluation Initiative (OAEI) [EFH+09] contest. This benchmark for the 2009 edition has been created by the TaxMe 2.0 method presented in [GYAS09], by combining an automatic collection of data with a human process of discovery. This dataset is constituted by two large taxonomies of respectively 2857 and 6628 classes, extracted from the Yahoo![1], Google[2] and Looksmart[3] directories. Google directory is based on the Dmoz OpenDirectory project[4]. The top levels for Yahoo! and Google

[1] http://dir.yahoo.com
[2] directory.google.com
[3] http://www.looksmart.com
[4] http//www.dmoz.org

directories are pictured in Figure VII.1. A sample of these taxonomies is illustrated in Figure VII.2.



(a) Sample directory from Yahoo!

(b) Sample directory from Google

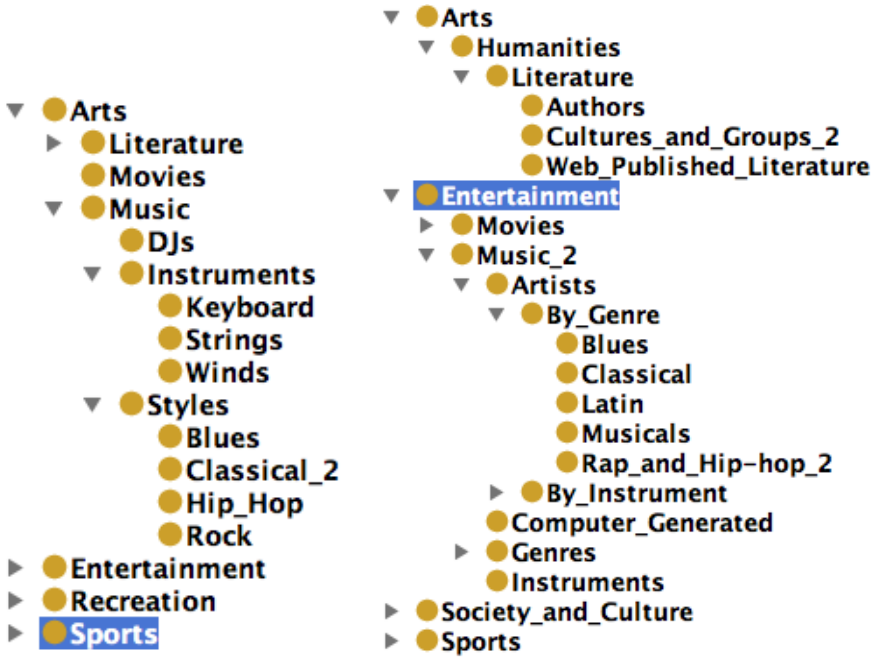
Figure VII.1: Top levels of Yahoo! and Google directories

In addition to the two provided taxonomies of classes, the Directory benchmark comes with the reference set of correspondences to be discovered by alignment methods. This set is constituted of equivalences between classes. We will detail below three ways to compare our result against this reference to measure Precision and Recall.

For the OAEI contest, due to scalability issues, the two initial taxonomies of the Directory benchmark are split into the set of their branches. A subset of more than 4,000 pairs of branches is also provided to participants as a modified dataset. Participants are invited to discover mappings between all pairs of branches: thus they have to match very small taxonomies but 4,000 times. Up to now, all participants have used this split version of the dataset whereas the full version has also been provided. In contrast, our algorithm is able to handle the two whole taxonomies, thus taking advantage of the complete structure of the taxonomies. It is important to note that without pruning, this would lead to a search space of more than 30 million mappings.

Evaluation method

The output of ProbaMap is constituted by *inclusion* mappings. We use three ways for evaluating this output against the reference set of equivalence correspondences:



(a) Sample directory 1 (b) Sample directory 2

Figure VII.2: Samples of the OAEI Directory benchmark extracted the two respective taxonomies to match

1. Standard Precision and Recall: Precision and Recall are computed with the classical formula, and the set of discovered correspondences is computed from the set of discovered mappings by these two options:

   (a) a correspondence $A \equiv B$ is discovered if and only if the two inclusion mappings $A \sqsubseteq B$ and $B \sqsubseteq A$ are discovered, i.e. iff
   $min(\widehat{P_c}(A \sqsubseteq B), \widehat{P_c}(B \sqsubseteq A)) > S_c$ and $min(\widehat{P_i}(A \sqsubseteq B), \widehat{P_i}(B \sqsubseteq A)) > S_i$

   (b) a correspondence $A \equiv B$ is discovered if and only if the respective arithmetic means of the probability estimations $\widehat{P_c}$ and $\widehat{P_i}$ in both directions exceeds their respective thresholds S_c and S_i,i.e. iff
   $mean(\widehat{P_c}(A \sqsubseteq B), \widehat{P_c}(B \sqsubseteq A)) > S_c$ and $mean(\widehat{P_i}(A \sqsubseteq B), \widehat{P_i}(B \sqsubseteq A)) > S_i$

2. Semantic Precision and Recall (inspired on [Euz07]) are computed based on a modified reference, that includes all inclusion mappings corresponding to the original reference plus the union of the consequences of each of them according to the taxonomies.

The semantic Precision and Recall seem to be more relevant for our approach focused on inclusion mappings and logical coherence. Indeed, including the logical consequences of the original reference mappings in the evaluation is fair and justified by the fact that any inclusion mapping discovered and entailed by the correct reference set should impact positively the Recall and the Precision.

Population of classes using WordNet

The OAEI Directory dataset does not contain any instance. For compensating the absence of available instances for these taxonomies, we use a method inspired by [SBMZ03] to automatically populate the classes with WordNet [C.98] synsets.

WordNet is an english thesaurus constituted by a set of *synsets*. Each synset represent a meaning of a word (or everal words). Synsets are connected to each other by a large set of semantic relations, like *HYPONYM* (which means "is a") or *MERONYM_PART* which means "part of". In addition, WordNet provides a way to find all possible synsets corresponding to a (possibly compound) word.

The principle of this population is based on associating with each class C a set of WordNet synsets that reflect the right meaning of the label of C in the context where it appears, i.e. the labels of the ancestor classes of C in its taxonomy. This help to disambiguate the meaning of a word: for instance, the label "Arizona" can correspond to a state of the U.S.A. or to a snake. If the Arizona class is a child class of "Animals", the label "Animals" can be used to guess that "Arizona" means the snake species.

More precisely for populating the class C, our population algorithm follows these principles:

1. To find all the possible meanings of the label of C
   (a) to find the main component of the label of C which has at least a corresponding synset in WordNet, for example:
      - "car" → car
      - "Rock'n roll" → Rock'n roll
      - "Software engineering" → Software
   (b) to query WordNet for all synsets of this main component and store it in the $Kernel$ set
   → *at this step all senses of the word are found, so it appears to be not sufficient to correctly populate C*
   (c) to query WordNet and to store in the $Kernel_Ext$ set all synsets connected to the synsets of $Kernel$ with a path of length bounded by a max_depth parameter, with relations *is a* or *part of* (more precisely, in WordNet : *HYPONYM, HYPONYM_INSTANCE, MERONYM_PART, MERONYM_MEMBER, MEROBYM_SUBSTANCE*).
2. To find the synsets related to the context of the label of C
   (a) to create the *context* set of synsets for C: constituted by all synsets $CS_1, \ldots, CS_n$ corresponding to all the words of the labels of the ancestors classes of C in its taxonomy, plus the words of the label of C that are not in the main component (e.g. "engineering" for "Software engineering").
   (b) to query WordNet to obtain for each context synset $CS_1, \ldots, CS_n$ all the connected synsets $CS_Ext_1, \ldots, CS_Ext_n$ by semantic relations ("is a" and "part of") with a path of max length max_depth.

3. Select synsets of $Kernel_Ext$ that are closest to all the synsets in $CS_1, \ldots CS_n$ in term of the shortest relation path in WordNet.

Note that for a compound label with a connector like "&, and, or", the label is split into all its parts leading to a population for each part. The population of the compound label is obtained by doing the union of the population of each part. Indeed, the semantics of a class labeled "Pop & Rock" should be the union of Pop music and Rock music.

The population phase can be tuned by two parameters:

1. the maximum depth of related synsets for each query of synsets related to a word
2. the size of the context (up to which ancestor to take the labels into account ?). May be set to 0 in order to ignore the context. Taking all the context correspond to unbound this parameter.
3. the selection level for the step 3 of the process.

max_{depth} and selection level have been tuned on multiple couple of branches, for example by trying to match the branch from the first taxonomy containing the class *Hip_Hop* with the branch of the second taxonomy containing the class *Rap_&_Hip-Hop*.

All classes can not be populated by this processing phase. For instance, a class labelled "Alphabetical_Order" can not be populated by our system, because there is no corresponding entry for the main word that constitutes it.

For tuning the population phase, we do some preliminary tests on small taxonomies for which we could compute the Precision (by hand).

Input characteristics and results are summarized in the following table:

$\|\mathcal{T}_1\|$	50 (total) - 49 (populated)
$\|\mathcal{T}_2\|$	102 (total) - 98 (populated)
Search space	9604 mappings
s_i	0.9
S_c	0.8
max_depth	12
Precision without context	64%
Precision with all the context	90%

We could not compute Recall for this preliminary test.

Using the context to populate the classes significantly increases the Precision, whatever the size of the context above three (included). Therefore the population process seems to be relevant and capture the real meaning of classes.

VII.1.2 Results

Data characteristics for this experiment (input data and algorithm run) on the whole directories are summarized in the Table VII.1 below.

Classes of $\mathcal{T}_1$ Classes of $\mathcal{T}_2$	2857 (total) - 2608 (populated) 6628 (total) - 5988 (in WordNet)
Classes not populated by WordNet Total instances number	887 (9,3%) 31724
Threshold S_i Threshold S_c	0.9 0.75
Size of the search space Pruning ratio Pruned mappings Valid mappings	31 233 408 mappings 4.2% 29 891 577 117 553 (0.2%)
Population time ProbaMap time	5h 11'

Table VII.1: Measures for the run of ProbaMap on OAEI directory dataset

As the references with which OAEI organizers computes Recall and Precision are only available for the split dataset, we add a postprocessing phase: for all pair of branches of the split dataset, we return mappings that are found by ProbaMap between respective classes of these branches. We noticed that more than 90% of mappings discovered by ProbaMap are lost during this step (for the partial reference, see below) and so they can not be evaluated.

Scalability result

On the two whole taxonomies, the population phase produces about 30000 instances and takes 5 hours while the mapping discovery algorithm itself only takes 11 minutes. These are very reasonable computational times for handling 30 million possible mappings.

Evaluation on the partial reference

For evaluating the quality of the set of mappings discovered by our algorithm, we were provided a partial reference by OAEI, which covers a subsets of the pairs of branches provided to participants. Based on this, we could compute an estimated Recall over the restriction involved by this partial reference, and we could only compute a lower bound of Precision. This is due to the fact that this partial reference contains only a subset of the mappings to discover, but is not complete.

The table in Figure VII.3 gives semantic Precision and Recall, and standard Precision and Recall computed with both "min" and "mean" versions, crossed with three values for the S_c threshold.

The line with the value "wordnet/regular" for Recall corresponds to the fact whether the evaluation takes into account the mappings of the partial reference that involves classes which could not been populated by WordNet. These mappings can not be discovered by ProbaMap in this setting. We can notice that the difference between the two evaluations (in respective both columns of Recall) can be neglected.

The first conclusion is that the Recall is very low for each parameter, and that the evaluation for the semantic measures are better than those for the standard measures.

Threshold Sc	Semantic measures			Standard measures (min)			Standard measures (mean)		
	Precision	Recall wordnet	Recall regular	Précision	Recall wordnet	Recall regular	Precision	Recall wordnet	Recall regular
0.6	41 %	27 %	26 %	47 %	3 %	2 %	37 %	3 %	3 %
0.7	50 %	24 %	23 %	58 %	2 %	2 %	50 %	3 %	3 %
0.8	60 %	20 %	19 %	67 %	2 %	1 %	60 %	2 %	2 %

Figure VII.3: Results based on the partial reference

The second conclusion is that increasing S_c leads to a very better standard and semantic Precision whereas the Recall slightly decrease. Selecting a threshold of 0.8 for S_c appears to be a better choice.

The third conclusion is that the evaluation with the equivalence correspondences computed from the mean or the min of probabilities does not differ significantly.

Finally, these results seem quite promising, for the thresholds S_i and S_c respectively set to 0.9 and 0.8 we obtained a lower bound of Precision of 67%[1].

Evaluation on the complete reference

Our results have been evaluated against the complete reference by one of the OAEI organizer (Juan Pane) out of the contest. With this complete reference, the Recall is 1.4% and the Precision is 43.4%. Semantic Recall is 17.5% and Semantic Precision is 48.3%.

Note that others methods results are around 60% for Precision and 50% for Recall[1]. But we could not compare us to others in term of Semantic Precision and Recall, then the comparison is not totally fair, especially for our approach which connects to the logical semantics of mappings.

There are two reasons that explain such bad qualitative results:

1. the overfitting of the population phase on subsets of the whole dataset.
2. the lack of robustness of the population phase with regard to the changes in the dataset in input
3. WordNet is a linguistic-oriented resource based on a relation graph. This characteristic makes it hard to do an adequation with the principle of probability estimations based on extensions of populated classes and their ratios. A good perspective may be to analyse it in the spirit of methods proposed in the survey about using WordNet for Ontology Matching [LS08], especially of methods like [Res99] that proposes an uniform and independent of the corpus link distance between two labelled concepts.

The second experiment shows that providing non-artificial instances of classes makes ProbaMap give better result.

[1]Fore more details, see `http://disi.unitn.it/~pane/OAEI/2009/directory/result/`

	Yahoo!		Google		shared
	classes	instances	classes	instances	instances
`Autos`	947	4406	967	6425	837
`Outdoors`	2428	5511	1441	13863	623
`Software`	323	2390	2395	30140	572
`Photography`	168	1851	321	3852	286

Table VII.2: Statistics on data collected from subdirectories on Yahoo! and Google

VII.2 Comparative analysis on collected Web Directories

In this section, we test ProbaMap on part of Internet directories from Yahoo! and Google (actually based on Dmoz) that are rooted with similar or very close label. These sub-directories are considered as taxonomies, and URLs referenced inside each class of the taxonomy as instances.

The main difference with the sequence of experiments in the previous section is that the dataset contains original instances that are collected with their taxonomies, avoiding to process an artificial population.

We compare our approach to the SBI algorithm of Ichise et al. [ITH03, IHT04], which is dedicated to the discovery of mappings between Internet directories, and the integration of such directories.

Internet directories are trees of categories, which can be seen as taxonomies, categories being the classes. Each category contains a set of links (i.e. URLs to web sites), which can be seen as the instances of the class. Each link comes with a small text summary, whose words can be seen as instance attributes for classification.

Our datasets are corresponding locations in the Yahoo! and Google directories, that have also been used in the experiments of [ITH03, IHT04]:

- Yahoo! : `Recreation / Automotive` & Google : `Recreation / Autos`
- Yahoo! : `Recreation / Outdoors` & Google : `Recreation / Outdoors`
- Yahoo! : `Computers_and_Internet/Software` & Google : `Computers/Software`
- Yahoo! : `Arts / Visual_Arts / Photography` & Google : `Arts / Photography`

The data from the directories was collected in June 2010, so is different from the data of [IHT04] and [ITH03] which was collected in Fall 2001.
Table VII.2 shows for each dataset the number of classes and instances in each class, and the number of instances shared between the Yahoo! and the Google directories. Two instances are considered shared if they correspond to the same URL in both directories. For a fair comparison, we have implemented both ProbaMap and the SBI algorithm in Java.

VII.2.1 Experimental protocol

Adaptation of ProbaMap to fit the SBI framework

As detailed in Chapter III, SBI takes as input one source taxonomy, one target taxonomy, and the instances declared for each class of both taxonomies. For each class C_s of the source taxonomy, SBI returns a rule $C_s \rightarrow C_t^{predicted}$ associating C_s to a target class $C_t^{predicted}$ in the target taxonomy.

In order to fit the evaluation framework of SBI, we added a postprocessing to ProbaMap to obtain a similar form of results, i.e. a set of unique rules for each class of the source taxonomy.

The complete process is the following:

1. Application of ProbaMap on $\mathcal{T}_1$ and $\mathcal{T}_2$
2. For each class C_1 of $\mathcal{T}_1$,
   among all C_2 for which the two mappings $C_1 \sqsubseteq C_2$ and $C_2 \sqsubseteq C_1$ have been discovered,
   select the class C_2 for which $min(\widehat{P_c}(C_1 \sqsubseteq C_2), \widehat{P_c}(C_2 \sqsubseteq C_1))$ has the highest value.
3. For each class C_1 of $\mathcal{T}_1$, if there is no rule for C_1, associate to C_1 the rule of its closest ancestor in $\mathcal{T}_1$

In this way we obtain an unique rule $C_1 \rightarrow C_2$ for each class of $\mathcal{T}_1$, like the SBI system.

Evaluation measure for quality

The goal of our experiments is to compare the quality of Internet directories alignement for ProbaMap and SBI.

For the discovery of mappings, ProbaMap and SBI receive a "training" set of instances which is a subset of the shared and annotated instances. The test set used for evaluation of the discovered mappings is constituted by the remaining instances among the shared ones. In the case where ProbaMap is set to use classification, the training set is extended with all the non shared instances.

The classification is performed using the SVM implementation SMO [FL02] in Weka [WF05], where the classification attributes for an instance are the words of its summary. The previous experiments shows that SVM and C4.5 have the better quality and that C4.5 is quicker than SVM. Nevertheless we conduct the experiment with SVM which is expected to performs better with the preprocessed data which is quite sparse.

The evaluation is done by using the test set of instances. Each instance of this set belongs to a class C_s of the source taxonomy. Hence, we can compare:

- the class $C_t^{predicted}$ of the instance, predicted by the output rule $C_s \rightarrow C_t^{predicted}$
- the class C_t in which the instance is declared in the target taxonomy (each instance of the test set is common to the two taxonomies)

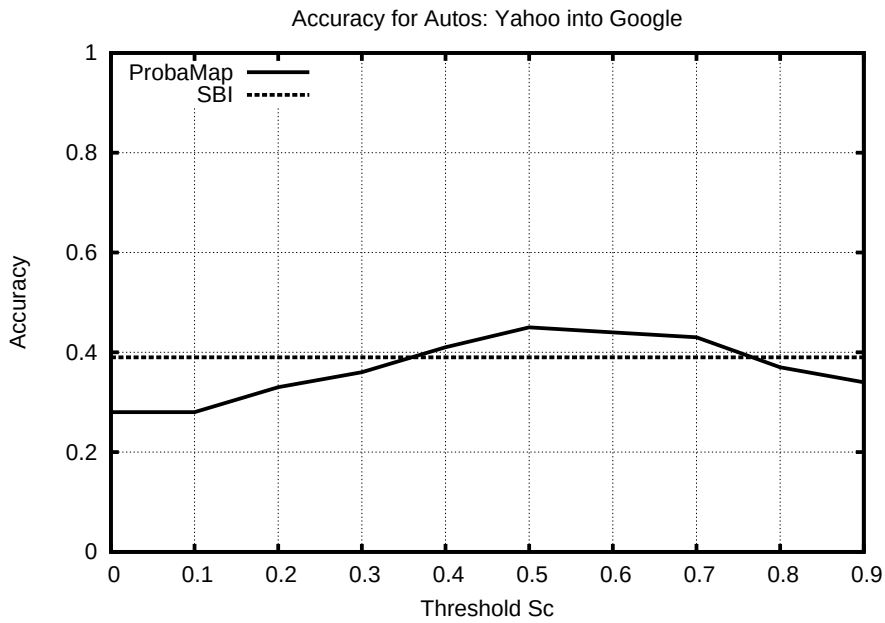
The *accuracy* measures the ratio of the instances in the test set for which $C_t^{predicted} = C_t$. Accuracy is a standard micro-averaged evaluation measure which is based on instances, whereas Precision and Recall used above are macro-averaged measures based on mappings themselves. As there is no reference of mappings provided for the considered Yahoo! and Google subdirectories, but a sufficient ratio of instances are shared by them. Therefore we use accuracy to evaluate ProbaMap in this experiment, and to fit the evaluation framework of SBI (used in [IHT04]).

