Semantic Enrichment of Ontology Mappings

Von der Fakultät für Mathematik und Informatik
der Universität Leipzig
angenommene

DISSERTATION

zur Erlangung des akademischen Grades

Doctor Rerum Naturalium
(Dr. rer. nat.)
im Fachgebiet Informatik

Vorgelegt

von Patrick Arnold, M. Sc. Informatik,
geboren am 26. Juli 1987 in Leipzig

Die Annahme der Dissertation haben empfohlen:

1. Prof. Dr. Erhard Rahm (Universität Leipzig)
2. Prof. Dr. Sören Auer (Universität Bonn)

Die Verleihung des akademischen Grades erfolgt mit Bestehen der Verteidigung am 15. Dezember 2015 mit dem Gesamtprädikat *magna cum laude*.

Acknowledgement

First, and foremost, I want to sincerely thank Prof. Rahm for his supervision and support, his encouragement and patience, as well as his professional advice. Much of the achievements and outcomes of this work would not have been possible without his guidance and counsel, and I am entirely convinced that it also added a lot to my personal development as a researcher in the field of computer science.

The support I obtained from my colleagues at the database department is ineffable, and I would never have believed how much I could learn from them within this time. I want to thank them for the many important pieces of advice they gave me, and for the help they offered whenever help was needed. I also want to thank them for the wonderful time we spent together; our daily lunch times, our annual summer seminars in Zingst and our lively group discussions shall always remain in my memories.

In particular, I want to thank Christian Wartner, who introduced me to the database department and remained a very close colleague for many years. I also thank Sabine Maßmann, who encouraged me still in my student days to become a researcher at our database department. I also want to thank Anika Groß and Eric Peukert, who gave me much support and advice for my research topic and PhD thesis. I thank Marcel Jacob and Max Möller, whose assistance to the project was very fruitful. Eventually, I want to thank our secretary Andrea Hesse, who always takes care about the many formalities and paperwork that are constantly accumulating, and whose daily help is beyond any description. I am very grateful to my family, who enabled my education and supported me throughout the entire time. I am also very grateful to the many proof-readers who helped me on very short notice, and who gave very important suggestions for improvement.

Additionally, I should not fail to mention that the many conferences and workshops I attended have been very supportive to me; they were very great experiences. There are only few people I could call by name, but the advice gained from them, reviewers, conference participants and fellow researchers alike, have been very crucial for my work.

At last, I want to express my deep gratitude for the European Commission and the Federal State of Saxony that funded my research at our grand Alma Mater. They are the actual institutions that made this research possible. I also want to thank the InfAI Leipzig e.V. that allowed me three years of participation in the Linked Design project, and Sebastian Fuß, who always took care about organizational and employment matters.

Leipzig, July 2015

Patrick Arnold

Abstract

Schema and ontology matching play an important part in the field of data integration and semantic web. Given two heterogeneous data sources, meta data matching usually constitutes the first step in the data integration workflow, which refers to the analysis and comparison of two input resources like schemas or ontologies. The result is a list of correspondences between the two schemas or ontologies, which is often called mapping or alignment. Many tools and research approaches have been proposed to automatically determine those correspondences. However, most match tools do not provide any information about the relation type that holds between matching concepts, for the simple but important reason that most common match strategies are too simple and heuristic to allow any sophisticated relation type determination.

Knowing the specific type holding between two concepts, e.g., whether they are in an equality, subsumption (is-a) or part-of relation, is very important for advanced data integration tasks, such as ontology merging or ontology evolution. It is also very important for mappings in the biological or biomedical domain, where is-a and part-of relations may exceed the number of equality correspondences by far. Such more expressive mappings allow much better integration results and have scarcely been in the focus of research so far.

In this doctoral thesis, the determination of the correspondence types in a given mapping is the focus of interest, which is referred to as semantic mapping enrichment. We introduce and present the mapping enrichment tool Stroma, which obtains a pre-calculated schema or ontology mapping and for each correspondence determines a semantic relation type. In contrast to previous approaches, we will strongly focus on linguistic laws and linguistic insights. By and large, linguistics is the key for precise matching and for the determination of relation types. We will introduce various strategies that make use of these linguistic laws and are able to calculate the semantic type between two matching concepts. The observations and insights gained from this research go far beyond the field of mapping enrichment and can be also applied to schema and ontology matching in general.

Since generic strategies have certain limits and may not be able to determine the relation type between more complex concepts, like a *laptop* and a *personal computer*, background knowledge plays an important role in this research as well. For example, a thesaurus can help to recognize that these two concepts are in an is-a relation. We will show how background knowledge can be effectively used in this instance, how it is possible to draw conclusions even if a concept is not contained in it, how the relation types in complex paths can be resolved and how time complexity can be reduced by a so-called bidirectional search. The developed techniques go far beyond the background knowledge exploitation of previous approaches, and are now part of the semantic repository SemRep, a flexible and extendable system that combines different lexicographic resources.

Further on, we will show how additional lexicographic resources can be developed automatically by parsing Wikipedia articles. The proposed Wikipedia relation extraction approach yields some millions of additional relations, which constitute significant additional knowledge for mapping enrichment. The extracted relations were also added to SemRep, which thus became a comprehensive background knowledge resource. To augment the quality of the repository, different techniques were used to discover and delete irrelevant semantic relations.

We could show in several experiments that STROMA obtains very good results w.r.t. relation type detection. In a comparative evaluation, it was able to achieve considerably better results than related applications. This corroborates the overall usefulness and strengths of the implemented strategies, which were developed with particular emphasis on the principles and laws of linguistics.

Contents

Part I

Introduction

1 Introduction

1.1 Motivation

For many years, schema and ontology matching are the intense focus of research and constitute a key element in the field of data integration. Given two database schemas resp. ontologies describing the logical structure of a database resp. the logical structure of a certain domain, these techniques are used to find the corresponding elements between the two schemas or ontologies. They are usually the first step in data integration and the result is a set of correspondences, i.e., a set of links between elements from the two schemas or ontologies. Such a set of correspondences is called *mapping* or *alignment*. Fig. 1.1 shows such a mapping between the two database schemas *Person* and *Employee*, consisting of 4 correspondences. Between *First Name*, *Last Name* (left schema) and *Name* (right schema) is an (n:1)-correspondence, which is also called *complex correspondence*. The remaining correspondences are *(1:1)-correspondences*.

Knowing how elements between the two schemas are related is an essential prerequisite for subsequent data integration steps. For example, a third schema could be derived from the two given schemas that would cover all elements from both schemas. This step is called *schema integration* or *merging*. It is also possible to transform all data objects from one database to the other database. For example, if two companies start to cooperate and decide to use only one database system instead of two, they may decide to take the database schema *Employee* to represent their data. The mapping is then used to copy data from the database *Person* (called *source database*) to the database *Employee* (called *target database*) and this process is usually called *data transformation* or *data translation* [19]. The schemas of source and target systems are analogously called *source* and *target schema* (or *ontology*).

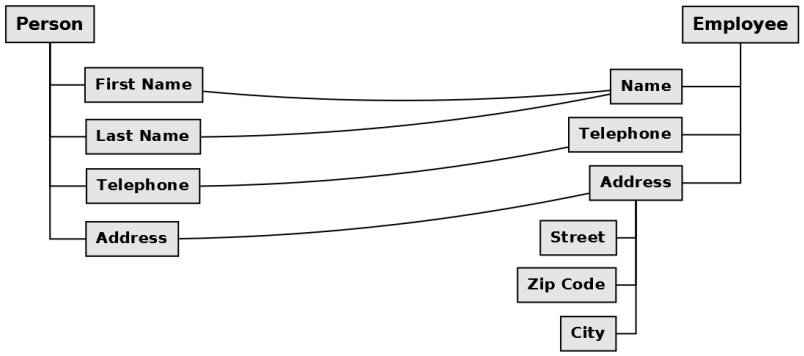

Figure 1.1: Sample mapping between two database schemas.

A large body of scientific work, survey papers, research prototypes and applications are the outcome of this schema and ontology matching research that spawned a variety of techniques to determine the relatedness of two schema elements or ontology concepts [132, 17, 133]. For example, match approaches can discover the relatedness between the two concepts *last name* and *surname* or *Student Id* and *STD_ID*, based on their lexicographic similarity. In fact, lexicographic and linguistic strategies are widely used in current approaches, often combined with reasoning [81], probabilistic techniques [144, 38] or machine learning [43, 138]. Other approaches focus on structural relatedness [93] or analyze instance data to find those relations [146, 36, 103]. Further on, background knowledge is widely exploited by many tools, either in the form of a dictionary or a domain-specific background knowledge ontology. Such resources are especially helpful to discover synonymous concepts, e.g., *laptop* and *notebook*, although the existence of comprehensive and reliable background knowledge sources is quite limited. Though being more than 30 years old, WordNet practically remains the only reliable English general purpose resource for the field of schema and ontology matching [48].

Still, whatever techniques or combinations of techniques have been used so far, schema and ontology matching tools remain far from being perfect. They support data integration tasks and can considerably reduce the effort of manual matching, but the generally high degree of heterogeneity between two given schemas resp. ontologies cannot be fully overcome with present methodologies. Mappings remain incomplete, erroneous and imprecise, as many techniques are not sophisticated enough for the high heterogeneity between ontologies, and the high complexity of languages. In fact, a strikingly low number of approaches actually draws on linguistic knowledge. Such a neglect can quickly result in false correspondences, e.g., between two concepts *stable* and *table*, which are lexicographic very similar, but for which there is no linguistic evidence for any semantic relation. To a high degree, linguistics is the key to accurate, high-quality matching and so far has not been thoroughly and satisfyingly exploited in this research area.

The correspondences produced by common match tools are often unspecific, suggesting that two elements or ontology concepts are somehow related, but without specifying what exact relation holds. Of course, given the frequently rather heuristic and limited techniques that are applied to determine the correspondences, such a specification is

simply impossible. Some tools focus directly on equivalence relations and a correspondence between two elements suggests that the two elements express the same thing. In larger match scenarios, such true equivalence relations become quite rare, though, and more specific types like subsumption (`is-a`) or aggregation (`part-of`) may occur more frequently. Other approaches may be able to handle different kind of relation types, but do not specify the correspondence type in the mapping, for the simple reason that the determination of a correspondence type is more difficult than the determination of the mere relatedness between two elements. For example, a match tool can discover the relatedness between two terms *city hall* and *city* because of the lexicographic similarity, but to specify the relation type (`part-of`) is much more difficult. Without relation type specification, the correspondence could be anything like `equal`, `is-a`, `part-of` or `related` and the mapping is too unspecific for more specific data integration issues like ontology merging or mapping evolution.

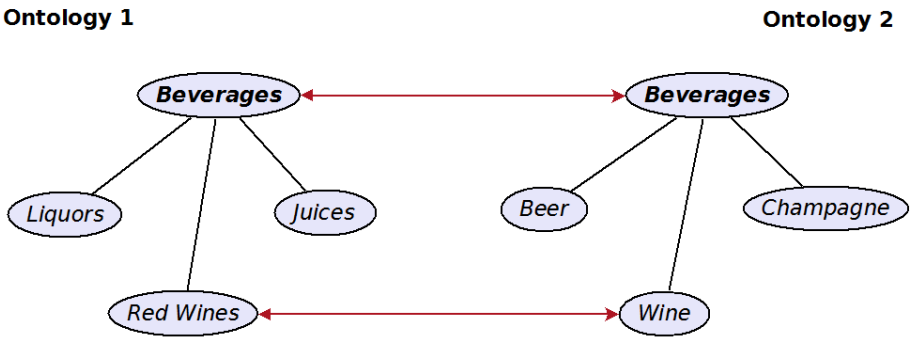

Figure 1.2: Merge example showing a possible mapping between two ontologies.

If the relation type of a correspondence is provided, it is also called a *semantic correspondence*. Determining semantic correspondences is thus called *semantic matching* and the outcome is a *semantic mapping*. Knowing the relation type of correspondences yields several advantages. First of all, such a more precise mapping allows far better results in schema and ontology merging [118, 121]. As an example, consider the two ontologies in Fig. 1.2 that need to be merged. A match tool has discovered the two correspondences *(Beverages, Beverages)* and *(Red Wines, Wines)*, yet no correspondence type is known. The target ontology O_T, which is to be derived from the two input ontologies, would obviously contain a top-level concept *Beverages* and the four sub-concepts *Liquors*, *Juices*, *Beer* and *Champagne*, though an important question remains: Which of the two matching elements *Red Wines* and *Wines* has to be used in the target ontology? If it was assumed that the correspondence is of type `equal` (which is usually the case if no relation type is provided), only one of them should be added to O_T in order to prevent any form of redundancy. Thus, if the relation type `equal` is assumed, a random element has to be chosen, so either *Wine* or *Red Wines*. The ontology merger might take the first element, *Red Wines*, for the target ontology O_T. The *Wine* element will not be part of the merged ontology in this case.

This approach is obviously erroneous, as it leads to an incomplete ontology O_T. *Wine* is a more general concept than *Red Wine*, and in order to cover the full semantic scope of

the two ontologies, the target ontology requires the *Wine* concept and not *Red Wine*. The *Red Wine* concept can be added as a sub-concept of *Wine* so that O_T covers all concepts from the two original ontologies. In any case, the knowledge about the relation type between *Red Wines* and *Wine* would successfully prevent the illustrated pitfall. Fig. 1.3 demonstrates the two possible merge results: the erroneous one (left) and the correct one (right).

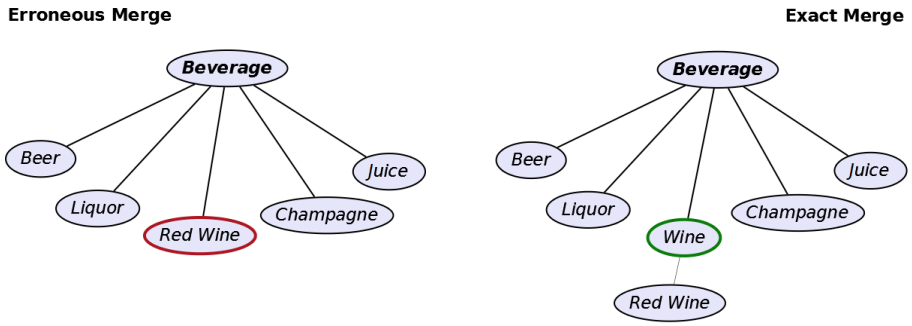

Figure 1.3: Juxtaposition of imprecise merge and exact merge.

Relation types are also crucial for mapping adaptation. Given two ontologies O_1, O_2 and a mapping M in between, mapping adaptation becomes necessary if at least one of the two ontologies changes. This process is referred to as *Ontology Evolution*. It occurs especially in the biomedical domain, where new concepts are frequently added or existing ones are deleted, updated or rearranged [63]. A simple example is depicted in Fig. 1.4. The concept *Cameras* in the original ontology version O has been replaced by the two more specific concepts *DSLR cameras* and *Compact cameras* in O'. Though such a split-operation can be detected by match and mapping adaptation tools like GOMMA,[1] knowledge about the semantic type is very crucial for some operators as the *split* and *merge* operator [69]. For example, knowing that *compact cameras* and *DSLR cameras* are specifications of *cameras* indicates that O' has become more specific and that instances once connected to *cameras* are now found in either of the two new concepts in O'. By contrast, if the two relations were of type `equal`, the mapping tool would recognize that the two new concepts are only synonyms and represent the same entities.

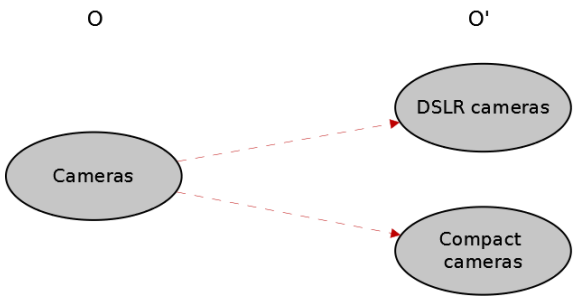

Figure 1.4: Classic example of ontology evolution (split-operation).