VII.2.2 Results

Impact of the threshold S_c

A preliminary experiment was conducted to choose the best threshold S_c and s_i. As s_i has a lower impact for these experiments when being below 0.9, we set it to 0.9, in order not to over-tune our algorithm to the dataset.

Here we focus on the variation of S_c, in Figure VII.4.



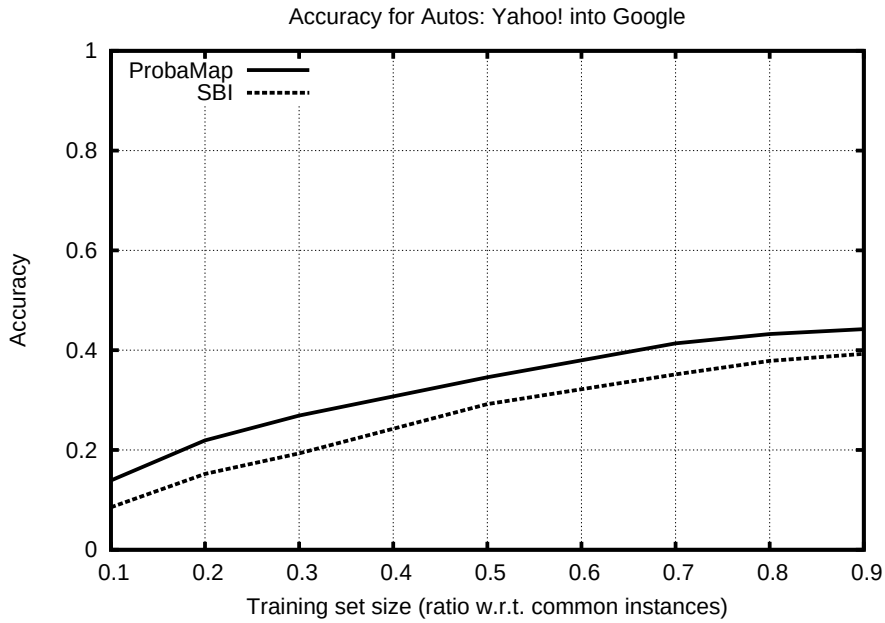
(a) `Autos`: Yahoo! into Google

Figure VII.4: Accuracy w.r.t. the threshold S_c

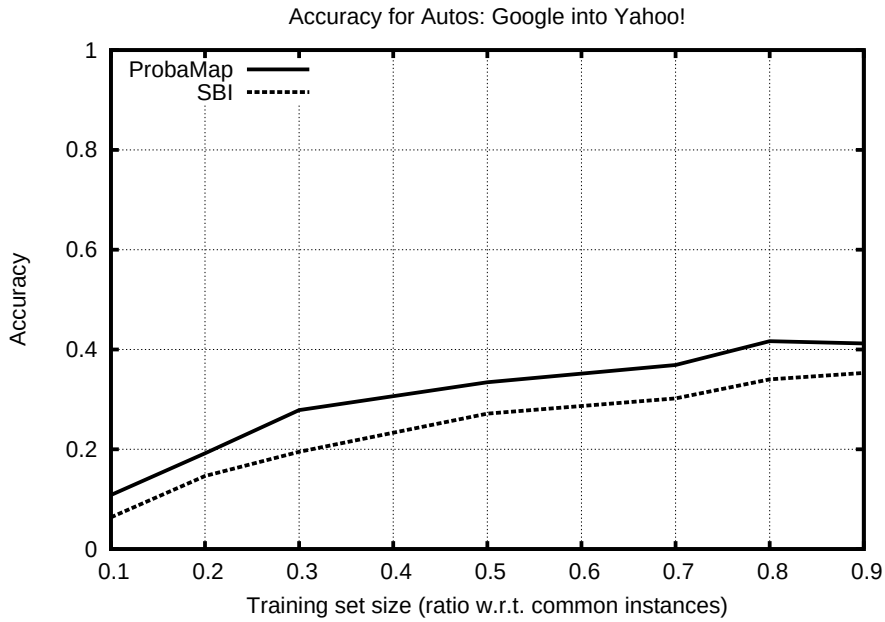
The best value for S_c is around 0.6 and we choose this value for the rest of the experiments. Remember that the rules are constituted by the more specific valid mapping, if it exists. Otherwise, it is the mapping for the parent class that is used (and so on). Therefore tuning the threshold corresponds to handle the trade-off between the risk of bad validated mappings (when S_c is too small) and the bad case where there is too many rules derived from the parent classes, when S_c is too high. For S_c very close to 1, all rules associate the class of the source taxonomy to the root class of the target taxonomy.

VII.2.2.1 Main comparative results

We computed the accuracy of mapping prediction on the test set, and conducted a ten-fold cross validation. The results when varying the size of the training set are shown in Figures VII.5, VII.6, VII.7, VII.8 for the fours dataset.

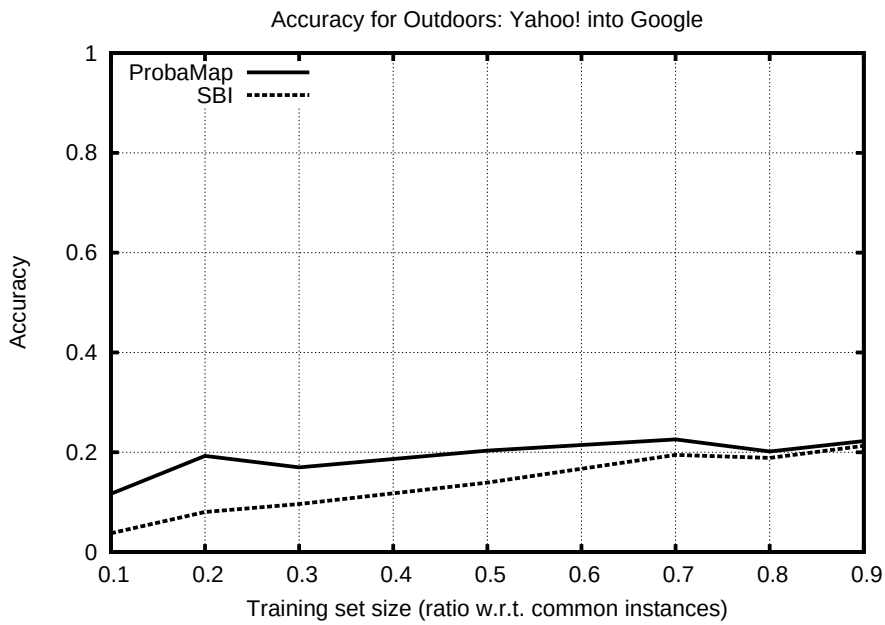


(a) `Autos`: Yahoo! into Google

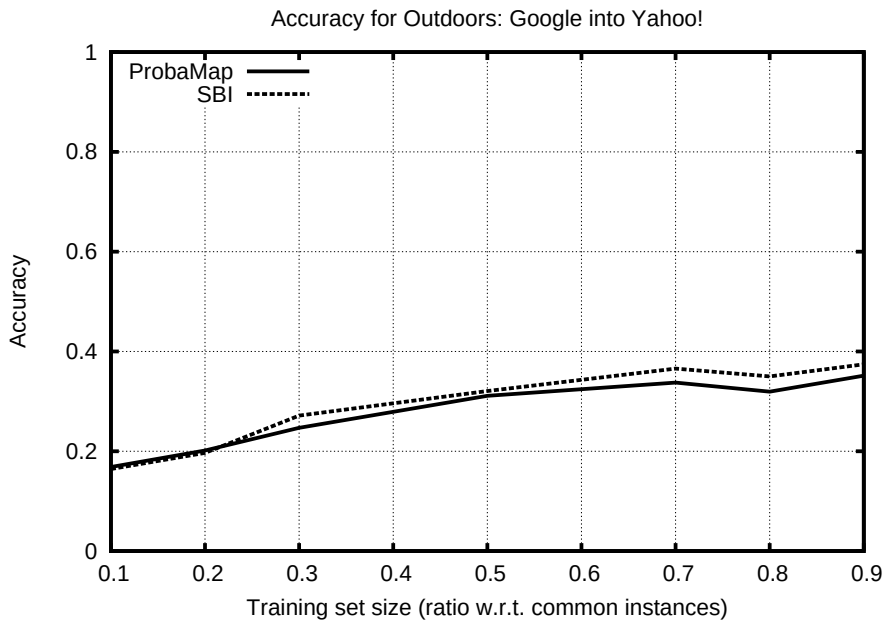


(b) `Autos`: Google into Yahoo!

Figure VII.5: `Autos`: Comparative accuracy for SBI and ProbaMap

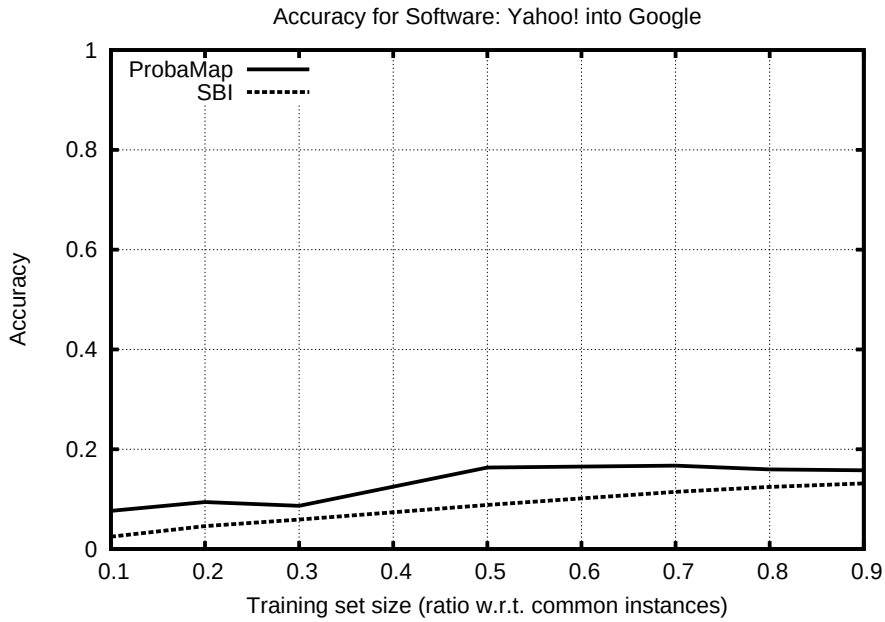


(a) `Outdoors`: Yahoo! into Google

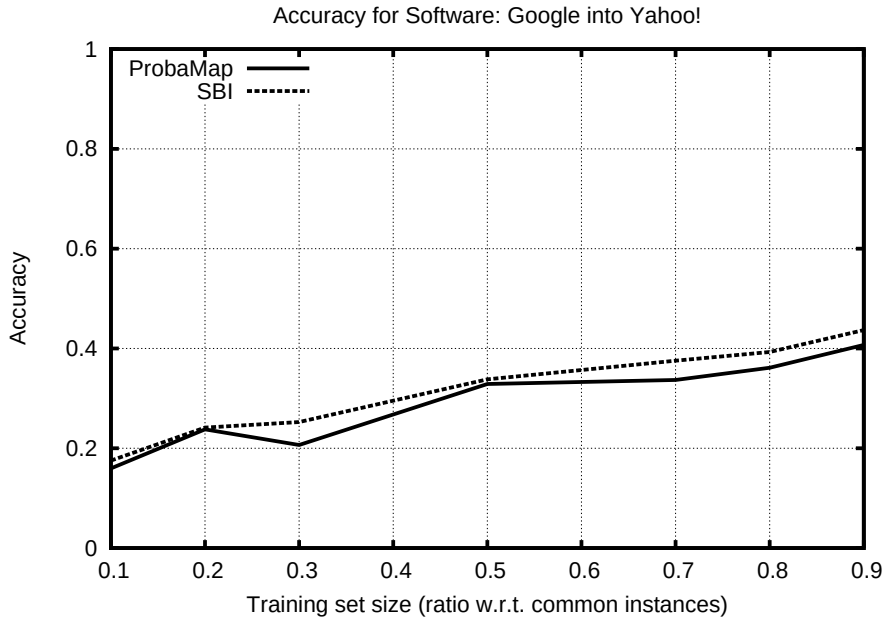


(b) `Outdoors`: Google into Yahoo!

Figure VII.6: `Outdoors`: Comparative accuracy for SBI and ProbaMap

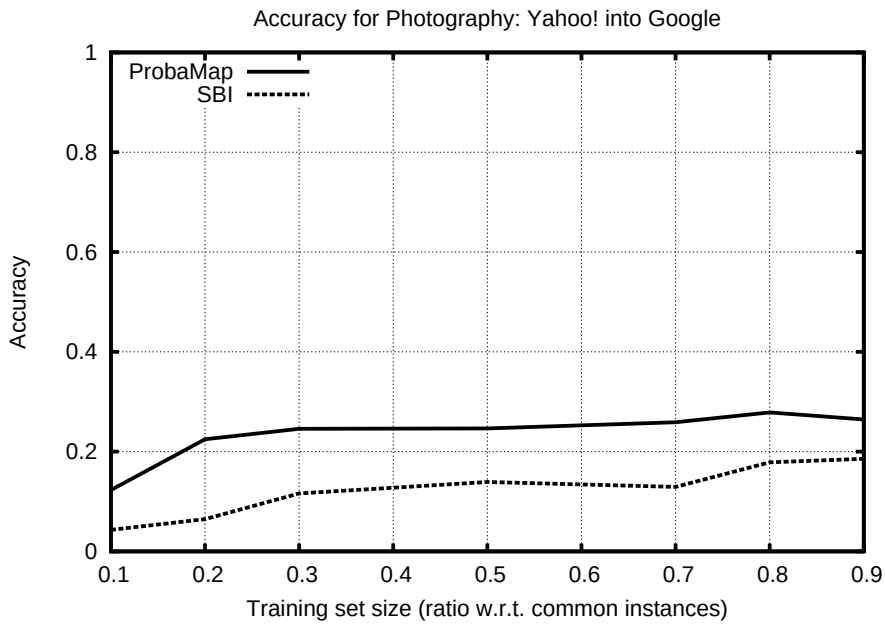


(a) `Software`: Yahoo! into Google

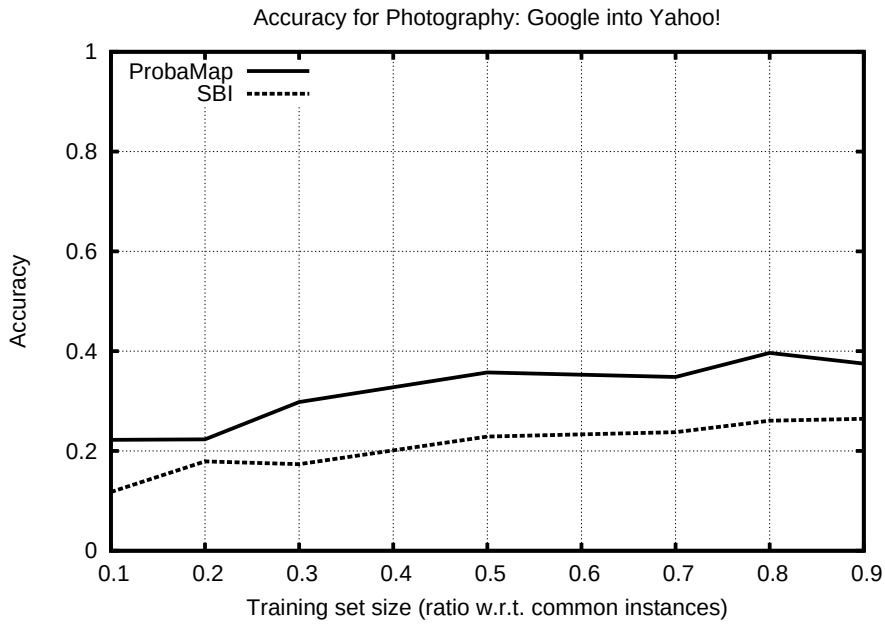


(b) `Software`: Google into Yahoo!

Figure VII.7: `Software`: Comparative accuracy for SBI and ProbaMap



(a) `Photography`: Yahoo! into Google



(b) `Photography`: Google into Yahoo!

Figure VII.8: `Photography`: Comparative accuracy for SBI and ProbaMap

From these results, we can conclude that ProbaMap outperforms SBI in average, with a .

SBI performs better for `Outdoors` and `Software` in the direction Google into Yahoo!, but it is distant of less than 5% in accuracy. Conversely ProbaMap significantly improves SBI results for `Autos` and `Photography` (about +10% better). In addition, ProbaMap is more robust than SBI against the quantity of instances used for discovering mappings and rules.

Now we shortly analyse these results with regard to he statistics for number of instances per class that are given in Table VII.3.

Subdirectory	Declared	Inferred	Classified	Classified + Inferred
Yahoo! - `Autos`	0.86	3.24	34.12	70.31
Google - `Autos`	0.84	3.18	23.93	65.25
Yahoo! - `Outdoors`	0.39	1.16	9.40	31.93
Google - `Outdoors`	0.66	1.96	15.83	53.80
Yahoo! - `Software`	1.70	6.00	107.41	402.62
Google - `Software`	0.23	0.81	14.48	54.29
Yahoo! - `Photography`	1.75	4.28	54.23	77.06
Google - `Photography`	0.91	2.22	28.21	40.99

Table VII.3: Number of instances by class

First, SBI and ProbaMap have better results when the number of instances per class is larger. The cases where ProbaMap does not outperform SBI correspond to the cases where the taxonomies has very few instances per class (less than 2 inferred): for `Outdoors` and `Software`. Therefore, we can deduce that a very low population of classes leads to bad results with ProbaMap.

It seems to contradict the robustness of ProbaMap against the decrease of instances amount in the x-axis of the accuracy figures. But in this robustness experiment, the decrease of instances number is equally distributed in each class, so the instances distribution remains roughly similar to the distribution in the initial directory. In contrast, `Outdoors` and `Google/Software` should have unbalanced distribution of instances in classes leading to a lot of (almost) empty classes. In fact, this situation is the worst case for ProbaMap because the Bayesian estimation of probabilities is based on ratios of the sizes of class extensions.

Impact of the classification

We have conducted some preliminary tests for determining the best classifier between SVM and C4.5 (as Naive Bayes has been discarded by the tests on synthetic data). We also wanted to determine the best preprocessing and parameters for it. The evaluation is based on cross-validation of classifiers. We found that SVM was the best and that the best preprocessing was a bag-of-words preprocessing on text abstract of instances pages that keep about 500 words, with a TF-IDF weighting and a stemmization step for words.