[1] `http://dbs.uni-leipzig.de/de/gomma`

Taxonomies are simplifications of ontologies used for scientific classifications, product catalogs, knowledge management and the like. An `inverse is-a` relation in taxonomies indicates that there is no suitable concept in the target taxonomy for a given source concept, and that the target taxonomy has to be manually adapted before data transformation from source to target taxonomy can commence. An example is illustrated in Fig. 1.5, in which it is assumed that data from a source Ontology 1 is integrated into a target Ontology 2. Each element in the source ontology is related to a concept in the target ontology. Without any knowledge about the relation types, instances of the three source concepts would be directly transformed to the three target concepts. However, the `inverse is-a` relation between *Wine* and *Red Wine* indicates an obstacle in this scenario. The *Wine* concept of the source ontology is more general than the *Red Wine* concept of the target ontology. It might contain other sorts of wine that would be transformed to the target ontology, which would become semantically incorrect. In this case, the relation type `inverse is-a` suggests the user to adjust the target ontology before data is transformed. For example, the *Red Wine* concept could be extended to *Wine*, or further concepts like *White Wine* and *Rosé* could be added to Ontology 2. The same argument holds for `has-a` relations (which is the inverse of a `part-of` relation). For instance, a correspondence *(doors, door handles)* leads to the same situation as in the *Red Wine* example.

The problem of matching ontologies covering different scopes is omnipresent and can occur just as well if the relation type is available. It is a problem related to data transformation and always requires manual interaction. However, relation types can indicate that there is such a different coverage, and in which part of the mapping it exactly appears. It would be much more difficult to handle if the semantic relation types were not provided.

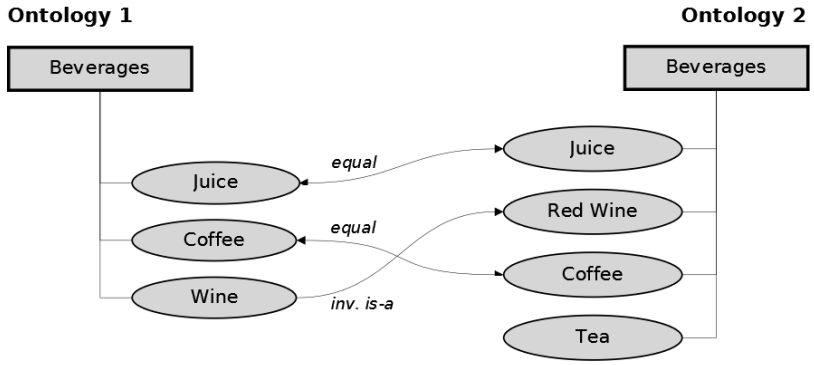

Figure 1.5: Example for ontology integration.

Among the numerous developed match approaches and match tools, there is only a small number of tools that can determine the semantic relation type of correspondences so far, e.g., [57, 122, 78, 38]. These approaches primarily use WordNet and selected lexicographic strategies to detect correspondences between ontologies and their relation type. Analyzing the quality of such tools, it became obvious that approaches calculating correspondences and relation type at the same time do not achieve completely satisfying results in real-world mapping scenarios. There are already some decent and useful tools to determine semantic correspondences, but as their main focus is not on the type detection,

but on the detection of the actual correspondence, they do not exhaust all possible methods for a reliable, more sophisticated determination of semantic correspondences. The frequently used WordNet and some lexicographic strategies can support simpler mappings, but in more complex scenarios, e.g., the mapping of two product catalogs, such techniques usually do not suffice. It was thus an important goal of this thesis to devise advanced strategies and apply more comprehensive background knowledge to augment the mapping quality compared to state-of-the-art tools. Besides, the focus was much stronger on linguistic knowledge, as it is the most essential aspect for the type determination between two given concepts.

In this thesis, the main focus of research comprises the annotation of given schema and ontology mappings with the semantic relation type. This procedure is called *mapping enrichment*, as a semantically richer mapping is the outcome of the approach. We propose a so-called two-step approach, which first uses a classic schema or ontology matching tool to calculate an initial mapping and secondly determines the relation type of the correspondences. Thus, the initial mapping is the input of the mapping enrichment approach and an enriched mapping is the output, which provides for any correspondence a specific relation type (see Fig. 1.6). This makes the approach very flexible, as different tools can be used in the first step.

There are different generic techniques that are exploited to determine the correspondence type, most of them being strongly related to linguistic insights. Such an approach comes with different difficulties, e.g., determining the relation type in hierarchical ontologies or telling the difference between `is-a` and `part-of`. Background knowledge plays an important part in this doctoral work as well, including the proper usage of dictionaries and thesauri and the development of new, more comprehensive resources. These different techniques are used to determine the six relation types `equal`, `is-a`, `inverse is-a`, `part-of`, `has-a` and `related`, which are described in more detail in Chapter 3.

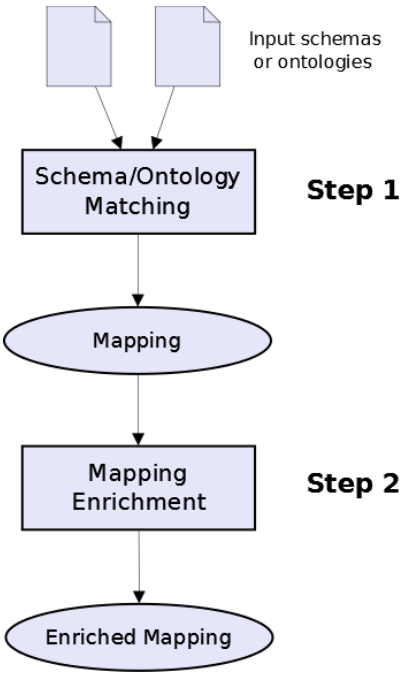

Figure 1.6: Basic notion of the two-step approach for mapping enrichment.

While the determination of the relation type between two words is already difficult, e.g., between the terms *student* and *academic institution*, it is even more difficult to determine the relation type in real-world correspondences, which may consist of itemizations like *"chocolate, candies and sweets"*. Sometimes, also the structure and hierarchy of an ontology has a considerable influence on the relation type, and occasionally it is even difficult for a user to decide on the correct correspondence type. These are important challenges that have to be addressed in the context of this research as well.

The research focus will be on the field of schema
and ontology matching and on related areas such as integration and merging. However, the techniques developed in the context of this research are rather generic, as they mostly decide how two concepts or sets of concepts are related to each other. Such techniques can also be used in areas like entity resolution and instance data analysis, e.g., to tell whether one set of instances is more or less general than another set of instances. They can also be used in text mining and text analysis, e.g., to compare relations between titles and headings or to determine the subject or domain to which a text refers. The background knowledge that was developed as part of this research can be of great help in this instance. Eventually, the scientific insights gained from this work can be also very useful for schema matching and ontology matching as such, and could be the basis for an intensive future work.

1.2 Scientific Contributions

In this thesis, the following contributions are made:

1. The need and usage of semantic mappings are discussed and illustrated.
2. Existing approaches for semantic matching are introduced and compared with each other. Current shortcomings and open issues are discussed, as well as solutions to solve them.
3. As a main contribution, several generic techniques are presented to determine the relation type of the correspondences within a pre-calculated mapping. These techniques also regard special structural aspects, like itemizations and the semantics across ontology hierarchies.
4. The automatic acquisition of background knowledge is a second major research issue within this thesis. It comprises the automatic extraction of semantic relations from a web resource (Wikipedia) and its integration in a repository, together with further background knowledge resources like WordNet.
5. The usage of background knowledge in the field of semantic matching is discussed, and in particular the relation type determination and confidence calculation of indirect (complex) relation paths.
6. Based on the developed techniques, two prototypes were implemented: STROMA, which annotates the relation type to the correspondences of a given input mapping and SemRep, a semantic repository containing relations from different background knowledge resources, including the extracted Wikipedia relations.
7. The developed techniques and SemRep are evaluated on real-world scenarios, and benefits and open issues are discussed. Additionally, novel techniques to evaluate both match quality and relation type correctness in a mapping are introduced and applied to different gold standards.

Most of these contributions and results have been already published as peer-reviewed conference papers and journal articles. First approaches for semantic mapping enrichment and preliminary insights were presented in [4] at the *25. GI-Workshop Grundlagen von Datenbanken* in May 2013. Subsequently, a full overview of the approach including a first evaluation was presented at the *ADBIS 2013* conference in Genoa, Italy [5]. An extended version of this approach was published in the *Data and Knowledge Engineering Journal* in 2014, which comprised additional strategies for semantic mapping enrichment and a more profound evaluation. The evaluation also included a comparison between STROMA, S-Match and TaxoMap [6]. In June 2014, a novel approach for extracting semantic concept relations from Wikipedia definition sentences was presented at the *4th Intl. Conference for Web Intelligence, Mining and Semantics* (WIMS) in Thessaloniki, Greece [7]. An extended version was published in the *International Journal on Artificial Intelligence Tools* (IJAIT) in April 2015. It contains various insights of the Wikipedia relation extraction and different propositions for quality improvement. The effect of the extracted relations on real-world mappings has also been evaluated [8]. In March 2015, the semantic repository SemRep has been presented at the *BTW 2015* conference in Hamburg. This publication discussed the design and implementation of a semantic repository, allowing the integration of several lexicographic resources to foster matching and mapping enrichment [9].

1.3 Outline

The first part gives an introduction into the research field of this thesis.

In **Chapter 2**, we give an introduction to the field of schema and ontology matching and data integration in general. In particular, we will define essential terms and techniques which are used throughout the rest of this thesis.

In **Chapter 3**, we give an introduction to linguistics, and in particular in the two subdisciplines *morphology* and *semantics*. These two disciplines are the basis for many strategies and techniques presented in this work.

In **Chapter 4**, we discuss related work. We will introduce related approaches for mapping enrichment or relation type determination and outline shortcomings and difficulties that are addressed in this thesis. Additionally, we will present and discuss background knowledge approaches and background knowledge resources, as they constitute a key role in this work.

The second part describes the mapping enrichment tool STROMA, which annotates the relation type to each correspondence of an input mapping.

In **Chapter 5**, we introduce the system STROMA and describe its basic architecture. We will also elucidate the workflow for mapping enrichment from its inception (a raw input mapping) to its final end (a fully enriched mapping).

In **Chapter 6**, we will present the different strategies to determine the relation type of a given correspondence. In addition to linguistic strategies, structural and partly heuristic strategies are used to find and annotate the correct relation type.

In **Chapter 7**, we will evaluate STROMA. We will demonstrate the effectiveness of STROMA on several manually created perfect mappings (gold standards) and compare STROMA with TaxoMap and S-Match. Additionally, we will evaluate the effectiveness of the different strategies as well as the time complexity of STROMA.

The third part describes the background knowledge repository SemRep, which combines linguistic relations from different resources. Besides already existing resources like WordNet, a new resource was automatically generated by parsing the definitions of some million Wikipedia articles.

In **Chapter 8**, we present a novel approach to extract semantic concept relations from Wikipedia articles. The approach is based on NLP-techniques (sentence parsing) and yields some millions of useful relations that are later integrated in the SemRep repository.

In **Chapter 9**, we introduce and describe SemRep, a semantic repository for lexicographic resources like WordNet and the previously extracted Wikipedia relations. SemRep is primarily designed for the ontology mapping domain and addresses different questions related to this area, e.g., how to keep the query execution time low or how to determine the semantic relation type in indirect paths.

In **Chapter 10**, we present and discuss several methods to augment the overall quality of SemRep, which is to some extent impaired by the automatic extraction of irrelevant and erroneous relations from Wikipedia.

In **Chapter 11**, we will evaluate both the Wikipedia relation extraction approach and the SemRep system. For the SemRep evaluation, we will use the gold standards as in the STROMA evaluation and show how SemRep can boost the effectiveness of STROMA compared to a non-background knowledge approach.

The fourth part is the conclusion of the thesis.

In **Chapter 12**, we give a summary of this doctoral work and highlight important outcomes and insights.

In **Chapter 13**, we discuss several possibilities for further improvement and adaptation of STROMA and SemRep.

The fifth part is the appendix of this thesis and contains an overview of the default configurations of STROMA and SemRep.

2

Basics of Schema and Ontology Matching

Schema and ontology matching are a crucial, inevitable part of data integration. In this chapter, we will first give a general introduction to data integration (Section 2.1) and will discuss the substantial steps that are usually carried out. In the remains of this chapter, we will illustrate some important sub-disciplines of data integration which are the primary focus of this work. In Section 2.2, we will shortly introduce the different metadata structures that are found in database and information systems. In Section 2.3, we will discuss mappings between those data structures and the different kinds of correspondences they can contain. Eventually, in Section 2.4 we will illustrate the different match techniques used to automatically generate mappings.

2.1 Data Integration

Data integration is a significant field in computer science that comprises different techniques to attain a complete, unique, correct and non-redundant view on data from different, heterogeneous data sources [101]. It becomes necessary because of the heterogeneity between at least two databases. There are different types of heterogeneity that can occur if two data sources are to be integrated, among them:

- Heterogeneity between data models, e.g., XML schemas and relational schemas, or relational schemas and ontologies.

- Structural heterogeneity, e.g., the heterogeneity between the simple element *name* and the three elements *title*, *first name*, *last name*.
- Semantic heterogeneity, which includes synonymy between elements, like *employees / staff*, homonyms like *mouse (animal) / mouse (device)* or different units like *Fahrenheit / Kelvin*.

Data integration is a process that usually consists of different steps, depending on how it is actually carried out. Schema resp. ontology matching is the first step. Given two database schemas resp. ontologies as input, a schema resp. ontology matcher calculates a mapping between the two metadata structures, containing the corresponding elements resp. concepts. There are two forms of schema integration, which are also illustrated in Fig. 2.1:

- **Top-down approach:** Given a global schema or ontology G, source schemas or ontologies $S_1, S_2, ..., S_n$ are matched against G.
- **Bottom-up approach:** Given schemas or ontologies $S_1, S_2, ..., S_n$, an integrated (global) schema G is derived, which must cover all information contained in $S_1, ..., S_n$. This approach is also called schema (resp. ontology) integration or merging, as $S_1, ..., S_n$ have to be matched with each other first and subsequently merge techniques have to be applied to obtain S_G.

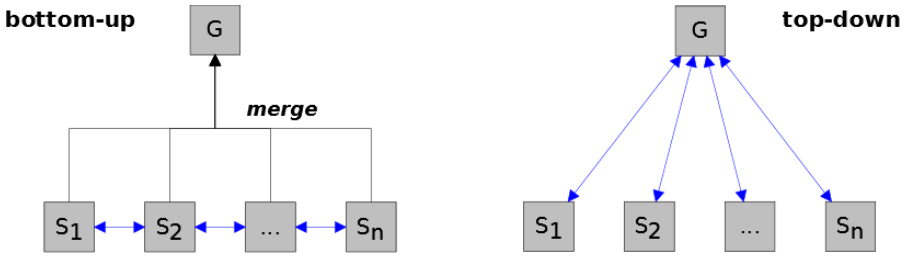

Figure 2.1: The two forms of schema integration.

The mappings can be created by a user or automatically by a program, a so-called schema or ontology matcher. If such a program is used, a user has to check the mapping for correctness and completeness and possibly refine it. If a bottom-up approach is followed, the integrated schema G has to be derived from the source schemas $S_1, ..., S_n$. This step is unnecessary if a top-down approach is followed, since the target schema is already available in this case.

When the target schema is available, transformation rules are formulated based on the previously created mapping. These rules describe how data objects from the source systems are translated into the target system. Such rules can be automatically derived from the mapping, e.g., by commercial tools like BizTalk[2], Stylus Studio[3] and Altova MapForce[4], or by research prototypes like Clio [117], HePToX [24] and Spicy [94]. There are

[2] http://www.microsoft.com/en-us/server-cloud/products/biztalk/
[3] http://www.stylusstudio.com/
[4] http://www.altova.com/mapforce.html

different standards to express such transformation rules, among them XQuery, XSLT and SQL for relational databases. The previously obtained mapping is the inevitable prerequisite to develop such a transformation script, which constitutes a link between the metadata level and instance level.

In the last major step, data objects are transformed to the target system based on the previously developed transformation script. The transformation of data objects can be either on demand (virtual data integration, as in a federated system) or persistently (physical data integration, as in data warehouses). A key aspect in data transformation (also known as data translation) is *data cleansing* where data objects need to be adjusted to match the integrity constraints of the target system [120]. For instance, if meteorological data from different countries is integrated and the target system uses temperature measurements in degrees Kelvin, units like Fahrenheit or Celsius used in the source systems have to be converted into degrees Kelvin. Additionally, values may be stored in the form *value + unit*, e.g., "11.4 m/s" while the target system may only store the values without their unit. In this case, strings have to be adequately converted to match the specific criteria of the target system.

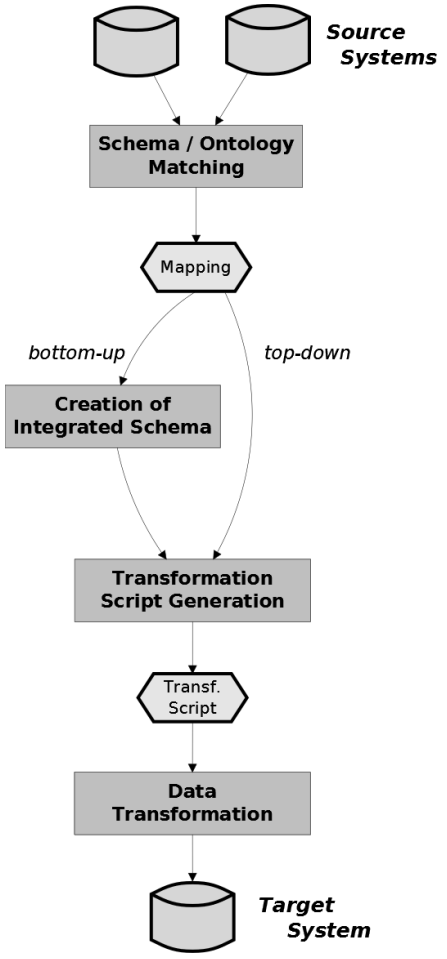

Figure 2.2: High-level illustration of a typical data integration workflow.

Data cleansing is a very complex workflow. An important sub-step of data cleansing is the discovery and elimination of all duplicate objects to avoid any form of redundancy and inconsistency. This field is referred to as *object matching*, *record linkage*, *entity resolution* or *duplicate detection* [26, 45]. For example, if customer data from different databases is merged, such approaches can help to discover that the two objects *{Peter Shall, Oxford Road, 2345 London}* and *{Shall, Peter, Oxford Rd., 2345, London}* refer to the same real-world entity and are thus duplicates. In addition to duplicate detection, data fusion plays an important part, because duplicate data objects may comprise different pieces of information and, potentially, even contradicting information. For instance, there might be two data objects referring to Peter Shall, but they might store two different telephone numbers about this person (it may be quite possible that one of the two numbers is not used anymore, i.e., it is invalid). In such intrinsic cases, manual effort is usually required to solve conflicts of this kind and to warrant a maximum degree of data quality, but entity matching can highly support this process.

Fig. 2.2 illustrates at a high level the steps carried out in a typical data integration process as described above. In this thesis, we will only focus on the first step in this workflow, i.e., on the matching phase and the enrichment of mappings.

2.2 Database Schemas and Ontologies

To represent and classify data within an information system, different models exist to describe the structure of data, which are called *data models*. For example, in the relational data model a schema consists of relations $R_1, R_2, ..., R_n$ and a list of relation attributes (columns) $R_i(A_1, A_2, ..., A_m)$ for each relation R_i [101]. Hierarchical databases, e.g., XML databases, store data sets in a tree-like schema [27]. They consist of a set of nodes N and a set of relations R, with $R \subseteq N \times N$ so that for all $(n_1, n_2) \in R$ holds n_1 is the superior element of n_2. The relation type of relations $r \in R$ is typically `part-of`.

An ontology defines the vocabulary for a specific domain [64] and can be formally described as a triple (C, R, a), with C being a set of concepts, R being a set of relations between concepts ($R \subseteq C \times C$) and a being a set of concept attributes that provide some specific information about a given concept. Such a concept attribute is a triple $(a_{concept}, a_{name}, a_{value})$ and typical concept attributes are *name*, *synonyms*, *definition*, *example*, etc. A relation $r \in R$ is a triple $(c_{source}, r_{type}, c_{target})$. Since ontologies can also represent instances (sometimes called individuals), an improved definition of ontologies seems reasonable which defines them as a 4-tuple (C, R, I, a) with I being a set of instances so that $\forall i \in I : \exists c \in C$ so that it holds: i is an instance of c.

In contrast to hierarchical databases, relations can be of any imaginable type, like *is-a*, *lives-in* or *is-used-for*, and even though the majority of ontologies are directed, acyclic graphs (DAGs), an ontology can also contain cycles. These features, together with the different possible concept attributes, make ontologies semantically richer than database schemas.

A *taxonomy* is a hierarchical, non-cyclic ontology used to classify data objects. Typical taxonomies are, for instance, product catalogs, biological classifications (for species, plants, diseases, genes, etc.) and thesauri (classification of words). Though they are similar to hierarchical databases, they are specific forms of ontologies, in which only the ontology relation type *subclass-of* (`is-a`) is used and no cycles are allowed.

Such data structures, whether relational schemas, taxonomies or ontologies, are also called *metadata structure*, in contrast to the *instances* they represent. Presently, metadata structures are most often strictly divided into two categories: database schemas (including relational schemas and hierarchical schemas) and ontologies (including taxonomies). If metadata structures are matched against each other to find the corresponding elements, it is thus called *schema matching* in case of schemas and *ontology matching* or *ontology alignment* in case of ontologies. Since databases have been used for much longer than ontologies (at least in the field of computer science), a vast amount of research has been dedicated to schema matching. Only within the last 10 or 15 years, research about ontol-

ogy matching has emerged and became solidly entrenched in information integration. It is now being conducted in parallel to schema matching research.

At times, there appears to be a hint of competition between researchers of either field, although the core challenge is generally the same: to find the matching concept pairs (or schema elements) between two metadata structures. In this instance, both research fields are very similar and seize, to a large degree, on the same techniques and scientific insights. Many ontology matching tools can also process schemas (e.g., XML files), while typical schema matchers like COMA can also process ontologies. In this case, it is even possible to match an ontology (e.g., an OWL file) with a schema file (e.g., a CSV or XML file), which clearly demonstrates that a strict segregation of the two fields is neither sensible nor necessary [76]. In [16], the authors use the term *schema* in a broader sense, which subsumes both database schemas and ontologies. This consideration corroborates that from a user's viewpoint there is no actual difference between schemas, taxonomies, ontologies or related models that are to be matched.

Thus, in the further course of this doctoral thesis, we will consistently refer to the field of *ontology matching*, but the techniques and notions presented throughout this work will also hold for schema matching. Likewise, if we use the term *ontology*, we also refer to schemas and thus regard all types of metadata structures. We will also use the term *concept* instead of *(schema) element*, though again, we treat both terms equally, unless there is a good reason to actually distinguish between the two terms resp. research fields.

In the following, we assume that an ontology relation type t is either `is-a` or `part-of`. Let c, c' be two concepts of an ontology. If there is a relation $r(c, t, c')$ then c' is called the parent concept of c and c is called the child concept of c'. If a concept c does not have any parent concept, it is called a *root* concept. Normally, there is at least one such root concept in an ontology. In taxonomies, there is exactly one root concept. If c has no outgoing relations, so that there is no $c' \in C$ for which holds c' is the child node of c, c is called a *leaf* concept. Concepts that are neither root nor leaf concepts are called *inner concepts*.

I denote a path from a root concept r to a leaf concept l as follows: $r.i_1.i_2. \ ... \ .i_n.l$, where $i_1, i_2, ..., i_n$ are inner nodes. For the representation of such a path, we will take the concept names and replace spaces by an underscore for better legibility, e.g., *Person.Employees.Accounts_Clerk*.

In most cases, data objects (instances) refer to the leaf concepts, i.e., inner concepts are used for structuring and concept organization, while leaf concepts represent the actual data. For example, if there is a computer taxonomy with a concept *printer* and subconcepts *inkjet printer* and *laser printer*, a specific data object *"Laser Printer X-1000"* would be most likely stored under the concept *laser printer* and not under some superior concept like *printers* or *devices*.

In most ontologies, a concept c has exactly one parent concept c', however, it is generally possible that it has multiple parent concepts $c'_1, ..., c'_m$. This phenomenon is called multiple inheritance and is often found in biological or medical ontologies [27, 134]. It can also appear in everyday taxonomies, as illustrated in Fig. 2.3 a). In this scenario, multiple inheritance occurs because the concept *student* can be part of different other concepts (*high-school*, *college* and *university*). Any form of multiple inheritance can be resolved as

shown in chart b), which leads to a redundant representation of concepts involved in multiple inheritance. This redundancy can be easily prevented by making the concepts more specific, e.g., changing the *student* concepts to *high school student*, *college student* and *university student*.

Multiple inheritance seems to appear less frequently in sole `is-a` hierarchies, though there are also possible examples, especially in so-called copulative compounds (I discuss compounds in more detail in Section 3.1.2), like *Person.Actor.Actor-Manager*, *Person. Manager.Actor-Manager*. Multiple inheritance also occurs if different views on a concept are possible, e.g., *Fruit.Nuts.Strawberries*, *Fruits.Berries.Strawberries* — from a biological point of view, a strawberry is a nut, but a user dealing with a fruit taxonomy might consider a strawberry to be a berry. For this reason, it is not unlikely that the concept *strawberry* might occur several times in the ontology and has several parent elements.

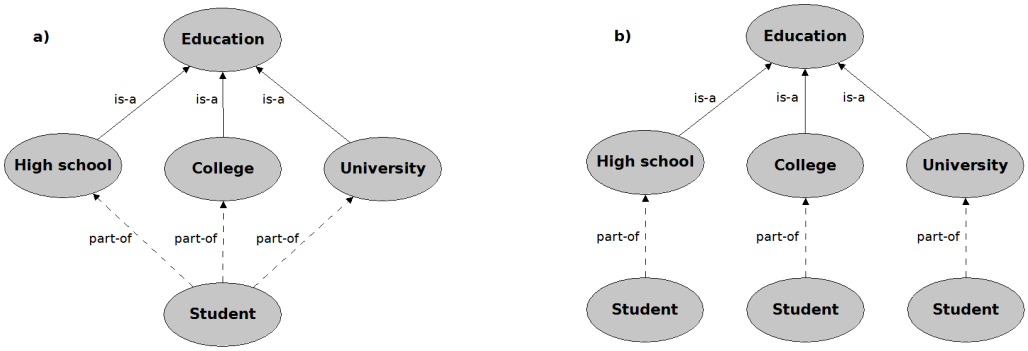

Figure 2.3: Scenario with multiple inheritance (left) and an equivalent case with resolved multiple linkage (right).

STROMA does not regard multiple inheritance, i.e., input mappings must consist of unique paths from root to leaf concepts. However, any multiple inheritance can be transformed to single inheritance as illustrated in Fig. 2.3. Such a transformation leads to a redundancy of concepts, though. In the *student* example, no actual redundancy occurs, because the several student concepts in chart b) are only specifications of the previous student concept in chart a) and there is no semantic overlap among them. However, in the *strawberries* example redundancy cannot be prevented, as the two paths *nuts.strawberries* and *fruits.strawberries* represent the same instances. Resolving multiple inheritance is thus not always an easy undertaking and may require more sophisticated techniques.

2.3 Matching and Mapping

The term *matching* refers to the semi-automatic detection of matching concepts between two input ontologies O_1, O_2. Such tools are then called *ontology matcher*. A few approaches are also able to calculate mappings between multiple (more than two) sources, which is called holistic schema resp. ontology matching [140, 70].

A correspondence is a relation between two matching concepts $c_1 \in O_1$ and $c_2 \in O_2$. In its simplest form, a correspondence is therefore a tuple (c_1, c_2). However, matchers often provide further information about a correspondence. In [133], a correspondence is defined as a 4-tuple (id, c_1, c_2, r), where id is a unique identifier of the correspondence and r is the relation type, like equivalence or subsumption. Correspondences in the mapping tool COMA 3.0 are 3-tuples (c_1, c_2, s), where s specifies the confidence value (score) of a correspondence on a scale from 0 to 1. A value of 1 indicates a true match, while a value of 0 indicates a false match.

In this thesis, a correspondence is defined as a 4-tuple (c_1, c_2, s, r), thus consisting of source and target concepts c_1, c_2, a confidence value s and a relation type r. However, since the confidence value is of subordinate importance in this thesis, we will write examples in a simplified form. If the relation type is known or of any importance, we use the form (c_1, r, c_2). In case that it is unknown or extraneous, we use the form (c_1, c_2). Examples are *(Person.Employees.Accounts_Clerk, `equal`, Staff.Accountant)* and *(Person.Employees.Accounts_Clerk, Staff.Accountant)* respectively. If the entire correspondence path is not necessary for an example or notion, we will only write the leaf concepts, e.g., *(Accounts_Clerk, `equal`, Accountant)*.

Given a concept $c \in O_1$ and two concepts $d, d' \in O_2$ with $d \neq d'$. If there are two relations $(c, r, d), (c, r, d')$, these two atomic correspondences form a complex correspondence, i.e., one concept in O_1 is related to at least two concepts in O_2 [42]. Such complex correspondences are also called one-to-many resp. many-to-one correspondences, in contrast to the one-to-one correspondences where exactly two concepts are in a relationship [40, 140, 75, 123, 155]. Given a concept c and matching concepts $d_1, ..., d_n$, we will denote a complex correspondence as follows: *({c} r {$d_1, ..., d_n$})*.

There are some interesting features about complex correspondences: First, the relation type of relations within a complex correspondence is different from `equal`, if it is assumed that d and d' are not semantically equivalent. This is normally the case, since it would otherwise mean redundancy in one of the two ontologies, though there are a few exceptions like the previously illustrated strawberry example. Still, it can be assumed that the relation type in a complex correspondence is either `is-a` or `part-of`. The following complex correspondence consists of three `part-of`-relations: *({title, first_name, last_name} , `part-of`, {name})*. If data objects are to be transformed into the target system, complex correspondences indicate that a specific transformation function is needed. In the *name* example, this would be the *concatenate*-function that needs to combine the values of *title*, *first name* and *last name* in order to obtain the *name* object required by the target ontology. In ontology mapping, $(1:1)$, $(n:1)$ and $(1:n)$ correspondences usually occur, while $(m:n)$ correspondences are very rare [119].

If two ontologies to be matched differ considerably in the number of concepts they contain, but roughly describe the same scope, one ontology can be assumed to be more specific, while the other is more general (top-level ontology). In such mapping scenarios, a high number of complex correspondences can occur, because a general concept from the top-level ontology would match several more-specific concepts of the opposite ontology, e.g., *({table, chair, wardrobe, shelf}, `is-a`, {furniture})*. Consequently, `is-a` and `part-of` relations normally predominate in such a mapping scenario.