For the use of classification, we should underline the following points:

- There is a few instances added directly by the classification with regard to the number of total instances, but these instances are correctly classified according to the cross-validation

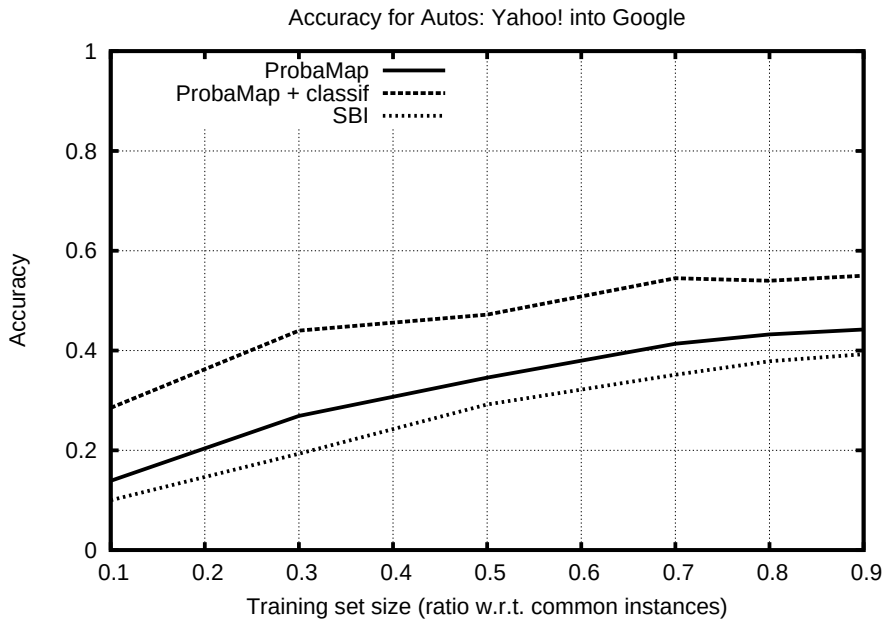
tests computed. Therefore, there is a good Precision on instances classification, but we do not know the Recall.

- As the number of total instances is very higher than the instances of classes, almost all probabilities P_c are close to 0 and P_i are close to 1. Then S_c is set to 0, and S_i is considering as very hard to set without doing overfitting or experiments on a large amount of different dataset, so it is set to 0 here. (Further experiments should be conducted to determine a good value for S_i.)
- Because of the very low values for all probabilities P_c, we adapted the post-processing phase allowing to obtain rules from mappings. We include the computation of probabilities for mappings in both directions, and for each source class A, we associate the target class B such that $min(P_c(A \sqsubseteq B), P_c(B \sqsubseteq A))$ is the highest among all possible target classes.

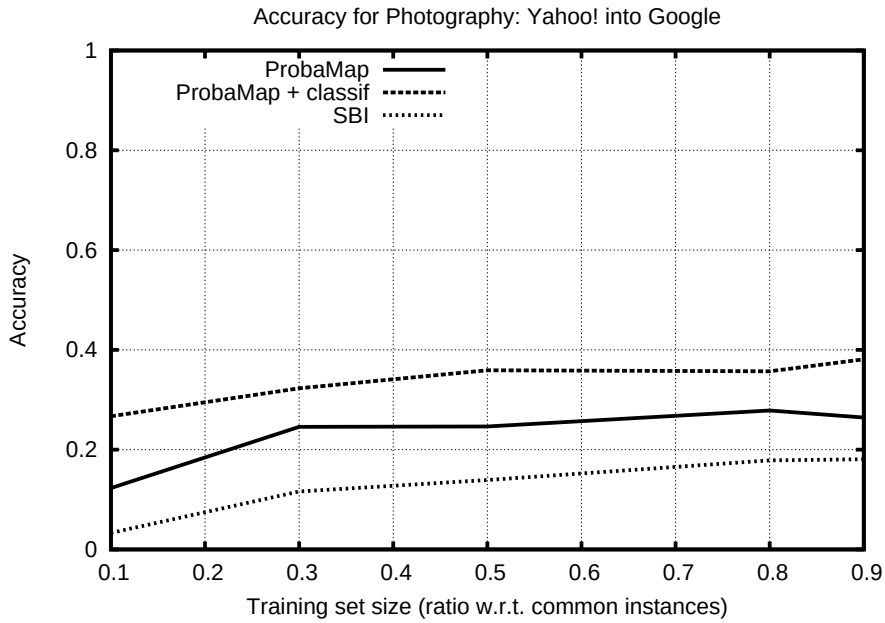
ProbaMap with classification outperforms both ProbaMap and SBI, with a better accuracy (at least 10%) in average for ProbaMap with classification. Note that when using classification, the average accuracy of ProbaMap with a limited training set size (50% of shared instances) is better in average than the accuracy of SBI with a training set containing 90% of shared instances. In particular, ProbaMap with classification significantly outperforms ProbaMap without Classification (about 10% better) and SBI (about 20% better) for the datasets `Autos` and `Photography`, whatever the size of the training set. Additionally we can see that ProbaMap with classification outperforms SBI on the `Software` directory up to 60% of instances for the training set, whereas ProbaMap without classification does not. For the `Outdoors` results pictured in Figure VII.10(a), ProbaMap with classification is better for a small training set ($\leq$ 20% of the shared instances). Thus, in the case where there are initially few instances by class and where ProbaMap without classification provides the lowest results, the classification allows to improve the results. But this improvement becomes slight if there are a small quantity of classified instances. For example, the dataset `Outdoors` is in this case, and the results for `Outdoors` performed by ProbaMap with classification might be explained by the very few amount of declared and inferred instances, that are not enough to train correctly the classifiers.

We have also check that all these results both for ProbaMap and ProbaMap with classififcation are rougly better in average than the results of a modified version of SBI using Naive Bayes classification so-called SBI-NB [IHT04]. More exactly, both versions of Probamap perform better for 6 of the 8 tested directories. (Note that we are provided results for SBI-NB with an old dataset collected in 2004, so the comparison is roughly done.)

The experiments conducted on collected Web directories show that the probabilistic mapping discovery method that we propose gives good results on real-world datasets, and can take advantage of classification techniques to compensate small training set sizes. This is an important quality for real world taxonomies built by different people, that are unlikely to have many instances in common. The worst case for ProbaMap is the lack of instances or the lack of descriptions for classifying them when classification is needed to merge instances between taxonomies.

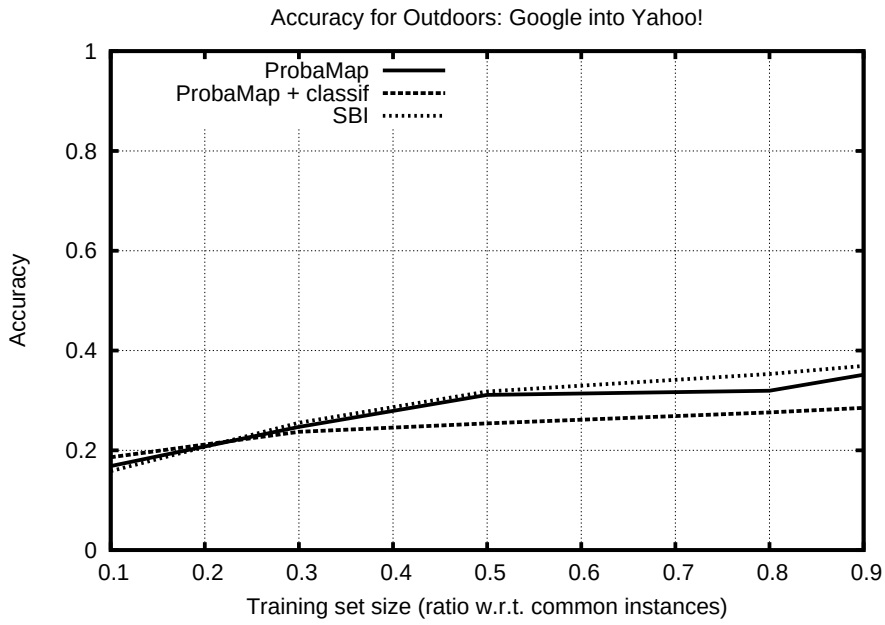


(a) `Autos`: Yahoo! into Google

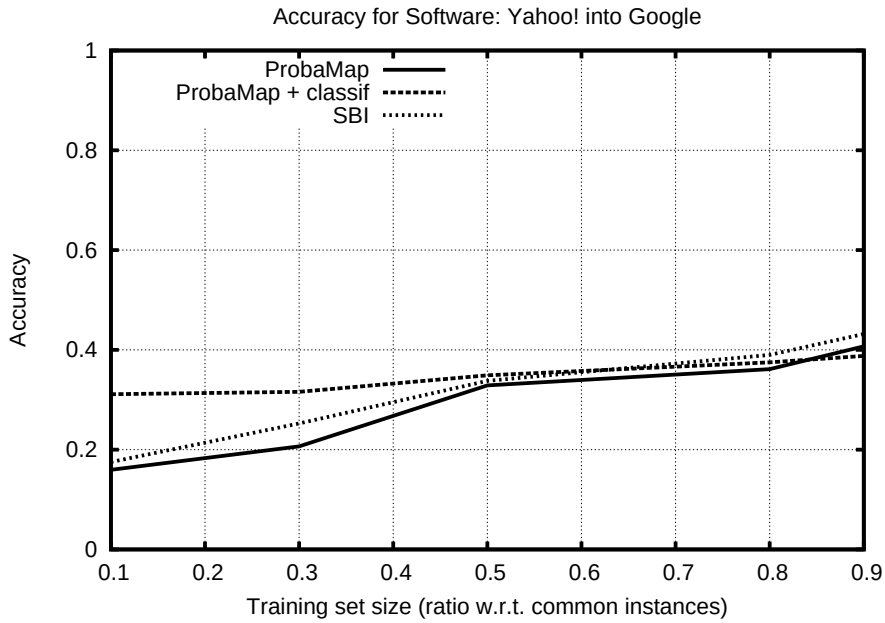


(b) `Photography`: Google into Yahoo!

Figure VII.9: Comparative accuracy results using classification (1)



(a) `Outdoors`: Google into Yahoo!



(b) `Software`: Google into Yahoo!

Figure VII.10: Comparative accuracy results using classification (2)

CHAPTER

EIGHT

CONCLUSION

The general problem addressed in this thesis is ontology matching. This problem turns out to be one of the main bottlenecks in the future Semantic Web, for example to enable collaborative exchange of data between multiple sources. In particular, we have focused on tackling the issue of handling uncertainty for *inclusion mappings* while defining a formal logical semantics for them.

In this perspective, we have presented two probabilistic functions that can be used and estimated to associate a confidence value to each discovered mapping of the form $A \sqsubseteq B$ where A and B are ontology classes:

- $P_i(A \sqsubseteq B) = P(\overline{A} \cup B)$
- $P_c(A \sqsubseteq B) = P(B|A)$

where P is a probability measure on a probability space of instances, and classes are subsets (i.e. events) in this space. By analyzing the properties of both probabilistic functions, we have shown that a combination of them leads to better quality and better algorithm time complexity.

We have proved a property of monotony for the probabilistic functions P_i and P_c with regard to the logical entailments: a mapping can not have a probability value lower than all its consequences. We have pointed out that this is true for P_i in any cases, and true for P_c only if the class on the left-hand side of the mappings remains the same.

We have even shown a more general result: under additional assumptions, all probabilistic functions that respect the strong property of monotony should be of the form $f(P_i(A \sqsubseteq B))$ with f a continous increasing function in $[0; 1]$ (see Theorem IV.2 page 50). Therefore P_i is the simplest function that respect this property. This theorem brings a grounded justification to the use of the probabilistic function P_i for measuring the probability of mappings.

For estimating probabilities values, we have presented a Bayesian method based on instances, i.e. extensions of classes and classes intersections. In the cases where the taxonomies are populated with disjoint pools of instances, we have adapted and tuned a common way for merging

instances of both taxonomies by exploiting classifiers trained with instances of taxonomies taken in isolation. Features used as data for classifiers are the preprocessed textual descriptions of the respective instances.

Then we have devised a generate-and-test algorithm called ProbaMap that discovers all possible inclusion mappings between two taxonomies for which P_i and P_c respectively exceed two given thresholds S_i and S_c. In order to be scalable, ProbaMap exploits the monotony property to prune its search space. It is important to note that the structure of the search space is not explicitly stored in memory, thus the pruning function is a sore point of the ProbaMap implementation.

Finally, we have conducted two kinds of experiments to validate our approach. The first phase was constituted by a thorough analysis of the behaviour of ProbaMap on controlled synthetic data. We have designed a full generator for synthetic taxonomies, mappings and instances. Different parameters allow to set the size of the taxonomies and then of the search space such that the size of the mappings to find out and the number of instances in each class. This generator has been used to conduct a thorough systematic analysis of the behaviour of ProbaMap by crossing input parameters with ProbaMap parameters. The results of this analysis confirmed that the combination of both probabilistic functions is better for quality and efficiency. They showed that ProbaMap is robust with regard to noise in data and that it requires a limited number of instances per class.

The second sequence of experiments has been conducted on real-world Web directories (Yahoo! and Google), and results are promising in term of scalability and quality. In particular, ProbaMap has been shown to outperform SBI [ITH03], a state-of-the-art algorithm for aligning and integrating taxonomies with instances, in term of quality.

We envision two main perspectives. The first one takes place in the setting of probabilistic reasoning by reusing probabilities associated with discovered mappings. The second one is intended to make the confidence values returned by standard matching method correspond to real probabilities. We give the detail of both perspectives below.

For the first perspective, we will study a setting for probabilistic query answering, in the spirit of probabilistic databases [DHY07]. In such a setting, probabilities associated to mappings are reused by the query answering algorithm to give some probability values for each answer. In particular, it should be interesting to focus on reasoning-based query algorithm like in the Somewhere [ACG+05] setting. There are existing works on introducing probabilities in logic and inference process. Probability logic [Ada98], or probabilistic descriptions logic like P-CLASSIC [KLP97] or P-$\mathcal{SHIQ}$ [CFL+08] define some probabilistic extensions of existing logics. For instance, in [Ada98], each formula of propositional logic F is associated with a probability value $P(F)$, and an additional kind of formula is added: the conditional $A|B$ where A and B are propositional formulas. Such a conditional has for associated probability value the conditional probability $P(A|B)$. In this theory, the uncertainty of a formula F denoted $u(F)$ is defined by $u(F) = 1 - P(F)$: it corresponds to the one's complement of the propability of the formula, which can be provided by the probabilistic functions we have defined. Then, an inference rule in such a language is said to be *p-valid* if the uncertainty of the conclusion is lower than the uncertainty sum of the premisses. A series of methods is given for deciding the p-validity of an inference rule, without computing the probabilities of each formula (that can vary anyway).

This work fits our proposed model in the way that the classical implication and the conditional

are both presents in the language. Works for probabilistic description logics generally introduces the conditionals formulas too. Based on inference rules and their properties about probabilities, such probabilistic and logical framework can be used to do probabilistic reasoning and to obtain probabilistic answers to queries involving probabilistic mappings.

Our second pespective tackles a central issue in ontology matching (see [SE08]), that is how to provide a formal probabilistic semantics to discovered mappings. Such a probabilistic semantics is necessary if one wants to reuse and interpret mappings correctly: for instance, probabilistic mappings are needed for probabilistic reasoning using mappings, in particular in a probabilistic query answering setting.

Most of the existing matching methods return a similarity value with each mapping. Such values can not directly be interpreted as probabilities, because they do not respect the monotony properties (see Proposition IV.1). These properties state that for two mappings m and m' such that m entails m' according to the matched taxonomies, the associated value for m' should be higher than the value for m, under some additional assumptions.

Thus, the key point of our second perspective is how to take advantage of the quality and efficiency of existing methods while guaranteeing such monotony properties in order to be able to interpret the returned alignment in a probabilistic framework. This perspective also consists in formulating and implementing a post-processing step to such matching methods in order to transform the returned coefficients into coefficients that can be interpreted as probabilities, i.e. that respect the above property of monotony. For this purpose, we plan to use the similarity flooding principle [MGR$^+$02] between mappings probabilities, in the spirit of N2R [SPR09] and OLA [EV04]. Coefficients for mappings are initialized by thoses returned by the existing method to postprocess. The properties of monotony are interpreted as influence between coefficients of mappings connected by an entailment relation. Theses influences between mappings coefficients are modeled by non-linear equations, involving particularly the max function.

Then the equation system may be solved by an iterative fix-point algorithm. A fix-point should be reached because of properties of the equation system, in particular its non-linearity. Therefore, this algorithm is expected to converge to a set of updated mappings coefficients that respect the properties of monotony, and that can be also considered as probabilities.

APPENDIX

A

PROOFS

A.1 Proofs for the characterization of monotonous confidence functions

A.1.1 Proof of the Lemma IV.α

Proof Proof of the Lemma IV.α:

For every four classes A, A', B, B', two taxonomies $\mathcal{T}$ and $\mathcal{T}'$ can be constructed for which:

- A', B' belong to $\mathcal{T}'$ and A, B belong to $\mathcal{T}$,
- $A' \sqsubseteq A$ is a specialization relation in $\mathcal{T}$ (possible because $A' \subseteq A$)
- $B \sqsubseteq B'$ is a specialization relation in $\mathcal{T}'$ (possible because $B \subseteq B'$)

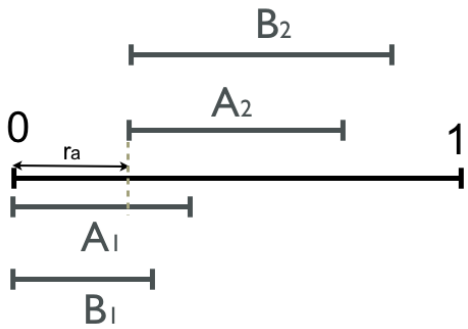
As $\mathcal{T}, \mathcal{T}', A \sqsubseteq B \models A' \sqsubseteq B'$ (by Proposition II.1 page 17), respecting the strong monotony property for this mapping means that $F_P^{\mathcal{T}_1,\mathcal{T}_2}(A, B) \leq F_P^{\mathcal{T}_1,\mathcal{T}_2}(A', B')$ should hold. As $F_P^{\mathcal{T}_1,\mathcal{T}_2}(A, B) = F_P(A, B)$ and $F_P^{\mathcal{T}_1,\mathcal{T}_2}(A', B') = F_P(A', B')$, $F_P(A, B) \leq F_P(A', B')$ should hold.

This is right for every classes A, A', B, B', then F_P should respect the property of the lemma.

□

A.1.2 Proof of the Theorem IV.2

We start by giving an intuitive interpretation of the assumption (ii) of the theorem, because it is invoked multiple times in the proof.



Considering the above figure and that two classes are mapped to subsets of the segment $[0;1]$. Let A_1 and A_2 be classes corresponding to the corresponding labelled subsets. The region which contradicts the mapping $A_1 \sqsubseteq A_2$ appears at the left of A_1 and measures $r_A = 1 - P_i(A_1 \sqsubseteq A_2)$. The assumption (ii) asserts in particular that for every class B_1 contained in A_1 and such that the region that contradicts $B_1 \sqsubseteq A_2$ is the same as the one for $A_1 \sqsubseteq A_2$,
$F_P(B_1, A_2)$ is equal to $F_P(A_1, A_2)$.

A similar equality $F_P(A_1, B_2) = F_P(A_1, A_2)$ is asserted with B_2 containing A_2 and the region that constradicts both mappings $A_1 \sqsubseteq A_2$ and $A_1 \sqsubseteq B_2$ are the same.

Actually, the assumption (ii) asserts that when P_i is constant between pairs of mappings for which either the left-hand side or the right-hand side are related by an inclusion, the confidence function F_P should keep constant.

We now give the proof of the theorem.

Proof :

Given four classes A_1, B_1, A_2, B_2 such that $P_i(A_1 \sqsubseteq A_2) \leq P_i(B_1 \sqsubseteq B_2)$, we will exhibit two classes C_1, C_2 such that:

- $F_P(B_1, B_2) = F_P(C_1, C_2)$
- $C_1 \subseteq A_1$
- $A_2 \subseteq C_2$

It will leed to $F_P(A_1, A_2) \leq F_P(C_1, C_2)$ by hypothesis, and then to

$$F_P(A_1, A_2) \leq F_P(B_1, B_2)$$

It will demonstrate the point (1) of the theorem.

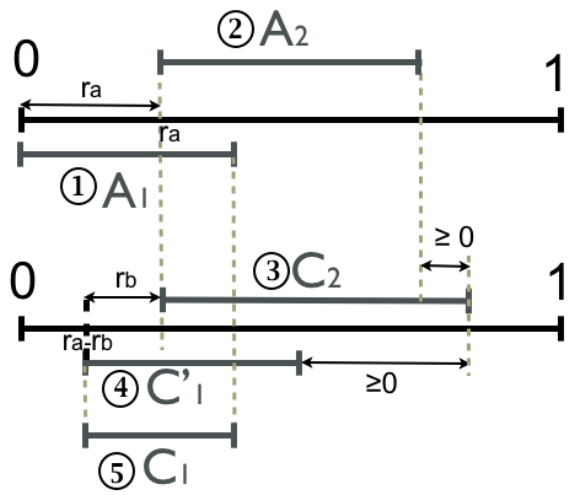
Let us introduce :

- $r_a = 1 - P_i(A_1 \sqsubseteq A_2) = P(A_1 \setminus A_2)$
- $r_b = 1 - P_i(B_1 \sqsubseteq B_2) = P(B_1 \setminus B_2)$

By hypothesis on P_i, $r_a \geq r_b$.

There always exist a bijection $B : \Omega \to [0;1]$ such that $B(A_1) = [0; P(A_1)]$ and $B(A_2) = [b; b + P(A_2)]$, as illustrated in the figure below.

- **First case:** $P(A_2) \leq P(B_2)$



We introduce :

- $C_2 = B^{-1}([r_a, r_a + P(B_2)])$ of probability $P(B_2)$, if $r_a + P(2) \leq 1$ or $C_2 = B^{-1}([r_a; 1] \cup [0; P(B_2) - 1 + r_a])$ else.
- $C'_1 = B^{-1}([r_a - r_b; r_a - r_b + P(B_1)])$ of probability $P(B_1)$.
  In the case where $r_a - r_b + P(B_1) > 1$, then $C'_1 = B^{-1}([r_a - r_b, 1]) \cup B^{-1}([0; P(B_2) - 1 + r_a - r_b])$. Note that all the elements that belong to $C'_1 \cap \overline{C_2}$ are contained in $B^{-1}(r_a - r_b; r_b)$ by construction. (remember that $r_b \leq r_a$ by hypothesis).

$F_P(B_1, B_2) = F_P(C'_1, C_2)$ because

$$F_P(B_1, B_2) = G(P(B_1), P(B_2), P(B_1 \cap B_2)) = G(P(C'_1), P(C_2), P(C'_1 \cap C_2)) = F_P(C'_1, C_2)$$

Let us introduce $C_1 = A_1 \cap C'_1$. As $C'_1 \setminus C_2 \subseteq A_1$ (there are in $B^{-1}(r_b - r_a, r_a)$),

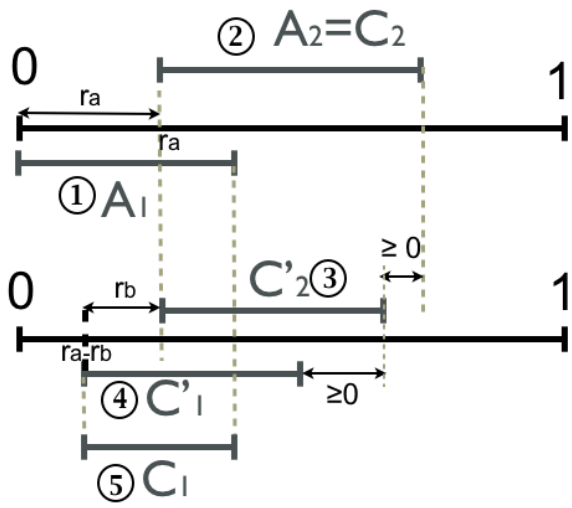
we have $P_i(C_1 \sqsubseteq C_2) = P_i(C'_1 \sqsubseteq C_2)$

and then, thanks to the assumption (ii): $F_P(C_1, C_2) = F_P(C'_1, C_2)$. So:

$F_P(C_1, C_2) = F_P(B_1, B_2)$, and by construction:

$C_1 \sqsubseteq A_1$ and $A_2 \sqsubseteq C_2$.

- **Second case:** $P(A_2) \geq P(B_2)$



We introduce :

- $C'_2 = B^{-1}([r_a, r_A + P(B_2)])$ of probability $P(B_2)$
- $C'_1 = B^{-1}([r_a - r_b; r_a - r_b + P(B_1)])$ of probability $P(B_1)$.

By construction, we have: $F_P(B_1, B_2) = G(P(B_1), P(B_2), P(B_1 \cap B_2)) = G(P(C'_1), P(C'_2), P(C'_1 \cap C'_2)) = F_P(C'_1, C'_2)$

- $F_P(A_1 \cap C'_1, A_2) = F_P(A_1 \cap C'_1, C'_2)$ thanks to the assumption (ii) and:
  - $C'_2 \subseteq A_2$ by construction (remember that $P(C'_2) \leq P(A_2)$)
  - $P_i(A_1 \cap C'_1 \sqsubseteq A_2) = P_i(A_1 \cap C'_1 \sqsubseteq C'_2)$:
    actually, elements that belongs to $A_1 \cap C'_1$ and not to C'_2 are only at the left of C'_2, i.e. those that are in $B^{-1}([r_b; r_a])$. This is due to the fact that the right bound of C'_1 is at the left of the right bound of C'_2, by construction. Then the right bound corresponding to $A_1 \cap C'_1$ does not exceed the right bound of C'_2.
- $F_P(A_1 \cap C'_1, C'_2) = F_P(C'_1, C'_2)$ because the application of (ii) again because $A_1 \cap C'_1 \subseteq A_1$ and P_i is the same for $A_1 \cap C'_1 \sqsubseteq C'_2$ and $C'_1 \sqsubseteq C'_2$.
  This is due to the fact that the elements that belong to C'_1 but not to C'_2 are all in A_1, by construction: they are in $B^{-1}(r_a - r_b; r_a)$ which is contained in A_1.

So we have, $F_P(A_1 \cap C'_1, A_2) = F_P(C'_1, C'_2)$,
and by denoting $C_1 = A_1 \cap C'_1$, and $C_2 = A_2$, it becomes:
$F_P(C_1, C_2) = F_P(B_1, B_2)$ and

$C_1 \sqsubseteq A_1$ and $A_2 \sqsubseteq C_2$.

The point (1) of the theorem is now proved.

Let us prove the point (2).

As $F_P(A, B) = G(P(A), P(B), P(A \cap B))$ and $P_i(A \sqsubseteq B) = 1 - P(A) + P(A \cap B)$, there exist G' such that $F_P(A, B) = G'(P_i(A \sqsubseteq B), P(A), P(B))$

Then G' is an increasing function of P_i, we have:

$$X \leq X' \Rightarrow 0 \leq G'(X, \overline{y}) - F_P(X', \overline{y'})$$

whatever $\overline{y}, \overline{y'}$ representing all other arguments.

As G is continuous, G' is also continuous with regard to its first argument when $X' \to X$, $G'(X', \overline{y'}) \to G'(X, \overline{y'})$. The inequality above becomes then:

$$O \leq G'(X, \overline{y}) - G'(X, \overline{y'})$$

By swapping $\overline{y}$ and $\overline{y'}$ in the previous reasoning, we obtain:

$$O \leq G'(X, \overline{y'}) - G'(X, \overline{y})$$

That leads to $G'(X, \overline{y}) = G'(X, \overline{y'})$ for all X and all pair $\overline{y}, \overline{y'}$.

Then G' is only dependent of its first argument and it leads to the form:

$$F_P(A \sqsubseteq B) = G'(P_i(A \sqsubseteq B))$$

where G' is an increasing function.

□

A.2 Proof of the correctness of the generator

A.2.1 Proof of the proposition VI.1

Proof :

By following the proof of Proposition IV.1 page 45, by replacing each P_i, P_c and probability by its estimation (we can do it because only used pure arithmetic is used), we obtain:

$$\widehat{P_c}^O(m) - \widehat{P_i}^O(m) = \frac{1 - \widehat{P(A)}^O}{\widehat{P(A)}^O}(\widehat{P(A \cap B)}^O - \widehat{P(A)}^O)$$

Given the fact that:

- $\widehat{P(A)}^O = \frac{1+\widehat{Ext}^O(A, \mathcal{O}_1 \cup \mathcal{O}_2)}{2+N}$
- $\widehat{P(A \cap B)}^O = \frac{1+\widehat{Ext}^O(A \cap B, \mathcal{O}_1 \cup \mathcal{O}_2)}{2+N}$
- $\widehat{Ext}^O(A \cap B, \mathcal{O}_1 \cup \mathcal{O}_2) \subseteq \widehat{Ext}^O(A, \mathcal{O}_1 \cup \mathcal{O}_2)$ as $A \cap B \subseteq A$,

we have:

$$\widehat{P(A \cap B)}^O - \widehat{P(A)}^O \leq 0$$

and then

$$\widehat{P_c}^O(m) \leq \widehat{P_i}^O(m)$$

□

A.2.2 Proof of the Theorem VI.1

Proof :

It is obvious that the required sizes n_1 and n_2 of the two taxonomies are respected. As they are directed graph that do not contain any cycle (there are forest of general trees) they fit the definition of taxonomies. Due to constraints during the mappings generation, $\mathcal{M}_S$ are the most specific mappings among all the mappings they entails according to the taxonomies.

In fact, the core of the correctness property is the proof that the generated data respect the properties (PO) and (NO) (page 84). Below are the estimation formulas used with oracle classification. Note that the two sets of instances are disjoints (distinct URIs), and denoted $\mathcal{O}_1$ and $\mathcal{O}_2$. Remember that the number of instances by class is denoted n_p.

- $\widehat{P_i}^O(A \sqsubseteq B) = 1 - \frac{|\widehat{Ext}^O(A,\mathcal{O}_1\cup\mathcal{O}_2)|}{2+(n_1+n_2)n_p} - \frac{|\widehat{Ext}^O(A\cap B,\mathcal{O}_1\cup\mathcal{O}_2)|}{4+(n_1+n_2)n_p}$
- $\widehat{P_c}^O(A \sqsubseteq B) = \frac{|\widehat{Ext}^O(A\cap B,\mathcal{O}_1\cup\mathcal{O}_2)|\times(2+(n_1+n_2)n_p)}{|\widehat{Ext}^O(A,\mathcal{O}_1\cup\mathcal{O}_2)|\times(4+(n_1+n_2)n_p)}$

with the following formula for the estimated extension of intersection $A \cap B$:

$$\widehat{Ext}^O(A \cap B, \mathcal{O}_1 \cup \mathcal{T}_2) = \widehat{Ext}^O(A, \mathcal{O}_1 \cup \mathcal{O}_2) \cap \widehat{Ext}^O(B, \mathcal{O}_1 \cup \mathcal{O}_2)$$

Now we show that there exist $\epsilon \in [0;1]$ and a function $n_p \longrightarrow f(n_p)$ with $\lim_{n_p\to\infty} f(n_p) = 1$ such that

- (PO) $\widehat{P_c}^O(A \sqsubseteq B) \geq 1 - \epsilon$ for $A \sqsubseteq B$ expected mappings or specialization,
- (NO)] $P(\widehat{P_i}^O(A \sqsubseteq B) \leq 1 - 2\epsilon) \geq f n_p$ for the unexpected possible mappings or specializations

Then, setting the two thresholds S_c and S_i to $1 - \frac{3}{2}\epsilon$ leads to consider valid all the expected implication between classes, and invalid all the others, except for a small proportion of them.

- For all implication formula $A \sqsubseteq B$ such that $\mathcal{T}_1, \mathcal{T}_2, \mathcal{M}_S \models A \sqsubseteq B$:
  $\widehat{P_c}^O(A \sqsubseteq B) = \frac{\widehat{Ext}^O(A\cap B,\mathcal{T}_1\cup\mathcal{T}_2)\times(2+(n_1+n_2)n_p)}{\widehat{Ext}^O(A,\mathcal{O}_1\cup\mathcal{O}_2)\times(4+(n_1+n_2)n_p)}$ and
  $\widehat{Ext}^O(A \cap B, \mathcal{O}_1 \cup \mathcal{O}_2) = \widehat{Ext}^O(A, \mathcal{O}_1 \cup \mathcal{O}_2)$ because $A \sqsubseteq B$ is expected, then:
  $\widehat{P_c}^O(A \sqsubseteq B) = \frac{2+(n_1+n_2)n_p}{4+(n_1+n_2)n_p}$

  By fixing $\epsilon = \frac{2}{4+(n_1+n_2)n_p}$, both estimations of P_c and P_i are higher than $1 - \epsilon$.

- For all $A \sqsubseteq B$ such that $\mathcal{T}_1, \mathcal{O}_2, \mathcal{M}_S \not\models A \sqsubseteq B$:
  $\widehat{P_i}^O(A \sqsubseteq B) = 1 - \frac{\widehat{Ext}^O(A, \mathcal{O}_1 \cup \mathcal{O}_2)}{2+(n_1+n_2)n_p} - \frac{\widehat{Ext}^O(A \cap B, \mathcal{O}_1 \cup \mathcal{O}_2)}{4+(n_1+n_2)n_p}$ then
  $\widehat{P_i}^O(A \sqsubseteq B) \leq 1 - \frac{|\widehat{Ext}^O(A, \mathcal{O}_1 \cup \mathcal{O}_2) \setminus \widehat{Ext}^O(B, \mathcal{O}_1 \cup \mathcal{O}_2)|}{4+(n_1+n_2)n_p}$

  $P(P_i^O(A \sqsubseteq B) < 1 - 2\epsilon) = P(\frac{|\widehat{Ext}^O(A, \mathcal{O}_1 \cup \mathcal{O}_2) \setminus \widehat{Ext}^O(B, \mathcal{O}_1 \cup \mathcal{O}_2)|}{4+(n_1+n_2)n_p} > 2\epsilon)$

  As there is no path between A and B in the generated DAG of classes , there is at least one attribute in $Int(B)$ which is not in $Int(A)$. Then, each generated instance of $inst(A)$ generated especially for A is also an instance of B with a probability of $\frac{1}{2}$. The random set of instances of A that are not in B is denoted $Gen_{A \setminus B}$ and its size follows a binomial law $\beta(\frac{1}{2}, n_p)$.

  As $Gen_{A \setminus B} \subseteq \widehat{Ext}^O(A, \mathcal{O}_1 \cup \mathcal{O}_2) \setminus \widehat{Ext}^O(B, \mathcal{O}_1 \cup \mathcal{O}_2)$,

  $P(P_i(A \sqsubseteq B) < 1 - 2\epsilon) \geq P(|Gen_{A \setminus B}| > 2\epsilon(4 + (n_1 + n_2)n_p))$
  and then,
  $P(P_i(A \sqsubseteq B) < 1 - 2\epsilon) \geq 1 - 2^{-n_p} \sum_{k=0}^{2\epsilon(4+(n_1+n_2)n_p} C_{n_p}^k$
  By taking $\epsilon = \frac{2}{4+(n_1+n_2)n_p}$ as done for the (PO) property, it conducts to
  $P(P_i(A \sqsubseteq B) < 1 - 2\epsilon) \geq 1 - 2^{-n_p} \sum_{k=0}^{4} C_{n_p}^k$
  Thus, $f(n_p) = 1 - 2^{-n_p} \sum_{k=0}^{4} C_{n_p}^k$ is a relevant lower bound:
  As $\sum_{k=0}^{n_p} C_{n_p}^k = 2^{n_p}$, $f(n_p)$ is of the form

  $$1 - \frac{a}{a + \sum_{k=5}^{n_p} C_{n_p}^k} \longrightarrow 1 \text{ when } n_p \to +\infty$$

  The constraint (NO) is respected too.

□

APPENDIX

B

RÉSUMÉ LONG EN FRANÇAIS

B.1 Introduction

La quantité de données échangées à travers ce réseau a aussi considérablement augmenté, ce qui pose de nombreux défis. Le Web est constitué de plusieurs dizaines de milliard de documents, le plus souvent écrits en HTML, et sans structure commune. La plupart des requêtes au Web se font par l'intermédiaire de moteurs de recherches (Google, Yahoo!, Bing) qui indexent chacun un sous-ensemble du Web par mot-clefs.

La quantité d'informations disponible contraste fortement avec les moyens d'y accéder et de les traiter : le pouvoir expressif des requêtes par mot-clefs est limité et le traitement des requêtes est initialement purement syntaxique. En réaction à ce problème, l'émergence du Web Sémantique (parfois appelé Web 3.0) a pour objectif un accès et un traitement sémantique de l'information disponible à travers le Web.

Web sémantique, ontologies, et nécessité de découvrir des correspondances

Dans le Web Sémantique, les documents sont tous exprimés dans un langage logique (RDF, RDFS, OWL) associé à une sémantique formelle. Ils possèdent donc une structure qui leur donne un sens. Les entités de base pour décrire des informations au sein du Web Sémantique sont des triplets RDF du type (entité sujet, relation, entité objet). Les composants de haut niveau sont eux appelés *ontologies*. Parmi les exemples d'ontologies, les ontologies de domaines sont utilisées pour décrire un domaine particulier : une ontologies des maladies humaines, ou bien la classification phylogénétique des êtres vivants. A un autre bout du spectre, les folksonomies sont de petites ontologies créées d'un point de vue utilisateur par des individus qui désirent organiser leurs documents (leur musique par exemple). Les *taxonomies* sont les composants essentiels des ontologies : ils représentent leur squelette par une hiérarchie de classes (ou concepts). Le Web Sémantique n'est encore qu'une petite part du Web, mais sa croissance est très rapide. En mai

2010, on estime sa taille à plus de 13 milliards de triplets RDF [1].