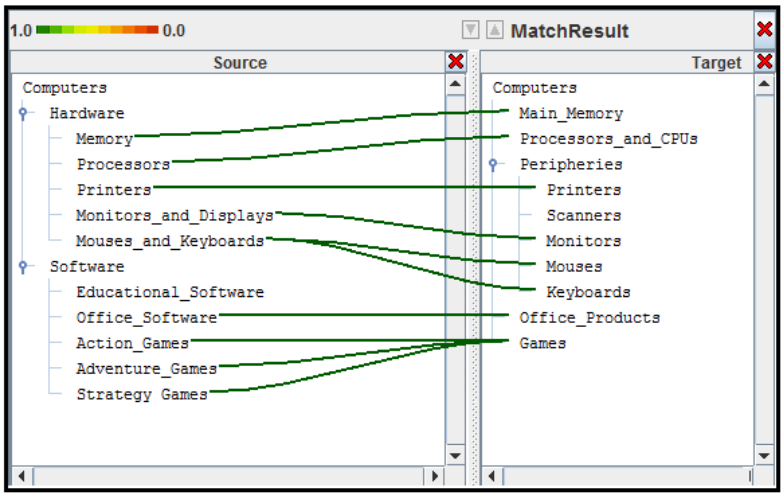

Figure 2.4: A sample mapping created with COMA 3.0.

A *mapping*, also called *alignment*, is a set of correspondences. It can be calculated by a match tool or manually created by some human expert. If it was determined by a match tool, it is also called the *match result*. Automatically calculated mappings may be incomplete and may contain erroneous correspondences, since automatic solutions are normally unable to achieve a 100 % correct mapping due to the practically insurmountable barriers of heterogeneity [53]. A correct mapping verified by a human expert is often called a "perfect" mapping, or a *benchmark*, *gold standard* or *reference alignment*. Such gold standards are frequently used to evaluate the match quality of match tools. Fig. 2.4 shows a sample mapping performed with COMA 3.0. It is a verified, perfect mapping, because it contains all relevant relations between the two computer taxonomies and no incorrect or irrelevant relations. It contains both simple and complex correspondences and correspondences of different types, e.g., equivalence for *(Printers, Printers)* and subsumption (`is-a`) for *(Action_Games, Games)*.

Given a source concept $s \in O_1$, most match tools aim to find the most relevant target concept $t \in O_2$. However, some approaches also provide so-called *top-k* mappings. For each s, they provide the k most relevant target nodes in O_2, with $k \geq 1$. The likelihood that the correct target concept t is among the top k target concepts $t_1, ..., t_k$ is much higher compared to a match tool only providing one match partner for s. However, this method inevitably requires the user to decide for each ontology concept which of the k provided correspondences is the correct one [52], i.e., user interaction can become more demanding than in top-1 approaches. This technique can also be used in the context of evaluations, e.g., in [40] the authors compare the match quality between a top-1 configuration and a top-3 configuration of their approach.

Since the outcome of any match process is a mapping, match tools are also called mapping tools, e.g., schema mapper or ontology mapper. This indicates, however, that the term *mapping tool* also refers to tools that do not calculate mappings automatically, but only provide an interface for a user to develop a mapping manually. Examples include proprietary software solutions like Microsoft BizTalk, Stylus Studio or Altova MapForce.

To avoid misunderstandings, we will prefer the term *match tool* against *mapping tool* in this thesis.

Lately, new approaches for mapping generation were introduced, which suggest the creation of correspondences by a crowd of volunteers instead of a single person. This advanced form of manual mapping development distributes the generally high effort on several persons so that manual mapping development becomes less tedious and time-expensive for each contributing person. However, motivating a crowd of volunteers to perform schema or ontology mapping, the absence of expert knowledge among volunteers, contradicting results among workers and unreliable users (e.g., users creating correspondences randomly) are serious issues connected with this approach [96, 106, 128].

2.4 Ontology Matching Techniques

A broad range of techniques can be used to automatically determine a mapping between two input ontologies O_1, O_2. Given the two concept sets $C_1 \in O_1$ and $C_2 \in O_2$, classic match tools compare each concept pair $(c_1 \in C_1, c_2 \in C_2)$ and for each pair determine the match likelihood (confidence). Some tools use *blocking* before the matching phase in order to reduce the number of comparisons [15].

Different classification of strategies were proposed, e.g., in [119] and [132]. Strategies can be basically distinguished by the input they receive (schema-based or instance-based) and by their granularity (element-based or structure-based). Element-based strategies only analyze a specific concept, while structure-based strategies also consider neighbor concepts like the parent concept and siblings. In this work, we will not regard instance-based techniques, but will solely refer to the schema-based techniques.

Lexicographic strategies or string-based strategies compare the concept name between two concepts and suggest a match if a certain lexicographic overlap is reached [35]. For instance, lexicographic matchers assume that two concepts *(telephone number, telephone no.)* are semantically related, because of their high lexicographic overlap. Many algorithms were proposed to calculate the lexicographic similarity between two strings, like *Levenshtein* (an implementation of *Edit Distance*), the *Jaccard* distance, *n-Gram* (e.g., Trigram) and *tf-idf* [66, 132].

Strategies that also use linguistic insights and techniques, like word stemming, prefix-matching or compounding (see Chapter 3) are also called *linguistic strategies*, though there seems to be no clear demarcation between the two terms in related literature. *External strategies* use additional resources to determine the relatedness between concept pairs. Such resources can be dictionaries, synonym tables, thesauri or background knowledge ontologies and knowledge bases. Basically, those are non-generic strategies, because they depend on a finite set of information and may not invariably return any result (e.g., if a concept name is not contained in the source). Generic strategies, however, will always return a result (a similarity value between 0 and 1).

In addition to structural, instance-based and string-based strategies, more complex strategies can be used like *reasoning* or *probabilistic strategies* as implemented in ap-

proaches like [98, 138, 81, 56]. Such approaches normally calculate an initial mapping using the aforementioned strategies and subsequently refine the mapping by means of a higher order logic, Bayesian networks, machine learning, etc. There is practically no approach that would work completely without string-based techniques.

Most match tools use multiple strategies to calculate mappings between metadata structures. The match tool COMA, which stands for *combined matching*, was among the first tools that combined different techniques to enhance the mapping quality achieved by single strategies [41]. In this context, *workflow management* plays an important part that takes care in which order strategies are carried out, and how their outcomes are combined and further processed. Some approaches use a set of pre-defined workflows or manual workflow management, while also self-configuring approaches exist that try to determine the perfect workflow for two given input ontologies automatically [108].

3 Basics of Linguistics

Most techniques for semantic enrichment of ontology mappings are based upon linguistic insights. The knowledge about the structure and formation of words is an important prerequisite for lexicographic strategies and refers to the sub-discipline *morphology*. The different relations between words and their meaning refers to the sub-discipline of *semantics*. Since the focus of ontology mapping is solely on the written language, and in particular on single words or phrases, other traditional fields of linguistics are irrelevant in the context of this thesis, namely, phonetics and phonology (the study of sounds and sound systems), syntax (the study of the structure of sentences) and pragmatics (the study of language in use). In this chapter, a brief introduction to two basic linguistic fields is given and their relevance for schema and ontology mapping is outlined. We will first give an introduction to morphology (Section 3.1) and then to semantics (Section 3.2).

3.1 Morphology

3.1.1 Morphemes

Morphology is the study of the internal structure of words and the formation of new words within a language [10, 28, 112]. A *morpheme* is the smallest meaningful unit of a language. Each word consists of at least one morpheme. For example, the word *unthinkable* consists of the three morphemes {un} (meaning the opposite of something), {think} and {able} (meaning a conversion from the verb *to think* to and adjective). The word *cars* consists of the morpheme {car} and the plural morpheme {s}. By contrast, the word *pertain* consists of only one morpheme: {pertain}. Though there are other words that start

with *per-* (e.g., *permanent*, *peruse*, *perception*), as well as words that end with *-tain* (e.g., *contain*, *sustain*, *maintain*), these units are not considered morphemes, as neither *per* nor *tain* have any obvious meaning, which is an absolutely essential property of morphemes. Words that consist of exactly one morpheme are called *simplex words*; words consisting of more than one morpheme are called *complex words*.

In some specific cases, words consist of both a morpheme and a construct that does not possess any obvious meaning. An example is *cobwebs*, where *web* and *s* are apparently morphemes, yet *cob* has no meaning and is practically unattested in English. Such morphemes are called *cranberry morphemes*, while *cranberry* contains such a specific morpheme itself. Further examples include: twilight, hinterland or lukewarm (cranberry morphemes are underlined).

There are two general types of morphemes: free morphemes, that can stand alone, and bound morphemes, that can only co-occur with other morphemes. In *unthinkable*, the morphemes {un} and {able} are bound[5] while {think} is a free morpheme. Bound morphemes can be further sub-divided in prefix morphemes and suffix morphemes. A prefix morpheme occurs before a free morpheme while a suffix morpheme occurs after it. In the above example, {un} is a prefix morpheme while {able} is a suffix morpheme.

In addition to this classification, bound morphemes can also be divided w.r.t. their meaning: An inflectional morpheme changes the grammatical function of the word where it appears, but does not change its general meaning. A derivational morpheme does not change the grammatical function, but the meaning of the word. Inflectional morphemes like {ed} (past tense form), {ing} (progressive form), {s} (plural form) or {est} (superlative form) are invariably suffix morphemes. Derivational morphemes can be both prefix and suffix morphemes (bound morphemes are underlined):

- **Derivational prefix morphemes:** unhappy, disregard, nonchalantly, interact.
- **Derivational suffix morphemes:** politeness, modernize, eatable, teacher.

In most cases, derivational suffix morphemes change the word class of the given word. They are of less importance for this thesis, because in schema and ontology mapping nouns are the prevailing elements (while all other word classes are very rare). Comparisons between nouns and other word classes are thus unnecessary and will be disregarded. Analogously, inflectional morphemes are also ignored in this thesis and the focus remains on derivational prefix morphemes.

Derivational prefix morphemes generally do not change the word class, but indicate an (at times even considerable) change in semantics, as in *connection* and *disconnection* or *employment* and *unemployment*. There are a few exceptions though, like the morpheme {en} which transforms a word to a verb (as in *enlarge*, *enact*, *enable*, etc.) or {dis} and {de} (as in *discourage* and *derail*). Fig. 3.1 depicts an overall classification of the morphemes and some examples for each type.

[5] Note that *able* is a special morpheme which can be both free (as in the word *able*) and bound, as in *unthinkable*, *deliverable*, etc.

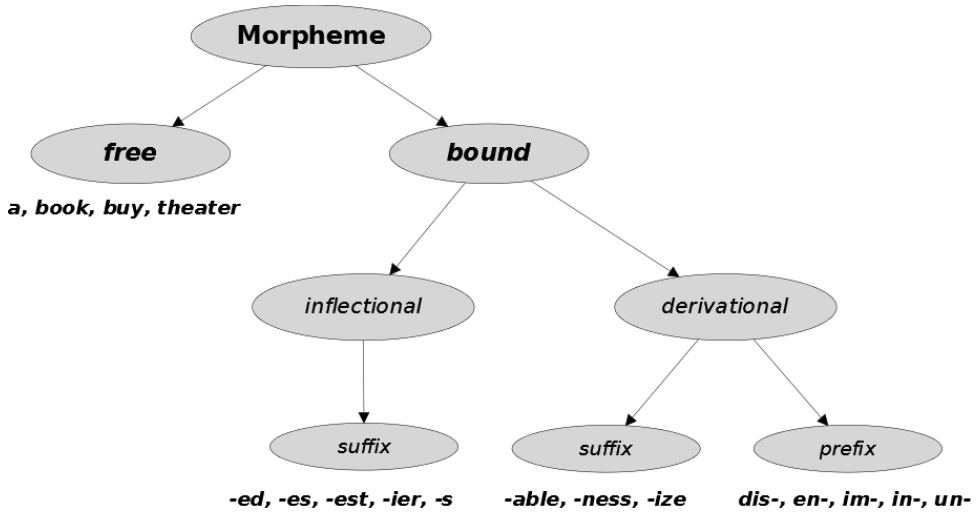

Figure 3.1: Classification of morphemes.

In the context of this research, derivational prefix morphemes play an important role, because they indicate unrelatedness between two words which are very similar from a lexicographic point of view. For example, the two words *unemployment* and *employment* have a very high similarity. The Jaccard metric is a frequently used match strategy to determine the similarity s between two words w_1, w_2 [91]. It holds:

$$s = \frac{|w_1 \cap w_2|}{|w_1 \cup w_2|} \tag{3.1}$$

In this case, $|w_1 \cap w_2|$ describes the number of overlapping letters or sequences of letters (so-called n-grams) and $|w_1 \cup w_2|$ describes the number of letters or n-grams occurring in both strings. If letters are regarded ($n = 1$), it holds:

$$s = \frac{\{e, m, p, l, o, y, m, e, n, t\}}{\{u, n, e, m, p, l, o, y, m, e, n, t\}} = \frac{10}{12} = 0.83 \tag{3.2}$$

This is a very high similarity and most tools would consider such a similarity as a hint for a correspondence between *unemployment* and *employment*, though the two concepts express directly opposite things.

3.1.2 Word Formation

Over the centuries, the evolution of the languages led constantly to new terms for different objects, processes, properties, abstract things, etc. While there are a couple of root words in each language, most of them of unknown origin, the majority of words is derived from the existing vocabulary. Such a derivation implies that there must be at least some semantic relatedness between the original word and its descendant, and this is a very important insight for linguistic-based strategies in schema and ontology matching.

The creation of new words is called *word formation* and several forms of word formation exist [13].

Derivations

In derivations, a prefix p or suffix s is added to a given word W, so that the derivation is either pW or Ws. The prefix and suffix is a derivational morpheme, just as illustrated in the previous section. Derivations are among the most frequently occurring word formations, as they can change the meaning of a given word (prefix morpheme) or allow its usage in a different word class (suffix morpheme).

Compounds

A compound is a special word C that is a combination ("compound") of two words H (called *head*) and m (called *modifier*) [129]. The head word generally occurs at the end of the compound and specifies its basic meaning; the modifier occurs at the beginning and modifies the compound so that C expresses something more specific than H. For example, a *database conference* is a specific *conference* and a *kitchen chair* is a specific *chair*. Together with derivations, compounds are the most productive means of word formation, especially since compounds can also be derived from existing compounds, like *kitchen chair manufacturer*, which is a specific *chair manufacturer*. In such compounds of higher order, the compound consists of more than one modifier, though it still consists of only one head. In this thesis, a compound C is defined as $C = m_1 m_2 ... m_n H$, thus consisting of n modifiers ($n \geq 1$) and one head.

From a lexicographic point of view, three types of compounds can be distinguished:

- **Closed compound:** Head and modifier are directly combined, e.g., *cookbook* or *blackboard*.
- **Hyphenated compound:** Head and modifier are separated by a hyphen, e.g., *get-together* or *see-saw*.
- **Open compound:** Head and modifier are separated by a space, e.g., *city hall* or *computer screen*.

Though there is no official regulation in the English language how to create and write compounds (in fact, some compounds can take up different forms, like *bus-driver* and *bus driver*), there are some characteristics for each type. Generally, most recently evolved compounds like *web space* or *smart phone* use the open form. The same holds for most compounds where either head or modifier consists of two syllables or more, like *building site* or *railroad company*. Otherwise, more established and more frequently occurring compounds often use the closed form, as *airbag* or *blackboard*. The hyphenated form sometimes marks a development from the open form towards the closed form (e.g., *data base – data-base – database*).

In the English language, compound words are normally combined without any further characters and only the head is modified w.r.t. grammatical inflections. From a technical point of view, this regularity makes it relatively easy to parse and process English compounds, while it is more difficult in other languages. For instance, in German compounds the modifier can change, as in *Städtebund* (*Stadt* + *Bund*) and additional characters may occur between modifier and head, as in *Handelsabkommen* (*Handel* + *Abkommen*). Only in some rare cases, an English compound modifier has changed so that it is no official word of the language anymore, e.g., *holiday* (*holy* + *day*).