A l'échelle du Web (sémantique), les organisations ou utilisateurs fournissant des données sont susceptibles de décrire leurs propres ontologies pour organiser et annoter leurs documents. Dans cette configuration, découvrir des correspondances entre ontologies propres est une nécessité pour échanger des documents ou des informations entre acteurs du Web. Plusieurs facteurs sont à l'origine de l'hététérogénité entre différentes ontologies : des différences terminologiques entre les noms des entités (synonymes par ex.), des différences sémiotiques propres à l'interprétation des entités par les utilisateurs, des différences de couverture lorsque les domaines décrits ne coincident pas, des différences de granularité lorsque les niveaux de détails retenus pour la description ne sont pas les mêmes, et enfin des différences de perspective, lorsque les points de vues et les aspects décrits ne sont pas les mêmes (par exemple différence entre une carte topographique et une carte politique de la France).
Afin de permettre une communication automatisée entre différentes ontologies, il est nécessaire d'établir des correspondances sémantiques entre ontologies. Dans cette perspective, le projet Somewhere [ACG+05] a servi de motivation à cette thèse. Il décrit une configuration organisée en réseau pair-à-pair dans laquelle chaque pair annote ses documents à l'aide d'une taxonomie (et donc d'un vocabulaire de classes) qui lui est propre. Le but d'une telle configuration est de permettre de répondre à des requêtes de la part de chaque pair grâce à un algorithme de raisonnement distribué . Les requêtes d'un pair sont effectuées dans son vocabulaire. Il faut donc fournir au système des correspondances sémantiques entre certains pairs pour permettre la propagation des requêtes en vue d'obtenir des réponses.

A grande échelle, l'établissement de correspondances entre ontologies est un problème impossible à résoudre manuellement, à cause du nombre et de la taille des ontologies existantes. C'est pourquoi d'importants travaux sont menés depuis une vintaine d'année pour l'automatisation de cette tâche, et forment le domaine de l'*ontology matching* [ES07] ou alignement d'ontologies en Français. La plupart combinent des techniques utilisant des similarités sur les noms, les structures ou les documents décrits. Depuis 2004, une compétition internationale appelée OAEI[2] [EFH+09] est organisée tous les ans dans ce domaine. D'importants progrès ont été réalisés en terme de qualité des correspondances découvertes automatiquement. Cependant, aucun net gagnant ne ressort de ces concours. Il manque encore une solution intégrée suffisamment robuste et utilisable par des non-experts.

Problèmes traités dans cette thèse

Premièrement, cette thèse s'intéresse exclusivement à la découverte automatique de correspondances entre taxonomies, qui représentent la partie la plus importante des ontologies. Nous prétendons que les *correspondances d'inclusions* sont d'une importance majeure pour l'échange collaboratif de documents, notamment pour la reformulation de requêtes. En outre, elles ont plus de chance d'exister entre deux taxonomies que des correspondances d'équivalence. Jusqu'à présent, peu de méthodes[BSZ03, HSNR09, SVV08, JK07] se sont concentrées sur les correspondances d'inclusions.

Deuxièmement, l'incertitude est intrinsèque à la découverte de mappings. La plupart des méth-

[1] `http://esw.w3.org/SweoIG/TaskForces/CommunityProjects/LinkingOpenData`
[2] `http://oaei.ontologymatching.org`

odes d'alignement associent un coefficient de confiance à chaque correspondance découverte. Ces approches considèrent souvent les correspondances de façon isolée les unes par rapport aux autres, et ne tiennent pas comptent des possible relations d'implications logiques entre correspondances induites par les taxonomies. Ce manque de cohérence interdit de considérer les coefficients de confiance comme de vraies probabilités. A l'inverse, nous considérerons des *mappings d'inclusions avec une sémantique probabiliste.*

Enfin, le dernier point traité consiste à tenir compte du passage à l'échelle pour la découverte de mappings, qui représente un réel défi pour l'avènement du Web Sémantique, étant donné sa taille et son expansion.

Contributions

Dans cette thèse, nous proposons une approche pour découvrir automatiquement des correspondances d'inclusion probabilistes entre taxonomies, que nous appelerons mappings. Tout d'abord, nous analysons et comparons deux moyens de modéliser des mappings probabilistes compatibles avec les contraintes logiques déclarées dans chaque taxonomie à aligner. Dans ces deux modèles, la probabilité d'un mapping entre deux classes est basée sur la probabilité jointe des deux classes. Un résultat additionnel permettant de caractériser les modèles de mappings probabilistes compatibles avec la logique est donné. Nous fournissons également un moyen d'estimer les probabilités de mappings à partir des statistiques sur les instances déclarées. Ensuite, nous proposons un algorithme appelé ProbaMap, de type "génerer et tester" ainsi que son implémentation. Fondé sur les deux modèles de mappings probabilistes, ProbaMap découvre automatiquement les mappings dont la probabilité est supérieure à un certain seuil. Une propriété de monotonie des probabilités utilisées est à la base de l'optimisation de ProbaMap, évitant d'inspecter tous les mappings possible entre deux taxonomies.
Dans le but de réaliser des expérimentations fouillées et contrôlées, nous avons conçu un générateur complet de taxonomies, d'instances et de mappings. Les résultats expérimentaux en terme de qualité et de temps de calcul sont analysés en croisant les paramètres propres à ProbaMap et ceux contrôlant les données en entrée.
Nous présentons aussi deux séries d'expérimentations effectuées sur des données réelles : le benchmark OAEI directory et les web directories Yahoo! et Google collectés. Sur ces derniers nous avons mené une expérimentation comparée avec SBI [ITH03], une autre méthode de l'état de l'art.

Enfin, nous esquissons deux perspectives : la première consiste à réutiliser les mappings probabilistes pour répondre à des requêtes de façon probabilistes, par du raisonnement probabiliste. La deuxième introduit un moyen de post-traiter les coefficients de confiance retournés par la plupart des méthodes d'alignement pour les transformer en coefficients interprétables comme de vraies probabilités.

Plan

Cette thèse est organisée de la fçon suivante :

- Le **Chapitre II** (section B.2 de ce résum'e) présente les prérequis nécessaires et la position du problème;

- Le **Chapitre III**(section B.3 de ce résum'e) présente l'état de l'art dans le domaine de l'alignement; d'ontologies

- La **Partie I** fournit deux modèles de *mappings probabilistes* et un algorithme pour découvrir les plus probables entre deux taxonomies
  - Le **Chapitre IV** (section B.4 de ce résum'e) présente et compare en profondeur les deux modèles des mappings probabilistes que nous introduisons dans ce travail, ainsi qu'un moyen d'estimer les probabilités associées en utilisant une estimation bayésienne et optionnellement des classifieurs
  - Le **Chapitre V** (section B.5 de ce résum'e) introduit l'algorithme de découverte automatique de mappings ProbaMap et son implémentation.

- La **Partie II** est consacrée aux expérimentations sur ProbaMap
  - Le **Chapitre VI** (section B.6 de ce résum'e) présente les expérimentations conduites sur les données générées ainsi que le principe du générateur utilisé
  - Le **ChapitreVII** (section B.7 de ce résum'e) présente les expérimentations conduites sur des données réelles.

- La conclusion et les perspectives figurent au **Chapitre VIII** (section B.8 de ce résum'e).

B.2 Préliminaires et position du problème

Des prérequis classiques de théorie des probabilités sont donnés dans la thèse à propos des notions suivantes : espace probabilisé, variable aléatoire réelle, espérance et variance.

Taxonomies et correspondances: définitions et sémantique

Définition B.1 (Taxonomie):
Etant donné un vocabulaire $\mathcal{V}$ représentant un ensemble de classes, une taxonomie $\mathcal{T}_\mathcal{V}$ est un graphe orienté sans cycles (DAG) dans lequel chaque noeud est étiqueté par un nom d'une classe distincte dans $\mathcal{V}$, et chaque arc entre deux noeuds étiquetés respectivement C et D représentent une spécialisation entre les classes C et D.

*Chaque classe d'une taxonomie peut être associée avec un ensemble d'*instances *possédant un* identifiant *et une description exprimée dans un langage attribut-valeur.*

Pour éviter toute ambiguïté et sans perte de généralité, nous considérons que chaque taxonomie a son vocabulaire propre, par convention nous indiçons le nom des classes par l'indice de la taxonomie à laquelle elles appartiennent.

En plus de la notation graphique, il existe plusieurs notations textuelles pour exprimer la relation de spécialisation entre classes et l'appartenance d'une instance à une classe, respectivement : $C \sqsubseteq D$ et $C(i)$ en logique de description.

Ces notations possèdent une sémantique logique standard fondée sur l'interprétation de classes comme des ensembles d'un domaine. Une interprétation $\mathcal{I}$ consiste en un domaine d'interpretation $\Delta^{\mathcal{I}}$ et en une fonction $.^{\mathcal{I}}$ qui interprète chaque classe comme un sous-ensemble non-vide de $\Delta^{\mathcal{I}}$, et chaque instance comme élément de $\Delta^{\mathcal{I}}$. Selon l'hypothèse du nom unique, deux identifiants d'instances différents ont une interprétation différente.

Définition B.2 (Modèle d'une taxonomie):
$\mathcal{I}$ est un modèle pour la taxonomie $\mathcal{T}$ si:

- *pour chaque spécialisation $E \sqsubseteq F$ de $\mathcal{T}$: $E^{\mathcal{I}} \subseteq F^{\mathcal{I}}$,*
- *pour chaque déclaration d'appartenance $C(a)$ de $\mathcal{T}$: $a^{\mathcal{I}} \in C^{\mathcal{I}}$.*

Une inclusion $G \sqsubseteq H$ est *inferée* par une taxonomie $\mathcal{T}$ (noté par $\mathcal{T} \models G \sqsubseteq H$) si et seulement si dans chaque modèle $\mathcal{I}$ de $\mathcal{T}$, $G^{\mathcal{I}} \subseteq H^{\mathcal{I}}$. Une appartenance $C(e)$ est *inferrée* par une taxonomie $\mathcal{T}$ (noté par $\mathcal{T} \models C(e)$) si et seulement si dans chaque modèle $\mathcal{I}$ de $\mathcal{T}$, $e^{\mathcal{I}} \in C^{\mathcal{I}}$.

Définition B.3 (Extension d'une classe):
*Etant donné un ensemble d'instance $\mathcal{D}$ associé à une taxonomie $\mathcal{T}$. L'*extension *de la classe C dans $\mathcal{T}$, notée $Ext(C, \mathcal{T})$, est : $Ext(C, \mathcal{T}) = \{d \in \mathcal{D} / \ \mathcal{T} \models C(d)\}$*

Les mappings que l'on considère sont des inclusions entre classes de taxonomies différentes :

Définition B.4 (Mapping):
Etant donné deux taxonomies T_1 and T_2, un mapping entre T_1 and T_2 est un quintuplet (id, C_1, C_2, r, n) tel que:

- *C_1, C_2 sont deux classes respectives de T_1 et T_2*
- *$r \in \{\sqsubseteq, \sqsupseteq\}$ est la relation d'inclusion entre C_1 et C_2*
- *$n \in [0, 1]$ représente le coefficient de confiance associé au mapping*

Un mapping entre C_1 and C_2 est noté $C_1 \sqsubseteq C_2$ ou $C_2 \sqsubseteq C_1$, en fonction de sa direction.

Un alignement (de mappings) entre deux taxonomies est un ensemble de mappings entre ces deux taxonomies. L'implication logique entre classes s'étend à l'implication entre mappings comme défini ci-dessous:

Définition B.5 (Implication logique entre mappings):
Soient $\mathcal{T}_i$ et $\mathcal{T}_j$ deux taxonomies. Soient m et m' deux mappings entre $\mathcal{T}_i$ et $\mathcal{T}_j$: m implique m' (noté $m \preceq m'$) ssi chaque modèle de $\mathcal{T}_i$, $\mathcal{T}_j$ et m est un modèle de m', ce qui peut s'écrire formellement: $\mathcal{T}_i, \mathcal{T}_j, m \models m'$.

$\preceq$ est une relation d'ordre (partiel) sur l'ensemble $\mathcal{M}(\mathcal{T}_i, \mathcal{T}_j)$ des mappings possibles entre les deux taxonomies $\mathcal{T}_i$ and $\mathcal{T}_j$. Si $m \preceq m'$, on dit que m est plus spécifique que m' et que m' est plus général que m. La proposition suivante caractérise l'implication logique entre mappings en fonction de l'implication logique entre les classes impliquées au sein de leurs taxonomies respectives:

Proposition B.1:
Soit $m = E_1 \sqsubseteq F_2$ et m' deux mappings entre $\mathcal{T}_1$ et $\mathcal{T}_2$:
$m \preceq m'$ (i. e. $\mathcal{T}_1, \mathcal{T}_2, m \models m'$) ssi:

(1) m' est de la forme $E'_1 \sqsubseteq F'_2$, et

(2) $\mathcal{T}_1 \models E'_1 \sqsubseteq E_1$ et $\mathcal{T}_2 \models F_2 \sqsubseteq F'_2$

Probabilités et estimation bayésienne

Dans la thèse il est procédé à plusieurs rappels classiques de théorie des probabilités : tribu, évènement, espace mesurable, variable aléatoire réelle, espérance, variance, distributions ([Gut05]). Nous noterons $(\Omega, \mathcal{F}, P)$ l'espace probabiliste où l'univers est Ω, l'ensemble des évènements (tribu) $\mathcal{F}$ et P la mesure de probabilité définies sur ces évènements.

Estimation bayésienne
Un estimateur bayésien [Deg04] fournit une valeur estimée pour un scalaire θ fondé à la fois sur un ensemble d'observations et sur une distribution *a priori* suivie par θ. L'estimateur bayésien de θ est une fonction $\hat{\theta}$ qui minimise un certain risque (usuellement les moindres carrés) pour une séries d'observation données $\mathbf{x}$. Usuellement l'estimateur bayesien est donné par l'expression

$$\hat{\theta}(\mathbf{x}) = E[\theta|\mathbf{X} = \mathbf{x}] \quad (\mathbf{BE})$$

Cette espérance fait intervenir la distribution *a priori* de θ. L'estimation bayésienne est souvent appliquée à un paramètre inconnu d'une distribution d en le modélisant par une variable aléatoire suivant une distribution *a priori* fixée. Lorsqu'on peut choisir cette distribution *a priori*, prendre la distribution conjuguée de d permet de simplifier grandement les calculs pour la formule (BE), car les distributions de θ et de $\theta|X = x$ appartiennent alors à la même famille algébrique de fonctions. Par exemple, dans cette thèse, nous exploitons le fait que la distribution conjuguée de celle de Bernouilli est une distribution Beta.

Position du problème

Comme mentionné dans l'introduction, l'incertitude est intrinsèque à la découverte de mappings. Nous prônons le fait que les mappings doivent être modélisés en utilisant la théorie des probabilités, afin de les associer à des valeurs de probabilités mesurant leur degré de confiance. Proposer de tels modèles revient à définir comment les mappings et les classes impliquées sont mesurés en temps qu'événements ou probabilités conditionnelles au sein d'un espace mesurable.

Un tel modèle probabiliste ne peut pas être indépendant de la sémantique logique. En particulier, un mapping en impliquant un autre doit avoir une probabilité supérieure à celui-ci.

En conséquence, le premier problème est de proposer des modèles de mappings probabilistes. On doit également fournir un moyen pratique de calculer les probabilités associées à chaque mapping.

Le second problème dans cette thèse est de concevoir un algorithme de découverte des mappings les plus probables entre deux taxonomies, de telle sorte que cet algorithme passe à l'échelle.

B.3 Etat de l'art

Comme le souligne l'introduction, les mappings sont primordiaux pour les systèmes d'intégration de données et l'avènement du Web Sémantique. Un grand nombre de méthodes d'alignement d'ontologies ou de schémas on été développées au sein des communautés des bases de données et du Web Sémantique. La plupart des méthodes existantes sont fondées sur des combinaisons :

1. de similarités textuelles entre labels des entités des ontologies (par exemple TaxoMap [HSNR09], H-MATCH [CFM03]), à l'aide de string-distances par exemple. D'autres techniques utilisent des ressources linguistiques (comme WordNet [C.98]) pour calculer une similarité basée sur le sens des mots.

2. de similarités structurelles entre graphes représentant les taxonomies, issus du principe de "similarity flooding" [MGR+02], qui permet d'affecter une similarité à chaque paire d'entités de deux ontologies considérés comme des DAGs. L'idée sous-jacente consiste à dire que deux éléments de deux ontologies sont similaires lorsque leurs éléments adjacents sont similaires. Cette idée est modélisée par un système d'équations où les similarités entre paires s'influencent les unes les autres, en fonction des structures des ontologies. Les similarités sont calculées par un algorithme de point fixe. Le travail présenté dans OLA [EV04] illustre cette approche.

3. de raisonnement logique : Ctx-Match [SBMZ03] est représentatif de cette technique. Cette technique réduit le problème d'alignement à un problème SAT. Chaque classe est encodée dans une formule logique dans laquelle les atomes sont des synsets WordNet correspondant aux labels de la classe et de tous ses ancêtres, filtrés pour représenter uniquement les sens cohérents en fonction du contexte. Ainsi, on peut effectuer un raisonnement à partir de chaque classe en utilisant les connaissances de fond pertinente (par exemple WordNet), pour inférer des inclusions, des équivalences ou des disjonctions entre classes.

4. de similarités à base d'instances : elles sont fondées sur les statistiques ou les distributions des extensions des classes, et peuvent aussi utiliser les descriptions des instances si disponibles. Ces techniques permettent de calculer des similarités entre classes à partir de leurs instances. Elles peuvent exploiter des classifieurs, par exemple pour apprendre des classes à partir de leurs instances et étendre leurs extensions. Notre travail se situe au sein de cette catégorie de méthodes, dites "extensionnelles" (GLUE [DMDH02], Implication intensity [DGGB06], [ITH03]).

   En particulier, dans la méthode GLUE qui découvre automatiquement des correspondances 1-1 entre classes, les classes sont modélisées à partir de leurs extensions, dans un univers fini d'instances. Une étape de classification permet d'associer à chaque classe une extension

dans l'ensemble des instances des deux taxonomies. Pour chaque couple de classe (C_1, C_2), une similarité (par exemple Jaccard) est calculée à partir des distributions de C_1, C_2 et $C_1 \cap C_2$. Enfin une phase de relaxation initialisée par les similarités calculées, et qui tient compte de contraintes de structure et de domaines fournies par les utilisateurs, permet d'associer à chaque classe de la première taxonomie une et une seule classe de la seconde.

D'autres approches utilisant des techniques d'apprentissage artificiel ont été étudiées. Certaines utilisent un corpus d'alignements (e.g., [MBDH05]) ou d'instances (LSD [DDL00], SemInt [LC00], GLUE [DMDH02], FCA-merge [SM01]). En particulier, le travail introduit dans [WES08] utilise des classifieurs directement dans l'espace des mappings, en représentant ces derniers dans un espace vectoriel construit à partir des propriétés des instances associées aux mappings.

En fait, la plupart des méthodes d'alignement existantes combinent ces approches élémentaires de différentes façons (par exemple, COMA++ [ADMR05] et COMA [DR02], Cupid [MBR01], H-MATCH [CFM03], Lily [WX09], S-Match [GSY04], Clio [CHKP07]). Les méthodes d'alignement peuvent être catégorisées selon plusieurs critères : leur technique élémentaire dominante (ci-dessus), l'utilisation ou non de ressources externes, semi-automatique ou automatique, ré-utilisation d'alignements antérieurs, usage d'apprentissage artificiel, langage de mappings découverts et langage d'ontologie accepté en entrée...

Les méthodes d'alignement associent couramment des coefficients à chaque mapping découvert, indiquant le degré de confiance qu'elles leur associent. L'incertitude est intrinsèque à la découverte de mappings, en raison du fait que deux classes ou entités créées indépendamment sont peu susceptibles de se correspondre parfaitement. Comme indiqué dans [SE08], un défi important est de mieux comprendre les fondations permettant de modéliser l'incertitude, notamment pour améliorer la qualité d'interprétation des mappings.