From a semantic point of view, three types of compounds can be distinguished [88]:

- **Endocentric compounds:** It holds that C is a specification of H, as in the example *blackboard* (which is a specific board). This is the classic form of compounds.
- **Exocentric compounds:** There is no (obvious) semantic relation between C and H, such as in *buttercup* (which is not a cup and has no other semantic relation to a cup).
- **Copulative or appositional compounds:** Head and modifier are at the same level and C is not a specification of H, but rather the sum of what m and H express. An example is *bitter-sweet*, which means both bitter and sweet (not a specific sweet).

Exocentric compounds are often of literal meaning, like *computer mouse*, which resembles a mouse, but has no semantic relation to an actual mouse. They can also be the result of words that changed their spelling, e.g., butterfly, which might originate from "flutter by", or cocktail, which might originate from French *coquetier*.[6] Eventually, they can be the result of two words that often co-occur, like *pickpocket* (someone who picks pockets) or *breakfast* (the time to break the fast after night) [14].

Copulative and appositional compounds are quite rare. Unlike endocentric compounds, they express something more general than the compound head. They are very often hyphenated compounds, as in *Bosnia-Herzegovina*, *actor-director* or *twenty-one*.

Shortenings, Blends and Acronyms

Shortenings (or clippings) are reductions of a word by deleting parts of its base. There are three forms of shortenings, according to which part of the original word remains.[7]

1. **Beginning:** *lab / laboratory*, *doc / doctor*
2. **End:** *net / Internet*, *phone / telephone*
3. **Middle:** *flu / influenza*, *fridge / refrigerator*

In some cases like *fridge*, the spelling of the shortening changes slightly. The first case is the most frequently occurring case in the English language [112].

[6] `http://www.etymonline.com/`

[7] `http://www.oxforddictionaries.com/words/shortenings`

Acronyms are abbreviations that follow the regular reading rules of a language, like *NATO* or *radar* (**Ra**dio **D**etection **A**nd **Ra**nging). Blends consist of two words w_1, w_2 where parts of at least one word are deleted. The remnants of w_1, w_2 are combined to a new word, very similar to a closed compound. Typical blends are *motel* (*motor* + *hotel*) and *brunch* (*breakfast* + *lunch*).

Conversion

Conversion refers to processes where a word of a specific word class is used for a different word class. For example, *professional* was originally solely used as an adjective, but converted towards a noun (*a professional*). Conversions do not influence the spelling of a given word.

Loan Words

Loan words are words that were imported from a different language. In many cases, the spelling was adapted to match the regulations of the English language, which makes it quite impossible to distinguish between a loan word from any other word by means of automatic approaches. Only in some rare cases a loan word was not adapted to the English language, like *kindergarten*, or only to some degree, like *iceberg* (German: Eisberg).

3.1.3 Arbitrariness of Language

Given a free morpheme (simplex word), the arbitrariness of language suggests that there is practically no correlation between its representation (spelling or pronunciation) and its meaning [10, 54, 112]. This implies that the meaning of a morpheme cannot be predicted by its representation. For example, nothing in the simplex word *tree* indicates any relation to the actual concept tree, nor does the construct `tree` resembles any tree. The different words used in other languages corroborate this theory, like *Baum* in German or *arbre* in French refer to the same object, but are completely differently spelled and pronounced. Therefore, simplex words appear to be completely arbitrary and only complex words carry semantics, like *blackboard*, which indicates a compound between the two arbitrary words *black* and *board*, or *politely*, which indicates the adverb form of the arbitrary word *polite*.

The knowledge about this arbitrariness can help in ontology enhancement, especially in the discovery of false matches. Many match algorithms assume that concepts are related if they have a notable overlap in spelling and most lexicographic strategies like Edit Distance, Trigram or Jaccard depend on this assumption. Yet the arbitrariness of languages suggests that there is generally no relation between two similarly spelled simplex words, as there is no semantic relations between *cable*, *fable*, *gable*, *stable*, or *table*, though they are similarly spelled and pronounced. Words that are in a semantic relations, like *chair* and *seat* or *house* and *building* have normally quite different representations and cannot be determined by lexicographic strategies alone. Only if at least one word consists of more

than one morpheme, a similar spelling can hint to some relatedness, though it does not has to be equality, as in *(cookbook, book)* or *(database, data record)*.

In the field of ontology matching, classic lexicographic strategies can discover the following equivalence relations:

1. Relations between words that have different spellings, like *color* and *colour*.
2. Relations between words with different inflections, like *computer*, *computers* and *computing*.
3. Relations between shortenings or abbreviations like *lab* and *laboratory*.
4. Relations between words that have a high lexicographic overlap and are indeed related, like *hotel* and *hostel* (which are rather rare, though).

They can also discover typos (like *employee – employe*), but such spelling errors usually do not appear in well-developed schemas or ontologies. In case 1 and 2, differences never appear at the beginning of the two words, but only in the middle or at the end. Also, most shortenings and abbreviations start with the same letter or sequence of letters compared to their counterpart (case 3). It can thus be assumed that if two words are very similar in spelling but do not start with the same letter, they are probably mismatches, just as in the sample correspondence *(stable, table)*. In such a case, the existence of such correspondences could be seriously doubted and possibly removed to increase the mapping quality.

However, if two concepts start with the same sequence of characters, no unique answer to the question of relatedness can be given, and the correspondence could be either correct (as in the four examples above) or wrong (as in *furniture* and *furnace*). However, examining the two words in more detail, would lead to the conclusion that the correspondence is rather a mismatch. In this case, the difference between *furniture* and *furnace* is too large for case 1 (different spelling). Since neither of the two concepts ends with any English inflection (case 2) and is obviously no shortening of the opposite concept (case 3), it appears very likely that the correspondence is false. However, as in case 4, this is an assumption that does not generally hold and there is a chance that the two words are ultimately related. Classic lexicographic strategies can also discover `is-a` relations like *(high-school, school)* and `part-of` relations like *(bed, bedroom)*, but they are unable to determine the semantic type (and may simply assume that all relations are of type `equal`).

The arbitrariness of language is a significant aspect that is very often ignored in schema and ontology matching. However, this linguistic law only refers to morphemes, not to words in general. For this reason, arbitrariness of language does not mean that lexicographic and linguistic match strategies are generally inconvenient or futile. It only means that the mere spelling comparison of two words is too simple to reliably decide whether they are in any relation or not. Knowing how words and morphemes are put together, and how to interpret the meaning of such combinations, can highly foster matching and semantic mapping enrichment.

Word formation	Relevance for OM?	Mapping Example and counterexample	Remark
Derivations	Yes	section ↔ intersection	Can help to discover false matches (antonyms) resp. pseudo compounds.
Compounds	Yes	bike ↔ mountain bike butterfly ↮ fly	Can help to discover `is-a` relations.
Shortenings	Partly	lab ↔ laboratory lab ↮ label	Can help in some cases to discover `equal` relations, but can also be misleading.
Acronyms, Abbrev.	Yes	EU ↔ European Union EU ↮ Essen University	Can help to discover `equal` relations.
Blends	No	motel ↔ hotel motel ↮ cartel	Blends cannot be unequivocally determined.
Conversion	No	(professional (adj.) ↔ professional (noun))	Word classes different from nouns are irrelevant for mappings.
Loan words	No	(doppelganger ↔ lookalike)	Loan words have often no relevant lexicographic overlap (require handling by dictionary).

Table 3.1: Overview of the different forms of word formation and their relevance for ontology mapping.

3.1.4 Relevance for Ontology Mapping

Knowledge about the structure of words and the procedure of word formation helps to decide whether two ontology concepts are likely to be a match or not, and what kind of relation type may hold. The important thing about the different word formations is a lexicographic overlap between two matching concepts, because generic match strategies primarily analyze the lexicographic similarity between two concepts. Table 3.1 provides an overview of the different word formations and their relevance for ontology mapping. In Chapter 6, the gain of these linguistic insights will be demonstrated in detail, since several generic strategies for relation type detection and correspondence verification are based upon them.

Derivations indicate a change of the meaning. If a matcher found a correspondence between *section* and *intersection*, because of the large lexicographic overlap, this knowledge can help to recognize an invalid match and thus to remove the correspondence. Counterexamples are rare, though in some cases a derivation may be semantically rather close to the original form, as in *relatedness* and *interrelatedness* or *presentation* and *representation*. Directional prefixes like fore- (as in *forehead*) or back- (as in *backyard*) tend to express part-of relations and have to be specially treated. **Compound** words can help to indicate `is-a` relations, but exocentric compounds lead to false conclusions, as they indicate no specific relatedness. However, they are relatively rare.

Shortenings can help to discover `equal` relations (or to confirm the correctness of a given correspondence in the first place). However, they can also lead to many false conclusions, since many other terms may start with the same sequence of characters. Some shortenings like *telly* (for television / TV) cannot even be discovered with simple methods that use "starts-with"-expressions. **Acronyms and abbreviations** whose constituents match the initials of a complex expression can indicate an `equal` relation (as in EU and European Union). Since many abbreviations have different meanings, false conclusions cannot always be prevented. **Blends** are difficult to handle. First, it is nearly unfeasible to discover a blend in a correspondence without any background knowledge. Second, it is also unfeasible to discover the two words that take part in such a blend. Since blends are very rare, their relevance for schema and ontology mapping is very low.

Conversions are irrelevant for ontology mapping, as most schema elements and ontology concepts are expressed by nouns. **Loan words** are also of low importance, since they are derived from a different language and thus have generally no lexicographic overlap with an English counterpart or related term. For example, lexicographic strategies would be unable to determine the semantic relation between *doppelganger* (German origin) and *lookalike* (English origin), since their lexicographic overlap is too low.

In this doctoral thesis, we will make use of compounds for semantic enrichment and derivations, which help to discover pseudo compounds (words that look like compounds, but are no compounds). Additionally, acronyms are used to discover `equal` relations between open compounds and simple words. All other word formations are ineligible for mapping enrichment and are thus not further regarded.

3.2 Semantics

3.2.1 Concepts

Semantics is the study of the meaning of words and sentences. Words, and in this particular context nouns, are used to refer to real-world objects or to abstract things. As there is an infinite number of real or imaginary objects a word could refer to, concepts are the central part of the field of semantics and serve as an abstraction for such objects.

A *concept* is a mental category that is defined by a list of properties and represents real or imaginary objects. For example, the concept AUTOMOBILE is a mental category that could be described by properties like "is a vehicle", "has a motor", "has wheels", "transports passengers". Any object that matches the definition is then represented by this concept. In this context, words (nouns) refer to one or more concepts. For instance, the words *car*, *automobile* or *motorcar* refer to the concept AUTOMOBILE, meaning that they can be used for all objects that are represented by this concept. The word *car* refers to different concepts, for instance AUTOMOBILE, TRAIN CAR and TROLLEY CAR. Fig. 3.2 summarizes the relations between words, concepts and objects.

The relation between word and concept is also called *meaning*, while the relation between word and object is called *reference*. Objects represented by a concept are called *instantiations* of the concept.

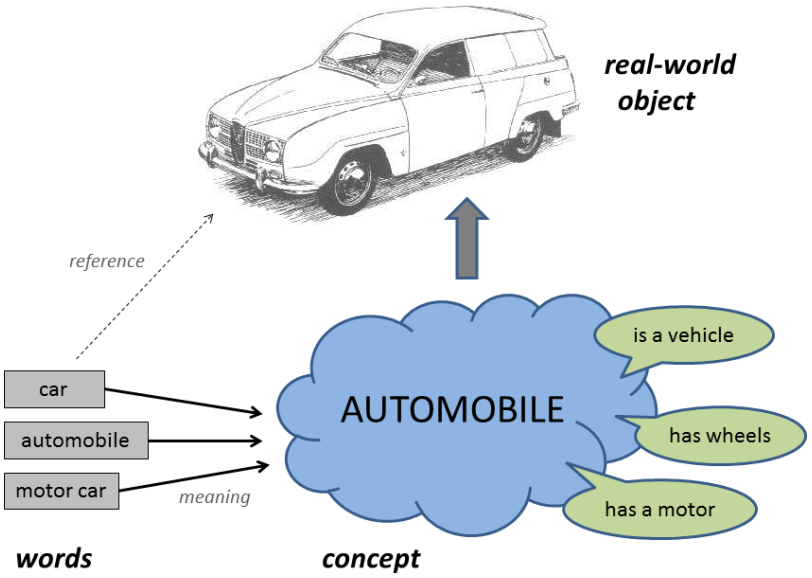

Figure 3.2: The basic notion of concepts: A concept (blue) is a mental category defined by properties (green).

In many cases, concepts are defined by at least one of the following aspects:

1. The next hypernym (e.g., *car is a vehicle*).
2. Where it occurs (e.g., *car is part of traffic*).
3. What it consists of (e.g., *car has wheels*).

The first point organizes the concept in the overall concept hierarchy. It is practically used whenever a concept is defined, e.g., in a dictionary or encyclopedia. The second point indicates a `part-of` relation to another concept, while the third point indicates a `has-a` relation to another concept. There are many other ways to define a concept, e.g., by its usage (*a lawnmower is used to cut grass*) or general properties (*a vehicle transports passengers or cargo*). However, the three enumerated points are crucial for this doctoral work, because they put the concept at hand in an `is-a`, `has-a` or `part-of` relation. Analyzing such definitions like *"A car is a vehicle that has wheels and a motor"* is an excellent way to extract semantic relations from definition texts and thus to collect new semantic relations in a thesaurus, which we will elaborate in Chapter 8.

Two words w_1, w_2 that refer to the same concept are called *synonyms*. If a word w refers to more than one concept, it is called a *homonym*. Synonyms and homonyms are key elements in the field of semantics and in data integration. Synonyms indicate an equivalence

relation, as in the correspondence *(car, automobile)*, while homonyms usually indicate a mismatch. Since two homonyms have the exact same physical representation (spelling), they highly impair the quality of ontology matchers, because most match tools assume an equivalence relation in case that two terms are equally spelled, e.g., mouse (animal) and mouse (device). If w_1, w_2 refer to concepts that express opposite things, they are called *opposites* or *antonyms*.

3.2.2 The Hierarchy of Concepts

As illustrated in Fig. 3.3, concepts are often arranged hierarchically. Given a concept c, a more-specific concept c' that shares all properties of c and defines some additional (more restrictive) properties is called a hyponym of c; c is then called the hypernym of c'. For instance, the concept VEHICLE is a rather general concept, possibly defined by the properties "movable object" and "transports passengers or cargo". An AUTOMOBILE is a specification of this concept (hyponym), with additional properties like "has wheels", "has a motor". Again, CONVERTIBLE is a specification of AUTOMOBILE, with the additional property "has an open roof".

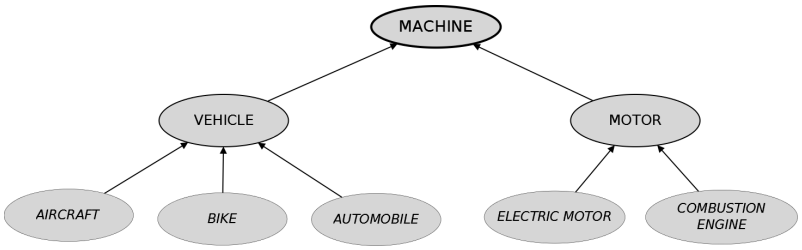

Figure 3.3: Sample hierarchy of 8 concepts.

Thus, a hyponym always depends on its hypernyms and inherits all of their properties. This principle is very much related to the set-subset relation in mathematics or the class-subclass relation in object-oriented programming languages.

Two concepts c_1, c_2 that have a common hypernym concept c are called co-hyponyms. For example, *apple*, *banana* and *orange* are co-hyponyms of the concept *fruit*. Thus, co-hyponyms share some properties (the properties of their hypernyms), but also differ in some specific properties. In linguistics, co-hyponyms are generally handled as opposites, i.e., those concepts have different meanings. However, co-hyponyms are often quite related to each other and are also relevant for the field of ontology mapping.