Cependant, les coefficients associés aux mappings n'ont pas de signification probabiliste et sont souvent utilisés pour du ranking. Au contraire, notre approche promeut une sémantique probabiliste pour les mappings et fournit une méthode pour calculer ces probabilités à partir des extensions des classes dans les taxonomies à aligner. Calculer des probabilités et non pas des coefficients de similarités à partir des extensions constitue la différence majeure avec GLUE.

Le travail présenté ici se distingue des autres approches en essayant d'établir un pont entre probabilité et logique, en fournissant des modèles probabilistes de mappings qui sont consistants avec la sémantique logique des taxonomies. Ainsi, notre approche généralise des travaux existants fondés sur la représentation algébrique ou logique de mappings (Ctx-Match [SBMZ03], Clio [CHKP07]). Le travail présenté dans [GATM05] fournit un modèle formel pour les réconciliations sémantiques à base d'ensemble flous et analyse théoriquement les facteurs qui influencent l'efficacité des algorithmes d'alignement. Ce travail se fonde sur la logique floue, contrairement au nôtre qui se fonde sur les probabilités.

Notre travail peut s'inscrire de façon théorique dans le cadre général établit dans [DHY07], qui permet pour gérer l'incertitude en intégration de données.

De façon plus générale, notre approche est complémentaire de travaux récents sur les bases de données probabilistes [BSHW06, DS05].

B.4 Modèles probabilistes de mappings

La découverte de mappings rend ces derniers incertains de façon intrinsèques, à cause des multiples facteurs d'hétérogénéité entre taxonomies différentes vus en introduction. En même temps, il est essentiel de conserver un lien avec l'implication logique, surtout lorsque les mappings sont utilisés pour du raisonnement ou de la réécriture de requêtes. C'est pourquoi nous proposons des modèles de mappings probabilistes dont la sémantique est cohérente avec l'implication logique entre mappings. Les théories les plus connues qui introduisent de l'incertitude en logique sont la logique probabiliste et la logique floue. Cette dernière permet de représenter de l'information vague et imprécise, en associant un degré de vérité à chaque formule, alors que la logique probabiliste associe la valeur binaire VRAI ou FAUX à une formule, et fournit un degré de probabilité que cette formule ait cette valeur. Les deux modèles que nous proposons sont issus de la logique probabiliste, car elle étend la logique standard de façon plus naturelle et conserve deux valeurs de vérité.

Etant donnés deux taxonomies $\mathcal{T}_1, \mathcal{T}_2$, nous considérons un univers probabiliste $\mathcal{D}$ constitué de l'ensemble des instances possibles entre $\mathcal{T}_1$ et $\mathcal{T}_2$. Les évènements de cet univers sont tous les sous-ensembles de $\mathcal{D}$, dont la probabilité est mesuré par une mesure de probabilité P fixée. La classe E est associée à l'évènement aussi noté E "une instance tirée au hasard appartient à E", qui suit une loi de Bernouilli de paramètre égal à $P(E)$. Les formules logiques et les évènements sont en correspondance : la classe $\neg E$ correspond à l'ensemble complémentaire de E dans $\mathcal{D}$, les disjonction et intersections de classes correspondent aux disjonctions et intersections d'évènements. Dans ce travail, nous adoptons une interprétation ensembliste des classes, qui est simultanément cohérente avec les théories logiques et probabilistes.

Modèles probabilistes des mappings

Le premier modèle définit la probabilité d'un mapping $A_i \sqsubseteq B_j$ comme la probabilité conditionnelle de B_j sachant A_i:

Définition B.6 (Probabilité conditionnelle d'un mapping):
Etant donné un mapping $m = A_i \sqsubseteq B_j$, la probabilité conditionnelle *de m est définie par :*

$$P_c(m) = P(B_j|A_i) = \frac{P(A_i \cap B_j)}{P(A_i)}$$

Le second modèle vient directement de l'implication logique entre A_i et B_j équivalent à "non A_i ou B_j".

Définition B.7 (Probabilité implicative d'un mapping):
Etant donné un mapping $m = A_i \sqsubseteq B_j$, la probabilité implicative *de m est définie par :*

$$P_i(m) = P(\overline{A_i} \cup B_j) = 1 - P(A_i \cap \overline{B_j})$$

P_i and P_c sont appelées *fonctions de confiance probabilistes* par la suite.

Etude comparative des deux modèles probabilistes

Afin de comparer les deux modèles, on énumère d'abord leurs propriétés :

Proposition B.2 (Propriétés des fonctions de confiance probabilistes):
Etant donné un mapping m entre deux taxonomies $\mathcal{T}_i$ et $\mathcal{T}_j$. Les propriétés suivantes sont valables :

1. $P_i(m) \geq P_c(m)$.
2. *m est un* mapping certain *(i.e., $\mathcal{T}_i \mathcal{T}_j \models m$, m est déclaré):* $\Leftrightarrow P_c(m) \Leftrightarrow P_i(m) = 1$
3. $P_i(m) = 1 + P(lhs(m) \cap rhs(m)) - P(lhs(m))$
4. $P(lhs(m)) = 0$ *or* $P(rhs(m)) = 1 \;\Rightarrow\; P_i(m) = 1$

Les deux fonctions de probabilités diffèrent en regard de la propriété de monotonie par rapport à l'implication logique (cf. Definition B.5).

Théorème B.1 (Propriété de monotonie):
Soient m et m' deux mappings.

1. *Si $m \preceq m'$ alors $P_i(m) \leq P_i(m')$ (monotonie forte)*
2. *Si $m \preceq m'$ et $lhs(m) = lhs(m')$ (mêmes classes de gauche) alors $P_c(m) \leq P_c(m')$ (motonie faible)*

Pour étudier de façon statistique P_c et P_i, on les considère comme des variables aléatoires fonctions de paires de classes aléatoires formant des mappings, avec comme hypothèse que chaque instance de l'univers a une probabilité $\frac{1}{2}$ d'appartenir à une classe aléatoire. On constate alors que l'espérance de P_c est de $\frac{1}{2}$ et que celle de P_i est de $\frac{3}{4}$.

Nous comparons maintenant les propriétés et montrons que P_i et P_c peuvent être utilisées en combinaison pour découvrir des mappings. L'objectif principal étant de déterminer les "bons" mappings, les probabilités P_i et P_c seront utilisées avec des seuils respectifs S_i et S_c. Les mappings dont les probabilités dépassent leurs seuils seront considérés comme découverts et valides.

- Etant données les distributions de P_i et P_c considérés comme variables aléatoires, le seuil pour P_c devrait logiquement être fixé à une valeur inférieure au seuil pour P_i. Cette dernière est distribuée de faǫn plus resserrée que P_c, son comportement sera donc plus sensible au seuil fixé que pour P_c.
- P_i correspond au modèle le plus proche de la logique. Il donne la même valeur pour un mapping $A_i \sqsubseteq B_j$ et son contraposé $\neg A_i \sqsubseteq B_j$ (on peut étendre les fonctions de probabilités à toutes formules d'inclusions, i.e. avec un $\sqsubseteq$ comme connecteur principal.)

- $P_c(m)$ donne une valeur proche de 1 indépendamment du mapping m si la classe de gauche d'un mapping m est très peu probable ou que sa classe de droite est très probable, P_m est très proche de 1, indépendamment du mapping m, ce qui n'est pas le comportement voulu. P_c gère mieux ces cas, et permet de les filtrer en fonction du mapping.

- P_i et P_c sont complémentaires : chacun permet de distinguer des mappings que l'autre ne distingue pas (on peut le voir en traçant la distribution de P_i sachant que $P_c > S_c$ à S_c fixé.

Enfin, nous avons montré un théorème qui caractérise l'ensemble des fonctions associant une valeur de confiance à un mapping qui sont monotones (fortes) par rapport à l'implication logique.

Sous certaines hypothèses, ces fonctions sont l'ensemble des fonctions de la forme $m \longrightarrow f(P_i(m))$ avec f croissante. Comme pour P_i, f est la fonction identité, P_i est le modèle cohérent avec la logique le plus simple.

Estimation des probabilités

Le calcul de $P_i(m)$ et $P_c(m)$ revient au calcul des probabilités de $P(lhs(m))$ et la probabilité jointe $P(lhs(m) \cap rhs(m))$. Ces valeurs sont inconnues et doivent être estimées. Elles peuvent être modélisées comme des paramètres de variables aléatoires de loi de Bernouilli associé aux évènements $lhs(m)$ et $lhs(m) \cap rhs(m)$. En suivant l'approche statistique bayésienne [Deg04], on modélise ces paramètres comme des variables aléatoires continues de distribution Bêta, et on utilise des observations pour inférer leur distribution a posteriori à partir de leur distribution a priori.

Theorem B.2 (Estimation des probabiités des mapping):
Soit $m : C_i \sqsubseteq D_j$ un mapping entre deux taxonomies $\mathcal{T}_i$ et $\mathcal{T}_j$. Soit $\mathcal{O}$ l'union des instances observées dans $\mathcal{T}_i$ et $\mathcal{T}_j$. Soient $N = |\mathcal{O}|$, $N_i = |\widehat{Ext}(C_i, \mathcal{O})|$, $N_j = |\widehat{Ext}(D_j, \mathcal{O})|$ et $N_{ij} = |\widehat{Ext}(C_i \cap D_j, \mathcal{O})| = |\widehat{Ext}(C_i, \mathcal{O}) \cap \widehat{Ext}(D_j, \mathcal{O})|$

- *$P(C_i)$ est estimé par $\frac{1+\widehat{Ext}(C,\mathcal{O})}{2+N_i} = \frac{1+N_i}{2+N}$*

- *$P(\widehat{C_i \cap D_j})$ est estimé par : $\frac{1+|\widehat{Ext}(C_i \cap D_j,\mathcal{O})|}{4+N} = \frac{1+N_{ij}}{4+N}$*

ce qui conduit à :

- $\widehat{P_i}(m) = 1 + \frac{1+N_{ij}}{4+N} - \frac{1+N_i}{2+N}$

- $\widehat{P_c}(m) = \frac{1+N_{ij}}{4+N} \times \frac{2+N}{1+N_i}$

L'estimation bayésienne est robuste en particulier pour un nombre réduit d'instance, pour lequel la distribution a priori pèse plus dans la formule. (termes 1 au numérateur, et 2 ou 4 au dénominateur).

Exploitation de classifieurs pour fusionner les instances des deux taxonomies

Comme les deux taxonomies sont créées indépendamment et annotent des ensembles d'instances potentiellement disjoints ou avec une intersection trop petite pour être significative, le calcul de l'extension $\widehat{Ext}(C_i \cap D_j, \mathcal{O}_i \cup \mathcal{O}_j)$ qui représente l'extension de l'intersection de deux classes au sein de l'union des instances des deux taxonomies pose problème.

Pour résoudre ce problème, nous utilisons une approche communément utilisée à base de classifieurs (qui peuvent être Naive Bayes, des arbres de décisions, SVM). Un classifieur par classe est entraîné pour apprendre cette classe au sein de la taxonomie qui la déclare. Les exemples pour cette classe est l'extension de cette classe dans cette taxonomie, et les contre-exemples l'ensemble complémentaire de l'extension dans cette taxonomie. Les attributs de données utilisés sont issus des descriptions des instances qui sont disponibles : méta-données, résumé sous forme de texte, etc. prétraités pour devenir des attributs numériques.

Une fois entraîné, chaque classifieur d'une classe d'une taxonomie est appliqué aux instances de l'autre taxonomie, permettant de déterminer pour chacune si elle fait partie ou non de la classe. Ceci permet d'étendre l'extension de chaque classe d'une taxonomie aux instances d'une autre taxonomie.

Les extensions calculées par classification peuvent ainsi être utilisée à la place des extensions initiales dans les formules d'estimation. En particulier, l'intersection de deux extensions de classes obtenues par classification est mécaniquement plus grande qu'initialement.

B.5 ProbaMap: un algorithme de découverte de mappings

Etant données deux taxonomies $\mathcal{T}_i$ and $\mathcal{T}_j$ (et leurs instances associées), soit $\mathcal{M}(\mathcal{T}_i, \mathcal{T}_j)$ l'ensemble de tous les mappings de $\mathcal{T}_i$ à $\mathcal{T}_j$ (de la forme $C_i \sqsubseteq D_j$). L'algorithme ProbaMap détermine tous les mappings m de $\mathcal{M}(\mathcal{T}_i, \mathcal{T}_j)$ $\widehat{P}_u(m) \geq S_u$ et $\widehat{P}_c(m) \geq S_c$, où S_u et S_c sont deux seuils appartenant à $[0, 1]$.

Génération des mappings candidats

Le principe de ProbaMap consiste à générer les mappings à partir des deux ensembles des classes des deux taxonomies à aligner selon un ordre topologique par rapport à la relation d'implication logique. Concrètement, il génère ces mappings à partir de deux boucles imbriquées parcourant les séquences ordonnées des classes dans chacune des taxonomies, l'une selon l'ordre topologique selon la relation d'implication logique, l'autre suivant l'ordre topologique inverse.

Elagage de l'espace des mappings à tester

Grâce à la propriété de monotonie de la fonction de confiance P_i (Theorem B.1), chaque mapping m' qui implique un mapping m tel que $P_i(m) < S_u$ vérifie $P_i(m') < S_u$. Ainsi, ProbaMap élague tous ces mappings en les stockant dans un ensemble. Lorsqu'ils seront généré, ils seront ignorés. De façon similaire, si un mapping m' est testé et qu'on calcule que $P_c(m') < S_c$, on peut

ignorer l'ensemble des mappings qui impliquent m' possédant la même classe à gauche, grâce à la propriété faible de monotonie.

La génération des mappings selon l'ordre induit par l'implication logique garantit que l'élagage est maximum.

L'algorithme ProbaMap

ProbaMap est décrit dans l'Algorithme 8, où

- IMPLICANTS and IMPLICANTS_C sont des fonctions primitives prenant en argument un mapping et retournant respectivement ses implicants ou bien ses implicants qui ont la même classe à gauche.
- REVERSETOPO et TOPO retournent les séquences de classes pour chaque taxonomie en argument en respectant l'ordre de l'implication logique
- ext_i and ext_j sont deux structures associatives (*maps*) qui associent chaque classe de $\mathcal{T}_i$ et $\mathcal{T}_j$ à leur extension (obtenue par classification ou non, selon qu'elle soit activée ou non).

Cet algorithme retourne uniquement les mappings de $\mathcal{T}_i$ à $\mathcal{T}_j$. Pour découvrir l'ensemble des mappings entre ces deux taxonomies, on doit l'appliquer en inversant ses entrées $\mathcal{T}_i$ et $\mathcal{T}_j$.

Algorithm 8 ProbaMap

Require: Taxonomies (DAG) $\mathcal{T}_i, \mathcal{T}_j$, thresholds S_c, S_i, Instances maps ext_i, ext_j
Ensure: return $\{m \in \mathcal{M}(\mathcal{T}_i, \mathcal{T}_j)$ such that $\widehat{P_i}(m) \geq S_i$ *and* $\widehat{P_c}(m) \geq S_c\}$

1: $M_{Val} \leftarrow \emptyset$
2: $M_{NVal} \leftarrow \emptyset$
3: **for all** $C_i \in$ REVERSETOPO($\mathcal{T}_i$) **do**
4: **for all** $D_j \in$ TOPO($\mathcal{T}_j$) **do**
5: let $m = C_i \sqsubseteq D_j$
6: **if** $m \notin M_{NVal}$ **then**
7: **if** $\widehat{P_i}(m, ext_i, ext_j, \mathcal{T}_i, \mathcal{T}_j) \geq S_i$ **then**
8: **if** $\widehat{P_c}(m, ext_i, ext_j, \mathcal{T}_i, \mathcal{T}_j) \geq S_c$ **then**
9: $M_{Val} \leftarrow M_{Val} \cup \{m\}$
10: **else**
11: $M_{NVal} \leftarrow M_{NVal} \cup$ IMPLICANTS_C$(m, \mathcal{T}_j)$ // Pruning using the weak monotony
12: **end if**
13: **else**
14: $M_{NVal} \leftarrow M_{NVal} \cup$ IMPLICANTS$(m, \mathcal{T}_i, \mathcal{T}_j)$ // Pruning using the strong monotony
15: **end if**
16: **end if**
17: **end for**
18: **end for**
19: return M_{Val}

Les mappings découverts sont stockés dans l'ensemble M_{NVal}, ceux qui sont élagués sont stockés dans M_{NVal}.

Concernant l'implémentation, les points qui ont retenu une attention particulière sont les suivants :

1. Calcul en amont de l'algorithme de la fermeture transitive des taxonomies et des extensions des classes
2. L'espace des mappings candidats n'est pas explicite, la structure des mappings candidats n'est pas stockée en mémoire. Ainsi, on ne peut pas l'élaguer directement. L'élagage consiste donc à générer des mappings "à ignorer" dans un ensemble (M_{NVal}). Durant les phases d'élagage, l'implémentation de ProbaMap permet d'éviter d'élaguer des mappings qui l'ont déjà été, permettant ainsi d'éviter d'élaguer de façon redondante des parties de l'ensemble des mappings candidats qui se recouvriraient.

B.6 Expérimentations sur données synthétiques

Nous avons évalué de façon qualitative et quantitative notre algorithme de découverte de mappings ProbaMap présenté dans le chapitre précédent, à la fois sur des données synthétiques et réelles.

Ce chapitre présente les évaluations sur les données synthétiques ainsi que les principes du générateur utilisé. L'évaluation qualitative concerne les critères standard de précision et de rappel : la précision est la proportion de mappings attendus (dans une référence) parmi ceux qui sont retournés par l'algorithme. Le rappel est la proportion de mappings retournés parmi ceux qui sont attendus. L'évaluation quantitative concerne le temps et le gain issu de l'élagage de l'espace de recherche, en vue d'un passage à l'échelle lorsque l'espace de recherche en terme de nombre de mappings possible est de l'ordre de plusieurs millions ou dizaines de millions.

Une évaluation systématique sur des données générées permet de tester ProbaMap sur un large spectre de conditions représentatives, en faisant par exemple varier la taille, le déséquilibre de taille entre taxonomies, le nombre d'instances par classe, ou bien le bruit dans l'annotation des instances. Tous ces paramètres de génération peuvent ainsi être croisés avec les paramètres de ProbaMap, afin d'analyser le comportement de ProbaMap de façon systématique.

Générateur de données synthétiques

La génération de données synthétique est un défi majeur pour nombre de domaines, y compris le nôtre. Elle est décomposée en trois étapes :

1. génération de deux taxonomies de tailles spécifiées comme des forêts d'arbres aléatoires, grâce à une approche existante de génération aléatoire de structures récursives ([DFLS04]) et une méthode de rejet pour filtrer les tailles
2. génération des mappings à découvrir les plus spécifiques. Leur nombre est un paramètre, mais on ne contrôle qu'indirectement le nombre total de mappings à découvrir (ensemble des conséquences de ces mappings générés), par un paramètre
3. génération des instances et de leurs descriptions vectorielles binaires. Le nombre d'instance par classe est un paramètre.