3.2.3 Meronyms and Holonyms

Concepts that are usually part of another concept are called *meronyms* [99]. For example, *arm* is part of *body*, which makes *arm* a meronym of *body*. *Body* is called the *holonym* and consequently is in a `has-a` relation with *arm*. Thus, meronyms resp. holonyms express

typical `part-of` resp. `has-a` relations, though they may not be invariably true. For instance, the semantic relation <*cellar*, `part-of`, *house*> is a typical meronym relation, but there are houses without a cellar and cellars not being part of a house (like a storm cellar or wine cellar). Thus, unlike hyponym and hypernym relations that are always true `is-a` relations, meronym and holonym relations only hold to some degree. Sometimes, the meronym has a stronger dependency on the holonym, e.g., <*foot*, `part-of`, *leg*>, while sometimes this dependency is less intense, e.g., screen `part-of` computer (many computers are used without a screen). In fact, such `part-of` resp. `has-a` relations may have different dependencies, depending on the perspective of the viewer. In the example *tree – forest*, the relation <*tree*, `part-of`, *forest*> is relatively loose, as there are many trees not being part of a forest. By contrast, the opposite direction <*forest*, `has-a`, *trees*> (or consists of trees) is a very strict relation and holds generally. This leads to 4 different cases:

1. Both meronym and holonym are in a loose relation. This refers to the example <*cellar*, `part-of`, *house*> (there are houses without a cellar and cellars not being part of a house).
2. The holonym depends on the meronym, but not vice versa. This case refers to the example <*tree*, `part-of`, *forest*>, in which the meronym (tree) can also exist independently from the holonym (forest).
3. The meronym depends on the holonym, but not vice versa. An example is <*oasis*, `part-of`, *desert*> (an oasis can only appear in a desert, but not all deserts may have an oasis).
4. Both meronym and holonym depend on each other. This case is relatively rare. Examples might be <*husband/wife*, `part-of`, *marriage*> or <*capital*, `part-of`, *nation*>.

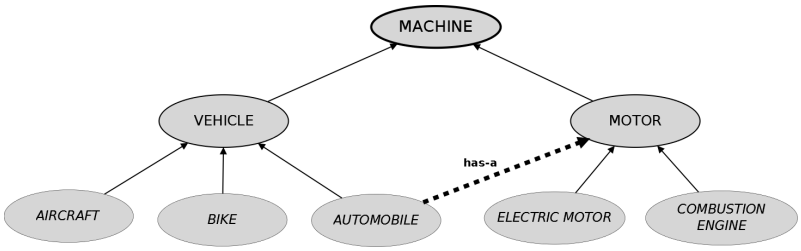

Figure 3.4: Sample hierarchy of concepts, extended by a has-a relation.

This itemization shows that only one of the four cases refers to `part-of` resp. `has-a` relations that are invariably true. Therefore, such relations have to be treated more carefully, as they often tend to become rather unspecific and less universally valid.

In the hierarchy of concepts, meronym resp. holonym relations take up a very special role, because they typically represent relations between two different branches. The relation <*car*, `has-a`, *motor*> leads to a link between the *vehicle* branch and *motor* branch (see Fig. 3.4, which extends Fig. 3.3). This means that there is no longer any evidence for a noticeable property overlap between the two concepts. Both concepts in such a relation could differ highly in semantics, even though they often co-occur. For example, there

is not much similarity between a *desk* (piece of furniture) and an *office* (room within a building), except that most offices have desks.

3.2.4 Relation Types

Relation types used in ontology matching are equivalent to the linguistic relations discussed above. However, different terminologies are used in scientific literature, and relation types are generally not precisely defined. Hyponym relations are called *is-a* [78], *kind-of* [58, 29], *subsumption* [38, 138] or *more/less-general* [57, 122], while synonym relations are called *equivalence* [57, 38, 138], *is equal* [122] or *same-as* [29, 78]. In OWL, the terms *subClassOf* and *sameAs* are used to describe hyponym or synonym relations between concepts.[8] In [132], the authors partly use the notation of description logic to express semantic relations. They mention equivalence ($=$), more-general ($\sqsupseteq$), overlapping ($\sqcap$) and disjointness or incompatibility ($\perp$).

In [30], the authors use the terms *part-of* and *contains* for meronyms and holonyms, while *associates* is used to describe relations similar to co-hyponyms [29]. Although *part-of* and *contains* describe inverse relations, this approach applies different confidence values to these types. This means, that a correspondence *(house,* contains, *room)* is differently treated compared to the inverse *(room,* part-of, *house)*.

Linguistic relation	**Example**	**Set theory (mathematics)**	**UML/OOP terminology**	**Ontology mapping**
Synonymy	river, stream	Equivalence		equal, equivalence, same as
Antonymy	valley, mountain	Disjoint		mismatch, disjoint
Hyponymy	apple, fruit	Subset-of	Specialization, Subsumption	is-a, kind-of, more-specific, less-general, subsumption
Hypernymy	vehicle, car	Inclusion, Superset-of	Generalization	inverse is-a, less-specific, more-general
Meronymy	roof, building		Aggregation, Composition	part-of
Holonymy	body, leg			has-a
Co-hyponymy	oak, maple, birch		(Association)	(related-to)

Table 3.2: Typical linguistic and semantic relations.

[8] `http://www.w3.org/TR/owl-guide/`

Co-hyponym relations are called *overlap* [57], *is close* [122] or simply *related*, as in this thesis. The term *related* is rather unspecific and can also mean *"any kind of relation"* in some approaches. Thus, the concepts *sun* and *sunburn* can be considered to be *related*, as the *sun* causes sunburns (yet there is no actual co-hyponym relationship between the two concepts). In this thesis, we treat the type `related` as a synonym to co-hyponym, thus meaning that two concepts share some properties but also differ in some properties (which is equivalent to the mathematical relation *overlap*).

Most tools simply ignore `part-of` and `has-a` relations, while some approaches like [58] comprise both hyponym and meronym relations under the term *subsumption* (which is semantically incorrect).

Table 3.2 provides a summary of the different terms used for semantic relations. First, the official linguistic expression and an example are provided. Further on, the table illustrates the respective terms used in set theory (in a mathematical sense), in UML or in object-oriented languages, as well as in ontology mapping.

Each of the linguistic relations is transitive, except for antonym relations. Synonym, co-hyponym and antonym relations are symmetric, while all other relations are asymmetric. For hyponym and meronym relations the most specific relation is normally desired. For instance, a correspondence <*thumb*, `part-of`, *body*> is not false, but the most specific and most relevant correspondence would be <*thumb*, `part-of`, *hand*>.

Since the focus of the thesis is on ontology mapping, the terminology from this domain is used, namely, `equal` for synonym relations, `is-a` for hyponym relations, `inverse is-a` for hypernym relations, `part-of` for meronym relations, `has-a` for holonym relations and `related` for co-hyponym relations. Antonym relations will not be regarded.

At times, it seems very difficult to determine the correct relation type between matching concepts even for humans. Considering a relation between *street* and *road*, what would be the semantic type? Are the two terms equivalent, or is street a specific form of road, or a road a specific form of street? WordNet, for instance, suggests that street is a specific form of road, but in some mapping benchmarks created by a crowd of volunteers, the majority of users decided on an equivalence relation [110]. Further on, it seems sometimes difficult to distinguish between `part-of` and `is-a`. Are legs extremities, or are they part of (the) extremities? Is a CPU a form of hardware, or is it part of (the) hardware? Though dictionaries usually provide clear answers, users might have different points of view in some instances (and may not always agree with those definitions).

4
Related Work

Much research has been dedicated to Schema and Ontology matching [119, 132] and a broad assortment of tools and approaches are the outcome of this research. The majority of present match tools cannot determine the relation type of correspondences, e.g., GLUE [43], Sambo [85], Falcon-AO [77], AgreementMaker [36], Sobom [154], CIDER [60], MaasMatch [130], or Yam++ [103]. Some tools use WordNet for matching, which means that the specification of a relation type would be actually possible [85, 36, 130, 103], but this semantic knowledge is not further used in the respective match tools. Most approaches only discover simple correspondences, with a few approaches focusing specifically on complex correspondences, e.g., [153, 40, 75]. Complex correspondences indicate invariably that a relation type different from `equal` occurs, but none of the approaches regards relation types as such.

Only a few approaches are dedicated to semantic matching so far. In the following chapter, we will first introduce and outline those approaches that are able to determine the relation type of correspondences (Section 4.1). Secondly, we will discuss the different forms of background knowledge that can be applied in semantic schema and ontology matching, and how background knowledge resources can be automatically generated (Section 4.2). Finally, we will discuss the field of mapping enrichment, in which an existing mapping is post-processed for semantic enrichment (Section 4.3).

4.1 Relation Type Determination

4.1.1 Match Tools

In contrast to the numerous schema and ontology matching tools that have been developed within the last decades, only few approaches are able to determine the actual relation type that holds between two matching concepts.

The open source framework **S-Match**[9] is among the most notable of those tools [57, 58, 56]. It is a re-implementation of the former matching tool CTX-Match [131] and distinguishes between *equivalence*, *more/less general*, *overlapping* and *mismatch* relations, thus covering synonymy, hypernymy, co-hyponymy and antonymy. Given two input ontologies O_1, O_2, S-Match provides for any concept pair (c_1, c_2) ($c_1 \in O_1, c_2 \in O_2$) one of the 5 relation types, while the most relevant relation between any concept pair will be selected for the alignment. The relation types have a different binding strength: Equivalence is preferred over more/less general, which is preferred over mismatch. The authors acknowledge that overlap relations are difficult to handle and are presently ignored by the system.

S-Match uses a library of lexicographic (syntactic) matchers like Trigram, or Edit Distance, as well as background knowledge (WordNet) and reasoning to derive the relation type for each concept pair. The node concepts are represented by a logical propositional language that also regards the internal position of concepts in the hierarchy of the ontology. For instance, a concept *cars and motorbikes* under the super concept *BMW* is represented as a formula like $(cars \sqcup motorbikes) \sqcap BMW$, referring to cars and motorbikes manufactured by BMW. For each concept pair, the most relevant relation type is finally calculated and verified by a satisfiability solver. A formula (resp. relation type) is valid if, and only if, its negation is unsatisfiable.

S-Match not only uses the linguistic relations between synsets to derive semantic matches, but also compares the WordNet glosses of the two synsets in which two concepts appear. Such WordNet glosses provide a short textual description (definition) of each synset. If two glosses share a certain amount of words, S-Match regards the two terms to be equivalent. Therefore, S-Match is able to find matching concepts that are not connected by a direct WordNet link [58].

Though S-Match is a very sophisticated tool, a tendency to very bulky mappings could be observed in experiments , with nodes in O_1 being related to a large number of nodes in O_2 by more/less-general relations (see Section 7.7). This clearly diminishes the overall usefulness of the approach, since users are normally interested in the most relevant correspondences.

TaxoMap[10] is another freely available ontology matcher that determines different relation types [122, 68]. It distinguishes between *isEq*, *moreGnl*, *lessGnl* and *isClose*, which refer to the elementary relation types equivalence, subsumption and relatedness. Unlike

[9] `http://semanticmatching.org/s-match.html`

[10] `https://www.lri.fr/~hamdi/TaxoMap/TaxoMap.html`

S-Match, TaxoMap draws more strongly on WordNet and lexicographic strategies, using them as the primary source for correspondence detection. The gist of the approach is to only work with specific parts (sub-trees) of WordNet and not with the full resource. For instance, if two food ontologies are matched, only the food branch of WordNet would be used, which has to be automatically discovered and extracted before the matching can commence. Using just a specific branch of WordNet, the authors can effectively avoid mismatches caused by homonyms of different domains, like *orange*, which can be a fruit or a color. The lexicographic strategy *label inclusion* suggests a subsumption relation if words appearing in a concept label c_1 also appear as part of a concept label c_2. This strategy can be seen as a simplified version of the compound approach that is used in STROMA (see Section 6.1). In later years, TaxoMap was improved by several advanced strategies, e.g., by reasoning on the similarity values and structures [67].

TaxoMap is one of the very few semantic match tools that has ever been evaluated w.r.t. the relation type [122]. The authors could achieve a very good precision in each relation type (*more/less general* and *isClose* have been evaluated), but were facing a rather low recall. In detail, the match tool achieved a precision of 83 % on `is-a` relations and 82 % on relations of type `related`, though the overall recall was only 23 % [122]. This also coincides with the comparing evaluations carried out in the context of this thesis (see Section 7.7).

The freely available ontology matcher **RiMOM**[11] uses multiple strategies like string-based, structure-based, statistical and background knowledge strategies (WordNet) to determine correspondences between ontologies. The tool can distinguish between equivalence and subsumption relations and handles both simple and complex (many-to-many) correspondences [145, 144, 87, 157].

ASMOV is another WordNet-exploiting approach that detects *same-as*, *isa* and *disjoint-from* relations [78]. Apart from the concept labels, also the concept properties, parent nodes and instance data are taken into account. In an improved version, the tool can also exploit the UMLS thesaurus as a background knowledge source [80].

AROMA is among the few tools that specifically intend to discover subsumption relations, though the approach is also able to find equivalence relations. Subsumption relations are discovered by using the so-called association rule model, which was successfully used in market data analysis and data mining in general [1]. Given two items A, B, the association rule model specifies the likelihood for $A \rightarrow B$. For example, if A is *bread* and B is *milk*, the association rule model describes the likelihood that someone buying bread is also buying milk. The authors adapt this approach for linguistics and the rule $A \rightarrow B$ can be interpreted as the likelihood that something describing A is also describing B, e.g., that a concept *vehicle* is also referring to *cars*. Allowing such conclusions, the documents (instances) associated with each concept are analyzed and specific terms are extracted and compared. AROMA is thus a probabilistic approach that tries to find implication relations corroborating subsumption relations between concepts. Additionally, lexicographic matchers are used to enhance the overall match quality. The AROMA ap-

[11] `http://keg.cs.tsinghua.edu.cn/project/RiMOM/`

proach was tested against a gold standard containing only equivalence relations so that subsumption relations could not be verified [37, 38].

LogMap is another well-known match tool which iterates between a discovery phase, in which new correspondences are detected, and a repair phase, in which the correspondences are verified and possibly deleted. Unlike the previous systems, the tool does not focus on relation type determination, however, after the repair phase, some critical correspondences can be "weakened". Weak correspondences are then considered subsumption relations. Although LogMap is a reasoning-based approach, lexicographic strategies and background knowledge (WordNet, UMLS) are used for the initialization phase. [81].

H-Match is a match tool that uses lexicographic, linguistic and structural techniques and context interpretation for ontology matching. For the context interpretation and different structural techniques, H-Match uses the internal semantics between ontology concepts and regards *equivalence*, *subsumption*, *aggregation* and *relatedness*. These types are used to indicate the relation strength between concepts, where equivalence has the highest value and relatedness the lowest. However, the resulting mapping does not contain any semantic knowledge about the correspondences, i.e., correspondences are simple pairs between ontology concepts with a specific confidence score, which does not provide any information about the semantic type [29, 30].

Spiliopoulos et al. present an approach called CSR (Classification-Based Learning of Subsumption Relations) that discovers subsumption relations between two ontologies by means of classification-based machine learning [138]. Different structural patterns within one ontology and between two ontologies are used for the training phase. The approach can subsequently discover subsumption and equivalence relations between the two input ontologies.

In [110], the authors calculate mappings between text mining taxonomies that contain `equal`, `is-a`, `inverse is-a` and `related` correspondences. The different types are determined by a novel approach that combines instance data analysis and meta data analysis. For example, the overlap between instance data is taken into account to determine relations different from `equal`. In [34], the authors present the method SURD (**SU**bsumption **R**elation **D**iscovery) that also analyzes instance data to determine equivalence and subsumption relations. It was among the very few approaches in which subsumption relations have ever been evaluated. The authors could obtain a precision of 73 % and 79 % in a mapping between the UMLS Metathesaurus and the two ontologies GENIA[12] and GRO[13], though the precision could only be approximated by verifying a randomly selected subset of the obtained mapping. They could not determine the recall of their approach, as they did not possess the correct (perfect) mapping between the ontologies.

Even though there is a considerable number of tools that distinguish between different relation types, they are apparently not the main focus of research. With the exception of SURD and TaxoMap, no approach was ever evaluated w.r.t. relation types, but evaluations rather concentrated on the mere correspondence detection between ontologies and comparing these results with other tools [57, 78].

[12] `http://www.medlingmap.org/taxonomy/term/102`

[13] `http://www.ebi.ac.uk/Rebholz-srv/GRO/GRO.html`

The Ontology Alignment Evaluation Initiative (OAEI) is an international initiative for evaluating and comparing ontology match tools by providing standardized match tasks and benchmark suits.[14] In the series of annual OAEI contests, only one match track has ever been dedicated to subsumption relations. The *Oriented Matching*[15] track in the OAEI 2009 contest required the discovery of equivalence, more-general and less-general correspondences. Among the 16 tools that participated in the OAEI contest, only 4 tools (ASMOV, CSR, RiMOM and TaxoMap) joined this specific track [79, 138, 157, 67]. In 2011, the organizers of the OAEI intended to run another *Oriented Matching* track, but eventually were compelled to cancel it, because the number of contributing tools was too little [46]. Thus, even though there are approaches that can deal with different relation types, there is apparently much need for a solid approach that would focus much more on semantic correspondences.

Tool	**Provided Types**	**Primary Techniques**	**Resources**	**First Introduced**
Aroma	equal, is-a, inv. is-a	probabilistic, instance-based		Université Pierre-Mendès-France, Grenoble (2006)
ASMOV	equal, is-a, inv. is-a	linguistic, structural, instance-based	WordNet	INFOTECH Soft Inc., Miami (2007)
CSR	equal, is-a, inv. is-a	structural, mach. learning		University of the Aegean (2010)
LogMap	equal, (is-a), (inv. is-a)	linguistic, reasoning	WordNet, UMLS	University of Oxford (2011)
RIMOM	equal, is-a, inv. is-a	linguistic, structural, probabilistic	WordNet	Tsinghua University, Beijing (2004)
S-Match	equal, is-a, inv. is-a, related	linguistic	WordNet	University of Trento (2004)
SURD	equal, is-a, inv. is-a	instance-based		Nanyang Technological University (2012)
TaxoMap	equal, is-a, inv. is-a, related	linguistic	WordNet	Université Paris-Sud (2007)
STROMA	equal, is-a, inv. is-a, part-of, has-a, related	linguistic	SemRep[16]	Leipzig University (2013)

Table 4.1: Comparison of the different tools providing different correspondence types.

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

[15] `http://oaei.ontologymatching.org/2009/oriented/`

[16] SemRep currently consists of the resources WordNet, UMLS, OpenThesaurus, relations extracted from Wikipedia and (optionally) ConceptNet

4.1.2 Comparison of Approaches

Table 4.1 provides a juxtaposition of the match tools that are able to discover correspondences of different types. For each tool, the provided types, their primarily used techniques for matching as well as their primarily used background knowledge resources are specified. The type *disjoint* (mismatch) was excluded from the table, as it does not indicate any relevant relatedness.

Comparing these tools, there are two obvious insights: first, that tools generally distinguish between equivalence and subsumption relations (and some distinguish relatedness as well), and second, that most tools exploit background knowledge to determine these relations. Some depend even strongly on background knowledge (e.g., TaxoMap), while others use it in addition to further strategies (e.g., LogMap, RiMOM). WordNet is practically the only resource exploited by the tools at hand, while LogMap also uses UMLS (medical domain). This strong dependency to WordNet and UMLS is simply caused by the absence of further sophisticated lexical resources, which we will discuss in more detail in the subsequent section.

So far, there is no tool that distinguishes both between `is-a` and `part-of`, but both relation types are frequently used under the term *subsumption* (which, by definition, only refers to the `is-a` type). In the field of ontology merging or ontology evolution, a distinction between `is-a` and `part-of` does not seem excessively important at first sight, because the two types are rather similar after all. However, there remains an important semantic difference between the two relations which should not be entirely ignored, because `part-of` relations can link two different (independent) concepts. For example, in a food taxonomy the relation <*red wine*, `is-a`, *wine*> indicates that the *redwine* concept can be added as a sub-concept of *wine*, just as shown in the initial example in Chapter 1. The relation <*rice*, `part-of`, *rice pudding*>, however, does not indicate such a possibility. From a semantic point of view, *rice* should never be added as a sub-concept of *rice pudding* (which otherwise would turn the taxonomic order upside down), but the two concepts should reside in completely different areas of the taxonomy (e.g., *rice pudding* under a concept *desserts*, and *rice* under a concept *staple food*).

4.2 Background Knowledge

4.2.1 Introduction

Applying background knowledge is an effective method to bridge the large gap between the formal representation of real-world objects and its actual meaning. Since there are various examples where generic strategies (like lexicographic and structural strategies) reach their limit, background knowledge plays an increasingly crucial role [2] [156] and generating knowledge bases and thesauri is the focus of research for a very long time. After giving a short introduction to the research field of background knowledge exploitation, we will discuss how background knowledge resources can be classified and how they can be distinguished in general (Section 4.2.2). Subsequently, we will discuss re-

sources that have been manually created (Section 4.2.3), resources that have been automatically created (Section 4.2.4) and approaches that make use of the world wide web, which are specialized, dynamic approaches (Section 4.2.5). Eventually, we provide a juxtaposition of the different approaches presented in this section (Section 4.2.6).

Basically, two types of background knowledge sources can be distinguished: linguistic (lexicographic) resources defining and organizing concepts, as well as knowledge bases trying to collect and provide a large body of information about instance data (entities) like persons, locations, organizations, buildings, movies, etc. In many cases, knowledge bases not only link entities to entities (like *Bach lived in Leipzig*), but also entities to values (*Bach died in 1750*) and entities to concepts (*Bach was a composer*). Linguistic resources solely link concepts (like *A composer is a musician*), though there may be some few exceptions (as WordNet also contains a few persons like *Martin Luther* or cities like *Leipzig*).

Different terms are used for linguistic resources. They will be typically called *dictionary* if they are meant to define the vocabulary of a specific domain or language. For this reason, terms defined by the dictionary are normally organized in an alphabetic order. They are called *thesaurus* if terms are semantically organized, i.e., the resource defines a hierarchy of categories (mental concepts) and enumerates terms (synonyms) that refer to those categories. Thus, thesauri primarily provide synonym relations and partly hypernym relations. If the focus is stronger on all linguistic relations (hypernymy, antonymy, meronymy, co-hyponymy), the term *semantic word net* will be often used instead of thesaurus, although there is apparently no clear demarcation between the two terms.

In contrast to thesauri and word nets, knowledge bases can use numerous individual relations types. Knowledge bases about persons may link entities by relations like *lives in*, *was born in*, *is married to*, *is head of*, etc. In the biological domain, relations like *is caused by* (disease), *was discovered by* (pathogen), *prevents* (vaccine or drug) or *interacts with* (protein) are used. In the geographic domain, relations like *is located in*, *is capital of*, *was founded in*, etc. may predominate.

Generally, each resource uses its own internal data structure. Given the complexity of languages, especially multilingual resources may need a rather complex data structure. Recently, the **UBY**[17] approach addressed this problems and provides a standardized framework for lexical resources, intending to cover all linguistic and semantic specialties such resources require [44]. Subsequently, different linguistic resources from two different languages (German and English) were automatically integrated in this framework, among them WordNet, VerbNet[18], FrameNet and linguistic information from Wiktionary and Wikipedia.

Applying background knowledge in ontology mapping not only requires the availability of an appropriate resource, but also strategies to effectively use these resources in a given mapping scenario. Aleksovski et al. were among the first to discuss this problem in detail; they present an approach in which a so-called anchor mapping between source resp. target ontology and background knowledge ontology is first generated and then reasoning is used to determine related concepts [2].

[17] `https://www.ukp.tu-darmstadt.de/data/lexical-resources/uby/`

[18] `http://verbs.colorado.edu/~mpalmer/projects/verbnet.html`

An important challenge in background knowledge exploitation is the handling of homonyms. Though some resources designate homonyms, e.g., by providing different meanings for a homonym concept like *mouse* (an animal, an input device, etc.), match tools are usually unable to determine to which meaning a given concept from the input ontologies refers to. In particular, homonyms can cause misleading indirect paths like *desk* `is-a` *table* `is-a` *data structure*, where the homonym *table* leads to the false conclusion *desk* `is-a` *data structure*. To alleviate such problems, some ontology matching approaches are also devoted to word sense disambiguation [148] and homonym handling [61].

4.2.2 Classification of Resources

Though a distinction between thesauri and knowledge bases appears to be the most reasonable background knowledge classification in the field of schema and ontology matching, background knowledge resources can be classified by far more criteria, among them:

- **Development:** Manual vs. (semi-) automatic
- **Area:** General purpose vs. domain-specific language
- **Data:** Concept data vs. instance/entity data
- **Supported link types:** Generic (linguistic) types vs. domain-specific types
- **Number of Languages:** Monolingual vs. multilingual
- **Size/Extent:** Smaller (incomplete) vs. larger (near-complete)
- **Availability:** Free vs. commercial
- **Access:** Local vs. external (web-based)

A further criteria is the reliability of resources. Manually developed resources like WordNet tend to contain more reliable knowledge (resp. less flaws and inconsistencies) than automatically generated resources, although this is not always easy to evaluate. At times, even linguistic experts may have different opinions about the relatedness between objects, as in the previously introduced strawberry example. Thus, there is no clear answer to this question whether the concept *strawberry* is in an `is-a` relation with the concept *berry* or not, as from a biological point of view it is a nut.

Some criteria can be further divided, e.g., manually generated resources can be created by experts or collaboratively by a community of laymen. Also, some features are interrelated, e.g., a semi-automatically generated resource may be of larger size than a manually created resource, yet may have a lower reliability.

4.2.3 Manually or Collaboratively Generated Resources

The Princeton **WordNet**[19] is one of the most popular and most well-known thesauri [48]. Founded and continuously evolved by George A. Miller in 1986, it is now among the most comprehensive linguistic resources for the English language. In analogy to linguistic semantics, WordNet is based upon mental concepts (synsets) that are linked with each other by hypernym, meronym or antonym relations. Words referring to the same synset are synonyms. The most current version is the Princeton WordNet 3.0, that consists of 155,287 unique words (117,798 nouns) in 117,659 synsets (82,115 noun synsets). To our experiences, WordNet is a very effective resource in the field of ontology mapping, however, the currently available version is from 2006, implying that several modern terms like *cloud computing*, *netbook* or *tablet PC* are not yet contained.

In [84], an approach to effectively use WordNet in schema and ontology matching is presented. Given a word A and a WordNet synset S in which A appears, the authors calculate Super WordNet Synsets (SWS) for A, which are an aggregation of all hypernyms, hyponyms, meronyms and holonyms of S. Given two possible match concepts A, B, the Super WordNet Synsets $SWS(A)$ and $SWS(B)$ are calculated, and according to the number of words those two SWSs have in common, the two terms are either considered as match or mismatch. They can even distinguish between `equal` and `is-a` relations by the number of overlapping words, although the relation type detection was not evaluated.

GermaNet[20] is the German language counterpart to WordNet and has been developed at the University of Tübingen since 1997. Technically, it uses the same data structure as WordNet, but slightly differs in some aspects. For instance, GermaNet only regards meronyms without any subdivision in part meronyms, member meronyms and substance meronyms as in WordNet, and it uses artificial concepts to enhance the taxonomic structure. It also provides some more-specific relation types, like entailment (e.g., *to try – to succeed*) and causality (e.g., *to kill – to die*) [83]. By 2014, GermaNet consists of 93,246 synsets, 110,738 terms and 110,170 relations.

Word nets were developed for many further languages, although the Princeton WordNet remains the most comprehensive and most frequently used resource.[21] A collection of available word nets for different languages is provided by the Open Multilingual Wordnet.[22]

EuroWordNet[23] is a multilingual thesaurus similar to GermanNet or WordNet that combines vocabulary of different European languages, while WordNet had served as a base for the development of this multilingual project. Therefore, the data structure is closely related to WordNet again, but also contains an upper ontology as a semantic framework for the different languages [83]. Altogether, word nets from eight languages (Czech, Dutch, English, Estonian, French, German, Italian, Spanish) have been integrated and are

[19] `http://wordnet.princeton.edu/`
[20] `http://www.sfs.uni-tuebingen.de/GermaNet/`
[21] `http://globalwordnet.org/`
[22] `http://compling.hss.ntu.edu.sg/omw/`
[23] `http://www.illc.uva.nl/EuroWordNet/`

interlinked by the so-called interlingual index (ILI). Integrating such resources entailed the development of equivalence relations between synsets of the specific languages and the WordNet synsets. The project has been finished in 1999. Some sample data is provided free of charge, while the access to the full database requires a commercial license.

The **Universal WordNet** approach is a learning-based approach to automatically develop a word net for more than 200 languages.[24] Starting from the Princeton WordNet 3.0 again, the authors make use of different web dictionaries, Wiktionary, word alignment techniques on web corpora, monolingual thesauri and existing word nets of other languages [39]. Today, the resource contains more than 1.5 million words and can be extended by an additional data set of named entities (MENTA). The generated data sets can be downloaded free of charge.

Crowd sourcing, in which a group of volunteers participates in the creation of linguistic resources, is a promising approach to alleviate the laborious development of such resources. **OpenThesaurus**[25] is an exemplary collaborative project for the German language, which comprises about 93,000 words in 30,000 synsets. Apparently, there is also a considerable number of entities (like German cities, companies or politicians) among the vocabulary. As the contributors are no linguistic experts, the quality seems slightly below resources like GermaNet or WordNet. Another example, though less influential, is **WikiSaurus**[26], a sub-project of Wiktionary providing synonyms, hypernyms, hyponyms and antonyms for selected concepts (while meronyms and holonyms are rare). It currently provides approx. 1,400 categories, though recent activity seems rather low and no API is applicable so far.

OpenOffice and LibreOffice use a list of thesauri in different languages (including German and English), which are included in office programs like Writer or Calc. The thesaurus for the English language is also provided as text file and was automatically derived from WordNet 2.0.[27,28] Investigating the thesauri file that contains about 142,000 concepts, it becomes clear that it does not contain any more information than WordNet 2.0, and that it is thus not a helpful resource if WordNet is already used. For the German version, OpenOffice and LibreOffice use the data from OpenThesaurus, so that no additional knowledge can be obtained from it if OpenThesaurus is already used.