La cohérence entre les descriptions des instances d'une part, et les structures de taxonomies et les mappings d'autre part est primordiale lors du processus de génération. On impose en fait aux instances d'avoir leurs descriptions qui soient cohérentes avec *toutes* les implications entre classes qu'on peut déduire des taxonomies et des mappings, *et seulement celles-là*. Concrètement, chaque classe possède une description intensionnelle dépendante de la structure de taxonomie et des mappings générés. Pour garantir la cohérence, chaque instance générée pour une classe particulière C doit avoir une description vectorielle binaire qui se conforme à la description intensionnelle de C.

Une propriété fondamentale du générateur assure que dans le cas "idéal" où ProbaMap utilise un classifieur qui connaît la description intensionnelle de chaque classe (classifieur oracle), alors P_i et P_c sont très proches de 1 pour les mappings générés ou leurs conséquences, et distants de 1 pour tous les autres mappings, sauf une proportion d'entre eux qui décroît statistiquement vers 0 lorsque le nombre d'instances par classe tend vers l'infini.

Ce générateur possède des points de connexion avec la génération de bases de données d'Armstrong pour les dépendances fonctionnelles : en effet dans celles-ci les tuples doivent vérifier toutes les dépendances, et seulement elles. Cependant, dans notre cas nous devons générer à la fois le schéma et les données. De plus, étant donné que P_c et P_i sont des probabilités, une seule instance ne suffit pas pour contredire un mapping qui n'est pas généré, alors que deux tuples suffisent à contredire une dépendance fonctionnelle. Ces deux différences rendent notre problème de génération plus complexe que celui des bases de données d'Armstrong.

Résultats sur données synthétiques

Nous présentons ci-dessous les principales conclusions des tests systématiques. Tous les tests ont procédé à 100 générations pour chaque instanciation de variables, afin de moyenner les variables "libres" (comme les structures des taxonomies par exemple). ProbaMap a été programmé en Java et testé sur une station de travail Linux Ubuntu 8.10 avec un processeur Intel Xeon Q6700 à 2.66 GHz et 4 Go de RAM.

Les deux premiers points étudient comment combiner de façon optimale les deux modèles de probabilités P_i et P_c. Ensuite, on compare l'efficacité en temps et la qualité de ProbaMap pour trois classifieurs (implémentés dans Weka) appartenant à des familles différentes et représentatives. Enfin, on étudie la robustesse de ProbaMap au bruit dans les données, en particulier dans la description des instances ici.

Impact des seuils S_c et S_i sur la précision

- les deux seuils ont une influence sur la précision → il faut combiner les deux
- P_c est plus discriminant que P_i
- P_i a un pouvoir discrimant pour les grandes valeurs, donc S_i doit être assez haut (proche de 0.9)

Paramètres influençant le gain dû au pruning

- l'élagage est plus important lorsque l'on combine P_c et P_i que lorsqu'on en utilise seulement un des deux (autre seuil correspondant fixé à 0). Plus des trois quarts des mappings candidats sont élagués lorsque l'espace de recherche dépasse les 2000 mappings.
- à tailles égales de l'espace de recherche, l'élagage est plus important lorsque les taxonomies sont de taille très différentes (déséquilibrées) que lorsqu'elles sont de même taille
- plus les mappings à découvrir qui ont été générés sont "spécifiques" (donc ont un grand nombre de conséquences), moins il y a d'élagage.

Influence du classifieur choisi sur le temps et la qualité des résultats

- le choix du classifieur n'affecte pas l'importance de l'élagage
- ProbaMap avec C4.5 et SVM (SMO) fournit un rappel bien meilleur qu'avec Naive Bayes
- ProbaMap avec C4.5 et Naive Bayes va plus vite qu'avec un SVM
- Avec C4.5 (meilleur compromis), le rappel augmente fortement avec le nombre d'instances par classe entre 10 et 60, et est haut à partir de 50 instances par classes. La précision n'est pas vraiment influencée par ce paramètre.

Robustesse au bruit dans la description des instances
On introduit un certain pourcentage de bruit dans les descriptions des instances (proportions de bits inversés dans l'ensemble des descriptions des instances des deux taxonomies).

- le rappel n'est vraiment affecté que par une proportion de bruit supérieure à 15%
- la précision est affectée négativement à partir de 24% de bruit

ProbaMap est donc robuste car ces proportions de bruit sont élevées.

En conclusion, les expérimentations sur données synthétiques montrent que combiner P_c et P_i pour sélectionner les mappings à renvoyer permet d'optimiser la précision et l'élagage de l'espace de recherche (et donc le temps). ProbaMap utilisé avec C4.5 et SVM permet d'avoir un rappel supérieur à celui obtenu avec Naive Bayes, et une précision équivalente.

B.7 Expérimentations sur données réelles

Nous avons confronté ProbaMap à des taxonomies réelles issues de directories web (Yahoo!, Google, Looksmart). Tout d'abord, ceux-ci ont l'avantage de représenter des domaines populaires et généraux, ce qui est cohérent avec la vision du web sémantique que nous avons présenté en introduction, dans lequel les utilisateurs annotent leurs documents eux-mêmes avec des folksonomies. De plus, comme ces directories comportent plusieurs centaines voire milliers de classes, ils permettent de tester le passage à l'échelle de ProbaMap. Enfin, aligner des directories web aussi populaires est directement utile pour l'avènement du Web Sémantique, car il permet d'intégrer de façon cohérente tout le contenu web initialement référencé par plusieurs

organisations. Ainsi, il suffit d'une seule requête pour accéder à toutes les pages concernées et référencées à la fois par Yahoo, Google et Looksmart.

Nous avons effectué deux séries de tests : l'une sur un jeu de donnée de la compétition annuelle OAEI, l'autre sur des directories web collectés, en comparaison avec une autre méthode de l'état de l'art appelée SBI [ITH03].

Tests sur le jeu de donnée directory d'OAEI

Le jeu de données directory d'OAEI a été construit de façon semi-automatique à partir des directories web Looksmart, Yahoo! et Google. Voici les points clefs du contexte expérimental :

- Les taxonomies contiennent respectivement 6628 et 2857 classes, mais ne contiennent aucune instances. L'espace de recherche est plus grand que 30 millions de mappings.

- Les taxonomies sont disponibles sous leur forme complète, ou bien décomposées en paires de branches (environ 4000) à aligner. Tous les participants ne fournissent leurs résultats qu'à partir de ce jeu modifié, alors que nous utilisons directement les structures brutes, ce qui permet de ne pas perdre d'information.

- Comme il n'y a pas d'instances et que notre méthode en nécessite, nous peuplons les classes par des unités linguistiques (synsets) à partir du thésaurus WordNet avec une approche similaire à celle utilisée dans Ctx-Match [SBMZ03]. Pour chaque classe, on tient compte à la fois de son label et du contexte dans lequel elle apparaît (sa hiérarchie) afin de capturer au mieux son sens. Pour certaines classes, ce processus échoue et elles restent non peuplées. Lors de l'évaluation, on peut restreindre le calcul du rappel et de la précision aux seuls mappings faisant intervenir des classes peuplées (cf. ligne wordnet/regular dans la figure B.1).

- Comme ProbaMap découvre des mappings d'inclusions et que la référence pour l'évaluation est constituée d'équivalences, on considère qu'une équivalence est trouvée (1, méthode "min") si les deux inclusions sont trouvées, ou bien (2, méthode "moy") si les deux inclusions sont trouvées avec des seuils plutôt bas et que la moyenne des scores respectifs de P_c et P_i pour les deux inclusions dépasse un certain seuil.

- L'évaluation qualitative se fait en terme de précision et de rappel par rapport à la référence fournie par OAEI. Cependant, on utilise aussi une précision et un rappel sémantiques pour lequel la référence est l'ensemble des mappings d'inclusions impliqués par ceux de la référence modulo les taxonomies, et l'ensemble de mapping évalué est simplement les mappings retournés par ProbaMap. Ces critères semblent plus adaptés a priori car ils tiennent comptent de l'implication logique.

Résultats

Dans ces expériences, S_u est fixé à 0.9, et S_c à 0.8 (sauf si autre valeur indiquée).

Threshold Sc	Semantic measures Precision	Semantic measures Recall wordnet	Semantic measures Recall regular	Standard measures (min) Précision	Standard measures (min) Recall wordnet	Standard measures (min) Recall regular	Standard measures (mean) Precision	Standard measures (mean) Recall wordnet	Standard measures (mean) Recall regular
0.6	41 %	27 %	26 %	47 %	3 %	2 %	37 %	3 %	3 %
0.7	50 %	24 %	23 %	58 %	2 %	2 %	50 %	3 %	3 %
0.8	60 %	20 %	19 %	67 %	2 %	1 %	60 %	2 %	2 %

Figure B.1: Résultats sur une référence partielle

	Yahoo!		Google		shared
	classes	instances	classes	instances	instances
`Autos`	947	4406	967	6425	837
`Outdoors`	2428	5511	1441	13863	623
`Software`	323	2390	2395	30140	572
`Photography`	168	1851	321	3852	286

Table B.1: Statistiques sur subdirectories collectés depuis Yahoo! et Google (Juin 2010)

- Une première expérience (cf. figure B.1) sur une référence partielle fournie par OAEI a permis de fixer S_c à 0.8 pour optimiser le compromis rappel-précision, et de constater que le rappel est bas.

- Les résultats sur la référence complète sont de 1,4% pour le rappel et de 43,4% pour la précision, de 17,5% pour le rappel sémantique et de 48,3% pour la précision sémantique. Les autres méthodes ont une précision autour de 60% et un rappel autour de 50%, mais nous ne connaissons pas leurs résultats en terme de précision et rappel sémantique.

- Ces résultats peuvent être expliqués par un sur-apprentissage des paramètres de la population sur le sous-ensemble retenu pour l'optimiser. Cette phase est capitale étant donné qu'aucune instance n'est fournie. Cette phase est difficile car WordNet est une ressource linguistique fondée un graphe de relations entre entités linguistiques, alors que l'estimation de P_i et P_c est basée sur les tailles des extensions des classes.

- En ce qui concerne l'aspect complexité, ProbaMap admet sans problème l'espace de recherche de 30 millions de mappings.

Evaluation comparative avec SBI sur Yahoo! et Google directories

La seconde série d'expériences porte sur l'alignement de directories collectés depuis Yahoo! et Google. Les données collectées sont directement peuplées d'instances, étant donnés que les catégories des directories sont les classes, et les liens référencés pour chaque catégories les intances. De plus, un résumé de chaque page constitue une description qui est utilisé lors de la phase de classification (si activée).

Proportion d'instances en commun fournies	**SBI**	**ProbaMap**	**ProbaMap + classif**
0.5	0.23	0.28	0.36
0.9	0.29	0.33	0.40

Table B.2: Averaged accuracy for SBI and ProbaMap

Dans cette expérimentation, ProbaMap est comparé à SBI [ITH03, IHT04] un outil d'alignement conçu d'abord pour aligner des directories web. Nous avons programmé et lancé SBI sur les mêmes données que ProbaMap, et comparé les deux en terme d'accuracy, mesure de qualité utilisée par les auteurs de SBI. L'accuracy est une mesure de qualité standard qui ne nécessite pas de mappings de référence. A partir d'un ensemble d'instances appartenant à la fois aux deux taxonomies conservé pour l'évaluation (et non utilisé pour la découverte de mapping), elle indique la proportion de ces instances qui sont cohérentes avec les mappings trouvés.

Afin de mener une comparaison fiable, nous avons ajouté un post-traitement à ProbaMap pour qu'un et un seul mapping $A_1 \sqsubseteq B_2$ soit retourné pour chaque classe A_1 de la première taxonomie, car SBI renvoie ses résultats sous cette forme.

Les variables dans les expériences conduites sont :

- l'activation ou non de la phase de classification
- la proportion d'instances en commun fournie pour la découverte (ce qui reste est utilisé pour calculer l'accuracy).

Résultats

- En moyenne, ProbaMap obtient de meilleurs résultats que SBI, et la version avec classification de meilleurs résultats que les deux autres.
- Lorsqu'il y a peu d'instances par classe, SBI et ProbaMap sont défavorisés, mais la classification permet d'améliorer les résultats dans certains cas.
- ProbaMap est plus robuste que SBI à la diminution du nombre d'instance (lorsqu'elle est uniformément répartie entre les classes), et la version avec classification l'est encore plus.

Ces expériences sont concluantes et montrent que ProbaMap donne de bons résultats sur les directories web, qui représentent de nombreux cas réels. La phase de classification permet d'améliorer les résultats, notamment lorsqu'il n'y a pas assez d'instances en commun.

B.8 Conclusion

Le problème général traité dans cette thèse est l'alignement d'ontologies, qui est une des pierres angulaires pour l'avènement du futur Web Sémantique, par exemple pour permettre un échange de documents collaboratifs entre plusieurs sources. En particulier, nous nous sommes concentrés

sur comment définir une sémantique formelle probabiliste gérant l'incertitude pour des mappings d'inclusions.

Dans cette optique, nous avons présenté deux fonctions probabilistes qui peuvent être utilisées et estimées pour associer un degré de confiance à chaque mapping.

- $P_i(A \sqsubseteq B) = P(\overline{A} \cup B)$
- $P_c(A \sqsubseteq B) = P(B|A)$

En analysant les propriétés théoriques de ces deux modèles, nous avons suggéré qu'une combinaison des deux était la meilleure solution. Nous avons montré une propriété de monotonie de P_i et de P_c par rapport à l'ordre induit par l'implication logique, qui permet de relier directement la sémantique logique avec les probabilités des mappings.

De plus, nous avons montré que sous certaines hypothèses, les seules fonctions de confiance probabilistes monotones étaient de la forme $m \longrightarrow f(P_i(m))$, indiquant que P_i est donc la fonction monotone la plus simple (f = identité).

Les probabilités P_i et P_c pour un mapping peuvent être estimées par une technique bayésienne à partir des extensions des classes mises en jeu dans le mapping. Dans le cas où les deux taxonomies sont peuplées par des ensembles d'instances disjoints et que les instances sont annotées par des descriptions, on utilise des classifieurs pour fusionner les instances des deux taxonomies.

Nous avons conçu et implémenté un algorithme de découverte de mapping basé sur P_i et P_c. Cet algorithme est de type "générer et tester" et renvoie tous les mappings possibles entre deux taxonomies pour lesquels P_c et P_i dépassent deux seuils fournis en entrée. ProbaMap exploite la propriété de monotonie de P_c et de P_i pour élaguer son espace de recherche constitué de l'ensemble des mappings possibles entre les deux taxonomies en entrée.

Nous avons effectué deux séries d'expérimentations pour valider notre approche. La première consiste en une analyse quantitative et qualitative approfondie du comportement de ProbaMap sur des données synthétiques contrôlées. Nous avons conçu un générateur complet de taxonomies, mappings et instances, pour lequel différents paramètres peuvent être réglés : taille des taxonomies, nombre d'instances par classe, etc. En croisant les paramètres du générateur et ceux de ProbaMap, nous avons confirmé que P_c et P_i doivent être combinés pour obtenir à la fois de meilleurs résultats en terme de précision et aussi en terme de temps. Nous avons aussi montré que ProbaMap est robuste lorsqu'il y a du bruit dans les descriptions des instances, et qu'il nécessitait un nombre minimal mais limité d'instances par classe. La seconde série d'expérimentations a été conduite sur des taxonomies issues de directories Web (Yahoo!, Google). Les résultats sont prometteurs en terme de passage à l'échelle et de qualité. En particulier, ProbaMap a montré de meilleurs résultats que SBI [ITH03], un algorithme de l'état de l'art dédié à l'alignement de directories Web.

Perspectives

Nous avons deux perspectives principales. La première consiste à étudier et concevoir un système pour répondre de façon probabiliste à des requêtes par réécriture en ré-utilisant les probabilités associées aux mappings découverts, dans l'esprit des bases de données probabilistes. Nous envisageons une approche fondée sur du raisonnement probabiliste. Plusieurs travaux introduisent les

probabilités dans la logique et le raisonnement (Probability Logic - Introduction dans [Ada98], P-CLASSIC [KLP97] P-$\mathcal{SHIQ}$ [CFL+08]), et nous comptons nous en inspirer.

La seconde perspective étudie le moyen de tirer parti de la qualité et de l'efficacité des méthodes d'alignement existantes, tout en garantissant que les coefficients associés aux mappings qu'elle renvoient respectent une propriété de monotonie (cf. Définition B.1), afin de pouvoir être interprétés comme des probabilités. Cette perspective consiste ainsi à formuler et implémenter un post-traitement à ces méthodes pour transformer leurs coefficients, à l'aide du principe de similarity flooding [MGR+02] dans l'esprit de N2R [SPR09] et d'OLA [EV04].

List of Figures

I.1 Size evolution of the World Wide Web (hosts) - ISC estimation (Feb. 2010) . . 1
I.2 Web size and indexation coverage 2
I.3 Semantic annotation of Web pages 4
I.4 Semantic Web Stack [BL00] 4
I.5 2 small personalized taxonomies 6
I.6 Figures about the Swoogle index 7
I.7 Heterogeneity of ontologies: parallel with maps 9

II.1 2 Taxonomies and associated instances 14
II.2 2 mappings between $\mathcal{T}_1$ and $\mathcal{T}_2$ 17

IV.1 Venn diagrams of two class extensions 42
IV.2 Case where $P_c(E'_1 \sqsubseteq F_2) = 0$ whereas $P_c(E_1 \sqsubseteq F_2) > 0$ and $E'_1 \sqsubseteq E_1$ 47
IV.3 Distribution histograms for P_i and P_c 47
IV.4 Distribution histogram for P_i given $P_c > 0.7$ 49
IV.5 Classifications of instances 54

V.1 Two taxonomies and associated DAG of mapping 57
V.2 Example of redundant pruning 64

VI.1 Example of result provided by the generation process 79
VI.2 Counter-examples on constraints applied to mappings generation 81
VI.3 Example of generation of intentional descriptions for each class 83
VI.4 Precision w.r.t. S_c and S_i thresholds 87
VI.5 Pruning factor with only $\widehat{P_c}$, only $\widehat{P_i}$ and $\widehat{P_c}$ *and* $\widehat{P_i}$ 88
VI.6 Pruning factor for unbalanced and balanced taxonomies 89
VI.7 Pruning factor w.r.t. specificity of seed mappings 90
VI.8 Pruning factor for different classifiers 90
VI.9 Computation time (s) for different classifiers 91
VI.10 Computation time in log scale for different classifiers 91
VI.11 Precision for different classifiers 92
VI.12 Recall for different classifiers 93
VI.13 C4.5 - Computation time (s) - impact of number of inst/class 93
VI.14 C4.5 - Precision - impact of number of inst/class 94
VI.15 C4.5 - Recall - impact of number of inst/class 94
VI.16 C4.5 - Precision and Recall - impact of noise level 95

VII.1 Top levels of Yahoo! and Google directories 98
VII.2 Samples of the OAEI Directory benchmark 99

VII.3 Results based on the partial reference 103
VII.4 Accuracy w.r.t. the threshold S_c 106
VII.5 `Autos`: Comparative accuracy for SBI and ProbaMap 107
VII.6 `Outdoors`: Comparative accuracy for SBI and ProbaMap 108
VII.7 `Software`: Comparative accuracy for SBI and ProbaMap 109
VII.8 `Photography`: Comparative accuracy for SBI and ProbaMap 110
VII.9 Comparative accuracy results using classification (1) 113
VII.10 Comparative accuracy results using classification (2) 114

B.1 Résultats sur une référence partielle 144

REFERENCES

[ACG+05] P. Adjiman, P. Chatalic, F. Goasdoué, M. C Rousset, and L. Simon. Somewhere in the Semantic Web. *Lecture notes in computer science*, 3703:1, 2005.

[ACG+06] Philippe Adjiman, Philippe Chatalic, François Goasdoué, Marie-Christine Rousset, and Laurent Simon. Distributed reasoning in a Peer-to-Peer setting: Application to the semantic web. *Journal of Artificial Intelligence Research (JAIR)*, 25:269–314, 2006.

[Ada98] E. Adams. *A Primer of Probability logic, CSLI.* Stanford University, Stanford, California, 1998.

[Adj06] Philippe Adjiman. *Peer-to-Peer reasoning in propositional logic: algorithms, scalability study and applications.* PhD thesis, Université Paris Sud, 2006.

[ADMR05] David Aumueller, Hong H. Do, Sabine Massmann, and Erhard Rahm. Schema and ontology matching with coma++. In *SIGMOD '05: Proceedings of the 2005 ACM SIGMOD international conference on Management of data*, pages 906–908, New York, NY, USA, 2005. ACM.

[AS01] R. Agrawal and R. Srikant. On integrating catalogs. In *Proceedings of the 10th international conference on World Wide Web*, page 603–612, 2001.

[BDFS84] C. Beeri, M. Dowd, R. Fagin, and R. Statman. On the structure of armstrong relations for functional dependencies. *Journal of the ACM (JACM)*, 31(1):30–46, 1984.

[Biz09] C. Bizer. The emerging Web of linked data. *IEEE Intelligent Systems*, pages 87–92, 2009.

[BL00] T. Berners-Lee. Semantic Web on XML. Keynote presentation for XML 2000, 2000.

[BSHW06] Omar Benjelloun, Anish Das Sarma, Alon Y. Halevy, and Jennifer Widom. Uldbs: Databases with uncertainty and lineage. In *VLDB*, pages 953–964, 2006.

[BSZ03] P. Bouquet, L. Serafini, and S. Zanobini. Semantic coordination: a new approach and an application. *The SemanticWeb-ISWC 2003*, pages 130–145, 2003.

[C.98] Fellbaum C. *WordNet: An Electronic Lexical Database (Language, Speech, and Communication).* The MIT Press, May 1998.

[CFL+08] S. Castano, A. Ferrara, D. Lorusso, T. H. Näth, and R. Möller. Mapping validation by probabilistic reasoning. In *Proc. 5th European Semantic Web Conference*, 2008.

[CFM03] Silvana Castano, Alfio Ferrara, and Stefano Montanelli. H-match: an algorithm for dynamically matching ontologies in peer-based systems. In *Proc. of the Semantic Web Databases Workshop (SWDB)*, pages 231–250, 2003.

[CFM06] S. Castano, A. Ferrara, and G. Messa. Results of the HMatch ontology matchmaker in OAEI 2006. In *Proceedings of the ISWC 2006 Workshop on Ontology Matching, Athens, GA, USA*, 2006.

[CHKP07] Laura Chiticariu, Mauricio A. Hernández, Phokion G. Kolaitis, and Lucian Popa. Semi-automatic schema integration in clio. In *VLDB*, pages 1326–1329, 2007.

[CLRS01] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms, Second Edition.* The MIT Press, September 2001.

[DCBM09] Fabien Duchateau, Remi Coletta, Zohra Bellahsene, and Renée J. Miller. Yam: a schema matcher factory. In *CIKM*, pages 2079–2080, 2009.

[DDL00] Anhai Doan, Pedro Domingos, and Alon Y. Levy. Learning mappings between data schemas. In *Proceedings of the AAAI-2000 Workshop on Learning Statistical Models from Relational Data*, 2000.

[Deg04] Morris H. Degroot. *Optimal Statistical Decisions (Wiley Classics Library).* Wiley-Interscience, April 2004.

[DFLS04] Philippe Duchon, Philippe Flajolet, Guy Louchard, and Gilles Schaeffer. Boltzmann samplers for the random generation of combinatorial structures. *Comb. Probab. Comput.*, 13(4-5):577–625, 2004.

[DGGB06] J. David, F. Guillet, R. Gras, and H. Briand. An interactive, asymmetric and extensional method for matching conceptual hierarchies. In *EMOI-INTEROP Workshop, Luxembourg*, 2006.

[DH05] A. H. Doan and A. Y. Halevy. Semantic integration research in the database community: A brief survey. *AI magazine*, 26(1):83, 2005.

[DHY07] Xin Luna Dong, Alon Y. Halevy, and Cong Yu. Data integration with uncertainty. In *VLDB*, pages 687–698, 2007.

[DMDH02] AnHai Doan, Jayant Madhavan, Pedro Domingos, and Alon Y. Halevy. Learning to map between ontologies on the Semantic Web. In *WWW*, pages 662–673, 2002.

[DR02] Hong Hai Do and Erhard Rahm. COMA - a system for flexible combination of schema matching approaches. In *VLDB*, 2002.

[DS04] Mike Dean and Guus Schreiber. OWL Web Ontology Language Reference. W3C recommendation, W3C, February 2004.

[DS05] Nilesh N. Dalvi and Dan Suciu. Answering queries from statistics and probabilistic views. In *VLDB*, pages 805–816, 2005.

[EFH+09] J. Euzenat, A. Ferrara, L. Hollink, A. Isaac, C. Joslyn, V. Malaisé, C. Meilicke, A. Nikolov, J. Pane, M. Sabou, et al. Results of the ontology alignment evaluation initiative 2009. In *Fourth International Workshop on Ontology Matching, Washington, DC*, 2009.

[Ehr07] M. Ehrig. *Ontology alignment: bridging the semantic gap*. Springer-Verlag New York Inc, 2007.

[ES07] Jérôme Euzenat and Pavel Shvaiko. *Ontology matching*. Springer-Verlag, Heidelberg (DE), 2007.

[Euz07] J. Euzenat. Semantic precision and recall for ontology alignment evaluation. In *Proc. 20th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 348–353, 2007.

[EV04] Jérôme Euzenat and Petko Valtchev. Similarity-based ontology alignment in owl-lite. In *ECAI*, pages 333–337, 2004.

[Fag82] Ronald Fagin. Horn clauses and database dependencies. *Journal ACM*, 29(4):952–985, 1982.

[FDP+05] Timothy W. Finin, Li Ding, Rong Pan, Anupam Joshi, Pranam Kolari, Akshay Java, and Yun Peng. Swoogle: Searching for knowledge on the semantic web. In *AAAI*, pages 1682–1683, 2005.

[FHM90] R. Fagin, J. Y Halpern, and N. Megiddo. A logic for reasoning about probabilities* 1. *Information and computation*, 87(1-2):78–128, 1990.

[FL02] Gary William Flake and Steve Lawrence. Efficient SVM regression training with SMO. *Mach. Learn.*, 46(1-3):271–290, 2002.

[Gal06] Avigdor Gal. Managing uncertainty in schema matching with top-k schema mappings. *Journal on Data Semantics*, 6:2006, 2006.

[GATM05] Avigdor Gal, Ateret Anaby-Tavor, Alberto Trombetta, and Danilo Montesi. A framework for modeling and evaluating automatic semantic reconciliation. *The VLDB Journal*, 14(1):50–67, 2005.

[GGMO03] A. Gangemi, N. Guarino, C. Masolo, and A. Oltramari. Sweetening wordnet with DOLCE. *AI magazine*, 24(3):13, 2003.

[GS05] A. Gulli and A. Signorini. The indexable web is more than 11.5 billion pages, 2005.

[GSY04] Fausto Giunchiglia, Pavel Shvaiko, and Mikalai Yatskevich. S-match: an algorithm and an implementation of semantic matching. In *Proceedings of ESWS*, pages 61–75, 2004.

[Gut05] A. Gut. *Probability: a graduate course*. Springer Verlag, 2005.

[GWW05] B. Ganter, R. Wille, and R. Wille. *Formal concept analysis*. Springer Berlin, 2005.

[GYAS09] F. Giunchiglia, M. Yatskevich, P. Avesani, and P. Shivaiko. A large dataset for the evaluation of ontology matching. *The Knowledge Engineering Review*, 24(02):137–157, 2009.

[Hay04] Patrick Hayes, editor. *RDF Semantics*. W3C Recommendation. World Wide Web Consortium, February 2004.

[HQ08] Wei Hu and Yuzhong Qu. Falcon-AO: a practical ontology matching system. *Web Semantics: Science, Services and Agents on the World Wide Web*, 6(3):237–239, September 2008.

[HSNR09] F. Hamdi, B. Safar, N. Niraula, and C. Reynaud. TaxoMap in the OAEI 2009 alignment contest. In *OAEI 2009 Campaign, Workshop ISWC'09*, 2009.

[HSRZ09] F. Hamdi, B. Safar, C. Reynaud, and H. Zargayouna. Alignment-based partitioning of large-scale ontologies. In *Advances in Knowledge discovery and Management, Studies in Computational Intelligence*. Springer, 2009.

[IHT04] Ryutaro Ichise, Masahiro Hamasaki, and Hideaki Takeda. Discovering relationships among catalogs. *In Einoshin Suzuki and Setsuo Arikawa, Editors, Discovery Science*, 3245:371–379, 2004.

[ITH03] R. Ichise, H. Takeda, and S. Honiden. Integrating multiple internet directories by instance-based learning. In *International Joint Conference on Artificial Intelligence*, volume 18, pages 22–30, 2003.

[IvdMSW07] Antoine Isaac, Lourens van der Meij, Stefan Schlobach, and Shenghui Wang. An empirical study of instance-based ontology matching. In *ISWC/ASWC*, pages 253–266, 2007.

[JK07] Y. Jean-Mary and M. Kabuka. ASMOV: ontology alignment with semantic validation. In *Proc. of Joint SWDB-ODBIS Workshop on Semantics, Ontologies, Databases, Vienna (Austria)*, 2007.

[KCRF+04] R. Khattree, A. Cr Rao, J. L Fleiss, B. Levin, and M. Cho. Statistical methods for rates and proportions. *Technometrics*, 46(2):263–264, 2004.

[KLP97] D. Koller, A. Levy, and A. Pfeffer. P-CLASSIC: a tractable probablistic description logic. In *Proceedings of the National Conference on Artificial Intelligence*, pages 390–397, 1997.

[LC00] Wen-Syan Li and Chris Clifton. SEMINT: a tool for identifying attribute correspondences in heterogeneous databases using neural networks. *Data Knowl. Eng.*, 33(1):49–84, 2000.

[Lev66] V. Levenshteiti. Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet Physics-Doklady*, volume 10, 1966.

[LS08] F. Lin and K. Sandkuhl. A survey of exploiting WordNet in ontology matching. *Artificial Intelligence in Theory and Practice II*, pages 341–350, 2008.

[LTLL08] J. Li, J. Tang, Y. Li, and Q. Luo. RiMOM: a dynamic multistrategy ontology alignment framework. *IEEE Transactions on Knowledge and Data Engineering*, page 1218–1232, 2008.

[MBDH05] Jayant Madhavan, Philip A. Bernstein, AnHai Doan, and Alon Halevy. Corpus-based schema matching. *International Conference on Data Engineering*, 0:57–68, 2005.

[MBR01] Jayant Madhavan, Philip A. Bernstein, and Erhard Rahm. Generic schema matching with Cupid. In *The VLDB Journal*, pages 49–58, 2001.

[MCWD06] C. Matuszek, J. Cabral, M. Witbrock, and J. DeOliveira. An introduction to the syntax and content of Cyc. In *Proceedings of the 2006 AAAI Spring Symposium on Formalizing and Compiling Background Knowledge and Its Applications to Knowledge Representation and Question Answering*, pages 44–49, 2006.

[MGR+02] S. Melnik, H. Garcia-Molina, E. Rahm, et al. Similarity flooding: A versatile graph matching algorithm and its application to schema matching. In *Proceedings of the International Conference on Data Engineering*, page 117–128, 2002.

[Mit97] T. Mitchell. *Machine Learning.* McGraw-Hill Education (ISE Editions), 1997.

[MNJ05] P. Mitra, N. F Noy, and A. R Jaiswal. OMEN: A probabilistic ontology mapping tool. *Lecture notes in computer science*, 3729:537, 2005.

[MP06] M. Mao and Y. Peng. PRIOR system: Results for OAEI 2006. *Proceedings of the Ontology Alignment Evaluation Initiative*, page 165–172, 2006.

[MPP+08] B. Motik, P. F Patel-Schneider, B. Parsia, C. Bock, A. Fokoue, P. Haase, R. Hoekstra, I. Horrocks, A. Ruttenberg, U. Sattler, et al. OWL 2 Web Ontology Language: Structural specification and functional-style syntax. *World Wide Web Consortium, Working Draft WD-owl2-semantics-20081202*, 2008.

[Nil86] N. J Nilsson. Probabilistic logic* 1. *Artificial intelligence*, 28(1):71–87, 1986.

[NS07] H. Nottelmann and U. Straccia. Information retrieval and machine learning for probabilistic schema matching. *Information Processing and Management*, 43(3):552–576, 2007.

[Qui93] Ross J. Quinlan. *C4.5: Programs for Machine Learning (Morgan Kaufmann Series in Machine Learning).* Morgan Kaufmann, January 1993.

[RB01] Erhard Rahm and Philip A. Bernstein. A survey of approaches to automatic schema matching. *VLDB Journal*, 10(4):334–350, 2001.

[Res99] P. Resnik. Semantic similarity in a taxonomy: An information-based measure and its application to problems of ambiguity in natural language. *Journal of artificial intelligence research*, 11(95):130, 1999.

[RMZ03] Ganesh Ramesh, William Maniatty, and Mohammed Javeed Zaki. Feasible itemset distributions in data mining: theory and application. In *PODS*, pages 284–295, 2003.

[SBMZ03] L. Serafini, P. Bouquet, B. Magnini, and S. Zanobini. An algorithm for matching contextualized schemas via SAT. *Proceedings of CONTEXT'03*, 2003.

[SE05] Pavel Shvaiko and Jérôme Euzenat. A survey of Schema-Based matching approaches. *Journal of Data Semantics IV*, pages 146–171, 2005.

[SE08] Pavel Shvaiko and Jérôme Euzenat. Ten challenges for ontology matching. In *On the Move to Meaningful Internet Systems: OTM 2008*, pages 1164–1182. Springer Berlin / Heidelberg, 2008.

[SM01] G. Stumme and A. Maedche. FCA-MERGE: Bottom-Up Merging of Ontologies. In *Proc. of the 17th International Joint Conference on Artificial Intelligence*, pages 225–234, 2001.

[SMS02] A. Sotnykova, S. Monties, and Stefano Spaccapietra. Semantic Integration in MADS Conceptual Model. In *In Heterogeneous InformationExchange and Organizational Hubs*. Kluwer, 2002.

[SPR09] F. Saîs, N. Pernelle, and M. C Rousset. Combining a logical and a numerical method for data reconciliation. *Journal on Data Semantics XII*, pages 66–94, 2009.

[ST05] U. Straccia and R. Troncy. oMAP: combining classifiers for aligning automatically OWL ontologies. *Web Information Systems Engineering-WISE 2005*, pages 133–147, 2005.

[SVV08] V. Spiliopoulos, A. G Valarakos, and G. A Vouros. CSR: discovering subsumption relations for the alignment of ontologies. In *Proceedings of the 5th European Semantic Web Conference (ESWC) on The Semantic Web: research and applications*, page 418–431, 2008.

[TSK06] P. N Tan, M. Steinbach, and V. Kumar. *Introduction to data mining*. Pearson Addison Wesley Boston, 2006.

[VR75] C. J. Van Rijsbergen. *Information retrieval*. Butterworths, London ; Boston :, 1975.

[WES08] Shenghui Wang, Gwenn Englebienne, and Stefan Schlobach. Learning concept mappings from instance similarity. In *Proceedings of the 7th International Conference on The Semantic Web*, pages 339–355, Karlsruhe, Germany, 2008. Springer-Verlag.

[WF05] Ian H. Witten and Eibe Frank. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann, 2 edition, 2005.

[WX09] P. Wang and B. Xu. Lily: Ontology alignment results for oaei 2009. *Shvaiko et al.[SEG+ 09]*, 2009.

[Zad65] L. A. Zadeh. Fuzzy sets*. *Information and control*, 8(3):338–353, 1965.

Mots-clefs :
alignement d'ontologies, Web sémantique, correspondance, probabilités, logique, instances, taxonomie, passage á l'échelle, Web Sémantique

Résumé :
Dans cette thèse, nous adoptons une approche formelle pour définir et découvrir des mappings d'inclusion probabilistes entre deux taxonomies avec une sémantique claire, dans l'optique d'échange collaboratif de documents. Nous comparons deux façons de modéliser des mappings probabiliste tout en étant compatible avec les contraintes logiques déclarées dans chaque taxonomie selon une propriété de monotonie, puis nous montrons que ces modèles sont complémentaires pour distinguer les mappings pertinents. Nous fournissons un moyen d'estimer les probabilités d'un mapping par une technique bayésienne basée sur les statistiques des extensions des classes impliquées dans le mapping. Si les ensembles d'instances sont disjoints, on utilise des classifieurs pour les fusionner. Nous présentons ensuite un algorithme de type "générer et tester" qui utilise les deux modèles de mappings pour découvrir les plus probables entre deux taxonomies. Nous menons une analyse expérimentale fouillée de ProbaMap. Nous présentons un générateur de données synthétiques qui produit une entrée contrôlée pour une analyse quantitative et qualitative sur un large spectre de situations. Nous présentons aussi deux séries de résultats d'expériences sur des données réelles : l'alignement du jeu de donnée "Directory" d'OAEI, et une comparaison pour l'alignement de Web Directories sur lesquels ProbaMap obtient de meilleurs résultats que SBI (IJCAI 2003). Les perspectives pour ces travaux consistent à concevoir un système de réponse à des requêtes probabilistes en réutilisant des mappings probabilites, et la conversion des coefficients retournés par les méthodes de matching existantes en probabilités.

Keywords:
ontology matching, semantic web, mapping, probability, logic, instances, taxonomy, scalability, taxonomy, Semantic Web

Abstract:
In this thesis, we investigate a principled approach for defining and discovering probabilistic inclusion mappings between two taxonomies, with a clear semantic, in a purpose of collaborative exchange of documents. Firstly, we compare two ways of modeling probabilistic mappings which are compatible with the logical constraints declared in each taxonomy according to a monotony property, then we show that they are complementary for distinguishing relevant mappings. We provide a way to estimate the probabilities associated to a mapping by a Bayesian estimation technique based on classes extensions involved in the mapping, and using classifiers in order to merge the instances of both taxonomies when they are disjoint. Then we describe a generate and test algorithm called ProbaMap which minimizes the number of calls to the probability estimator for determining those mappings whose probability exceeds a chosen threshold. A thorough experimental analysis of ProbaMap is conducted. We introduce a generator that produce controlled data that allows to analyse the quality and the complexity of ProbaMap in a large and generic panel of situations. We present also two series of results for experiments conducted on real-world data: an alignment of the Directory dataset of the Ontology Alignment Evaluation Initiative (OAEI), and a comparative experiment on Web directories, on which ProbaMap outperforms the state-of-the-art contribution SBI (IJCAI'03). The perspectives of this work are the reuse of probabilistic mappings for a probabilistic query answering setting and a way to convert similarities coefficients of existing matching methods into probabilities.