Less known, but still an interesting and beneficial resource is **ConceptNet**.[29] It is a multilinguistic, semantic network similar to WordNet, but contains more specific relations like *X is used for Y*, *X is made of Y* or *X is etymologically derived from Y* [90, 137]. ConceptNet is based upon the Open Mind Common Sense corpus (OMCS), which is a comprehensive repository of declarative sentences created and added by a crowd of volunteers. Such sentences, like *A cook stove is used to prepare a meal* contain basic knowledge used for semantic reasoning and related tasks. ConceptNet extracts such sentences and translates

[24] `https://www.mpi-inf.mpg.de/departments/databases-and-information-systems/research/yago-naga/uwn/`

[25] `https://www.openthesaurus.de/`

[26] `http://en.wiktionary.org/wiki/Wiktionary:Wikisaurus`

[27] `https://wiki.openoffice.org/wiki/Dictionaries`

[28] `http://www.openoffice.org/lingucomponent/thesaurus.html`

[29] `http://conceptnet5.media.mit.edu/`

them into semantic relations like *"cook stove used_for preparing meals"*. Additionally, ConceptNet connects knowledge from the Princeton WordNet, parts of DBpedia, Umbel and Wiktionary. Though there are many useful linguistic relations in ConceptNet, there is also numerous useless or irrelevant information, which is an obvious consequence of the automatic extraction approach. For instance, relations like <*computer*, `is-a`, *friend*> can be found in the data sets, which may origin from a sentence like *"The Computer is your friend."*. ConceptNet was used in the context of this research, and 687,000 of the approx. 8.7 million English relations had been extracted. However, this background knowledge led to worse results in the evaluation, mostly because of the many imprecise relations. Thus, it was not further pursued in this research, yet could be successfully used in other fields, e.g., for query expansion [73] or word sense disambiguation [74, 31], where a combination of WordNet and ConceptNet apparently led to the best results.

Umbel (Upper Mapping and Binding Exchange Layer) is a top-level (reference) ontology primarily used for Linked Open Data resources [105]. It defines some 28,000 general concepts within a strict taxonomic hierarchy and serves as a reference framework for domain ontologies by promoting the interoperability with external data sets and domains.[30] Among the concepts there are also some entities (such as car brands and companies).

The Cyc project is a logical framework to formalize common sense knowledge [86]. Founded in 1984, it was continuously extended by knowledge engineers at CycCorp and is primarily designed to foster artificial intelligence and semantic reasoning [90]. The open source version of this large knowledge base, called **OpenCyc**, provides endpoints for semantic web tools.[31] The concepts and assertions were manually created and concentrate on natural, simple facts like *cottage is made of wood* and *wood is able to burn*. Using Cyc, a reasoner is thus able to conclude that a cottage is able to catch fire. The knowledge is represented in the Cyc Language (CycL) and stored in different Cyc ontologies which are provided for download free of charge. OpenCyc was diversely used in the field of semantic web, e.g., to classify Wikipedia articles [113], for document annotation [151] or for rule generation in event processing systems [100]. The latest version 4.0 comprises about 2.1 million triples and 239,000 terms that are organized within a manually designed ontology.

FrameNet[32] is a different approach of organizing vocabulary of a language. Instead of synsets, hypernyms, meronyms or antonyms, the lexical database consists of semantic frames that can refer to a process type, situation type or event type. A semantic frame describes the entities and participants that take part in such a type, and which relations hold between them. For instance, the semantic frame *transfer* specifies that there is a person A (the "donor"), a person B (the "recipient") and an object C (the object to transfer). The frame can be activated by verbs like *to transfer*, *to give*, *to hand*, etc. Frames are related to each other, e.g., the frame "committing crime" is related to "crime investigation", which again is related to the frame "criminal process" (the relation type can be considered as *entails*) [49]. FrameNet consists of 800 hierarchically arranged frames that can be triggered by more than 10,000 elements, the so-called "lexical units" [83].

[30] `http://www.umbel.org/`

[31] `http://sw.opencyc.org/`

[32] `https://framenet.icsi.berkeley.edu/fndrupal/`

UMLS[33] is a large domain-specific knowledge base and thesaurus for the biomedical domain [20]. It was founded as early as in 1986 and combines the vocabulary from various biomedical thesauri and taxonomies in the so-called MetaThesaurus[34]. Additionally, the terms in the thesaurus are linked to the Semantic Network[35] of UMLS, which serves for classification and provides a large set of relationships between concepts. Altogether, 133 semantic types and 54 different relationships are distinguished, so that UMLS tends to be both a thesaurus and a knowledge base. It contains about 707,000 terms and 58 million relations, though there are many duplicate (or even multiple) relations. For example, there may be `part-of` relations of the kind *(Nail, Toe)*, *(Nail, toe)*, *(nail, Toe)* and *(nail, toe)* since UMLS is case-sensitive. Additionally, some obvious `equal` relations like *(Stomach, stomach)* are contained.

There is also a selection of knowledge bases that were manually developed. **WikiData**[36] is a collaboratively generated, machine-readable knowledge base about facts and entity data (like birth dates of persons), thus being similar to the automatically developed DBpedia project (see below). Similar to WikiData, but possibly stronger focusing on the Semantic Web and Linked Open Data cloud, **Freebase**[37] is a knowledge base which contains millions of facts for a broad range of topics which were entered by volunteers [21]. **GeoNames**[38] is a collection providing more than 10 million geographic names like countries, towns or rivers, together with information about geo-coordinates, elevation, population, etc. The knowledge base was derived from many geographic resources while volunteers are able to add new information or update existing ones.

Even with these many available resources, WordNet remains the only solid and appropriate resource for the schema and ontology matching domain. Many match approaches use WordNet as additional background knowledge [57, 85, 78, 122, 36, 81, 130, 103], but practical no other linguistic resource is used. Some match tools used for the biomedical domain exploit UMLS as domain-specific background knowledge, e.g., [85, 78, 81], but suitable alternatives for the general purpose domain are not available. We see a high demand for such general purpose resources to improve schema and ontology mapping based on background knowledge.

4.2.4 Automatically Generated Approaches

In addition to the user-generated thesauri and knowledge bases, many resources are generated automatically or at least semi-automatically (with manual interaction). In this context, Wikipedia plays a crucial part. It was exploited by most of the approaches, because of the vast amount of information it comprises, as well as the relatively high data quality, currentness and free availability.

[33] `http://www.nlm.nih.gov/research/umls/knowledge_sources/metathesaurus/index.html`

[34] `http://www.ncbi.nlm.nih.gov/books/NBK9684/`

[35] `http://www.ncbi.nlm.nih.gov/books/NBK9679/`

[36] `http://www.wikidata.org`

[37] `http://www.freebase.com/`

[38] `http://www.geonames.org/`

DBpedia[39] is among the most prominent and successful knowledge databases [11]. It is based on Wikipedia articles and exploits structural and syntactic specifics to extract valuable information about entities. The main focus is on the Wikipedia info boxes that provide structural information of entity data (especially persons, locations, albums, movies and the like, as well as chemical elements, biological or medical classifications), but also URLs, geo-coordinates and categories are detected and extracted. The authors use a pattern matching approach based on recursive regular expressions in order to find relevant templates in the raw article texts. If such templates are found, the page fragment is parsed and relevant information is extracted and post-processed to ensure high data quality [12].

Information extracted from Wikipedia is stored in an RDF framework and is interlinked with other data sources on the Web. This makes DBpedia to a central information hub in the Linked Open Data community. DBpedia provides SPARQL endpoints to retrieve information from its knowledge base, a web interface for manual information lookup and additionally allows downloading some content in SQL format. DBpedia is provided for 125 different languages. The current version of the English DBpedia describes about 4.5 million concepts, and about 580 million facts were extracted from the English Wikipedia.[40]

YAGO[41] is a similar project to DBpedia, but uses different procedures to generate knowledge. Instead of extracting structural information from article texts, YAGO matches Wikipedia articles (mostly entity data) to WordNet synsets. It thus combines a lexicographic resource with a knowledge base and contains both lexicographic relations (like *subclass-of*) and individual relations like *wasBornIn*, *hasWonPrize* or *locatedIn*. Relations between Wikipedia pages and WordNet synsets are derived by analyzing the categories where a specific Wikipedia page is located, analyzing the category title, extracting the most relevant concept term and finally matching it with WordNet. The category names are also analyzed w.r.t. relation detection, e.g., an article appearing in the category *1879 births* would lead to the relation *"Person" bornIn 1879*; category names starting with *Cities in* or *Attractions in* indicate that the respective article describes a place located in the specified region. Further on, re-direct pages are used to detect synonymy (*sameAs*) relations. The accuracy of YAGO relations is between 90.8 and 98.7 %, depending on the relation type [142]. YAGO currently consists of about 10 million entities being organized within about 350,000 conceptual classes. It contains about 120 million facts (relations) in 10 different languages.

Focusing more on the linguistic context, **BabelNet**[42] is a sophisticated multilingual approach that links Wikipedia pages to WordNet senses [102]. Given a Wikipedia page title like *balloon (aeronautics)*, the difficulty of this approach is to find the corresponding WordNet sense, which may be $balloon_1$. To find the correct correspondence, the categories and external links of the Wikipedia page are analyzed, as well as synonyms, hypernyms and hyponyms of each possible WordNet sense. The WordNet sense with the largest inter-

[39] `http://dbpedia.org`

[40] `http://wiki.dbpedia.org/about/facts-figures`

[41] `http://www.mpi-inf.mpg.de/departments/databases-and-information-systems/research/yago-naga/yago`

[42] `http://babelnet.org/`

section is chosen as match partner. Subsequently, WordNet synsets are enriched by the different languages where a Wikipedia article is provided, e.g., $Balloon_{EN}$, $Ballon_{DE}$, $Aerostato_{ES}$, which can be easily attained, as Wikipedia pages contain lists of equivalent articles in other languages. BabelNet can also retrieve translations in case that a link to a specific language is not available, by using the SemCor[43] corpus and machine translation techniques. The mapping algorithm for mapping Wikipedia pages to the respective WordNet sense achieved a precision of 82 % and a recall of 78 %. However, since BabelNet links Wikipedia pages to WordNet senses, it does not generate new semantic relations within a language and is rather useful in case that ontologies of different languages should be mapped.

The most recent version, BabelNet 3.0, covers 271 different languages and stores 6.4 million concepts in 13.8 million synsets. It contains more than 354 million lexicographic-semantic relations.[44] In addition to WordNet, further background knowledge resources have been automatically integrated, such as Wiktionary and Wikidata.

Sumida and Torisawa exploit the structuring of Wikipedia pages (such as headings, subheadings, sub-sub-headings, etc.) together with pattern matching and linguistic features to extract hyponym relations from the Japanese Wikipedia [143]. They were able to retrieve 1.4 million relations with a precision of about 75 %.

While the previous approaches focus rather on structural or semi-structural techniques for information extraction from Wikipedia, some approaches try to extract semantic relations directly from the article texts. Hearst patterns are an important prerequisite to extract semantic relations from unstructured texts. They comprise expressions like "A, such as B and C" or "A, which is a specific B" and indicate synonym and hyponym relations (for instance, B and C are hyponyms of A in the first example). Several approaches to learn such Hearst patterns are based on Wikipedia texts [124, 125] or on newswire corpora [135]. In [72], Hearst patterns are used to derive an ontology from biomedical Wikipedia articles. They obtain a rather poor recall (20 %), but excellent precision (89 %).

In [18] the authors developed patterns for part-of relations based on Hearst patterns. Using a newspaper corpus comprising 100 million words, they could achieve an accuracy of 55 %, meaning that 55 % of the extracted part-of relations were considered to be correct. Other approaches focus on machine learning of part-of relations [55] or on detecting part-of relations in corpora by means of word similarity measurement [89].

In [23], the authors developed an approach to extend WordNet by so-called telic relations by searching for specific patterns in the WordNet glosses. Telic relations describe the purpose between concepts like *wood – fire* or *wood – furniture* ("used for" relations) and are especially relevant for text analysis or question answering.

Further on, Wikipedia is widely used for related linguistic tasks, which do not primarily focus on semantic relation extraction. For example, Flati and Navigli parse Wikipedia articles to find the collocations of specific words, e.g., *"break *"*. From the results they obtain, they derive more general concepts like "break `<Agreement>`" or "break `<Body Part>`"

[43] `http://www.gabormelli.com/RKB/SemCor_Corpus`
[44] `http://babelnet.org/stats`

[50]. Wikipedia is also used to determine the semantic relatedness between concepts or expressions [139, 51] and for word sense disambiguation [114].

WikiTaxonomy is a further approach that extracts semantic relations from Wikipedia, which is based upon an earlier approach of Wikipedia taxonomy extraction [115]. The authors parse the whole category system of Wikipedia and could extract some 100,000 is-a relations using lexicographic, syntax-based and inference-based techniques [116]. The extracted taxonomy comprises both entities and concepts and the authors reached an F-measure of 88 % on the extracted relations. They also use approaches to designate entities and concepts within the extracted taxonomy [158]. Wu et al. use Hearst patterns based on a large web corpus. Parsing 1.7 billion web pages, the authors could extract more than 2.6 million concepts and about 20.7 million relations between concepts and sub-concepts as well as concepts and instances. Though their focus is more on language-specific techniques, like named entity recognition and question answering, the gathered knowledge seems also suitable for schema and ontology mapping [152].

MultiWordNet[45] is a multilingual thesaurus similar to the manually developed EuroWordNet. Instead of interlinking existing word nets, the authors propose the development of new word nets based on the semantic structure of the English Princeton WordNet. They obtain such word nets by the semi-automatic development of synsets from the Princeton WordNet using bilingual dictionaries [111]. As a result of this project, an Italian word net consisting of 28,120 synsets and 36,514 words has been derived, whose semantic structure is mostly identical to WordNet. However, no further word nets have been developed so far, but the project also interlinked some already existing resources. Altogether, it allows access to 7 languages (English, Italian, Spanish, Portuguese, Hebrew, Romanian and Latin).

4.2.5 Web-based Approaches

In addition to the development of background knowledge resources, much research has been dedicated to theoretic approaches and techniques to use background knowledge effectively.

A very intuitive way of using the web as background knowledge resource was introduced in [65]. Given two match candidates A, B the authors combine them with Hearst patterns, e.g., *A, including B* or *A, such as B* and send them to a search engine. Subsequently, they count the number of hits and analyze the result snippets to decide whether the concepts are related or not. The approach focuses solely on subclass relations and was developed for the food domain. Using an additional food dictionary for further improvement, they achieved a very good precision (up to 94 %), yet had to acknowledge that the approach does not scale well, mostly because of the limitations of search APIs (which was 1000 queries/day for the search engine they used).

Similarly, Gligorov et al. use Google to assist ontology matching. Given two terms, the normalized Google distance is calculated by counting the number of Google hits for ev-

[45] `http://multiwordnet.fbk.eu/english/home.php`

ery term and comparing it with the number of hits for both terms (co-occurrence in a web page) [59]. In [32], the authors follow a similar approach to determine some specific relation types between frequently co-occurring verbs, which are more relevant for fields like question answering or machine translation. In [22], the authors also use Google to specify the co-occurrency of two words A, B by applying four different measures (Jaccard, Simpson, Dice, PMI) and by analyzing the returned result snippets.

In [126] the authors present a method to use *Swoogle*, a search engine for ontologies, to automatically and dynamically find appropriate background knowledge resources for a given match task.[46] The results returned by Swoogle are then used to discover equivalence, subset and disjoint relations. The approach can exploit either one or several background knowledge sources. In the latter case, the authors have to deal with inconsistent or even contradicting results (like one resource suggesting an equivalence relation and another resource suggesting disjointness). Other approaches like [92] use a collection of schemas and mappings as background knowledge for matching.

4.2.6 Juxtaposition of Background Knowledge Approaches

Fig. 4.1 provides a juxtaposition of selected background knowledge resources and distinguishes between concept- and entity-oriented resources as well as manually and automatically generated resources. Manually generated resources are subdivided into resources created by domain experts and resources created by a crowd of volunteers. Automatically generated resources are subdivided into structure-oriented approaches that focus on linking concepts, matching or structural techniques, and text-oriented approaches that extract the semantic relations by working on natural texts and exploiting typical NLP-techniques like sentence parsing. Resources that contain at least 1 million concepts are highlighted gray and it can be seen that those resources were usually collaboratively or automatically created. The chart only contains selected resources, with the focus on resources that are publicly available or that became established and popular resources in the field of semantic web, natural language processing, reasoning, information retrieval and related areas.

On the very left side of the chart are the resources created by domain experts. On top are Umbel and FrameNet, which concentrate strongly on concepts, while WordNet (and EuroWordNet and GermaNet and all related projects) occur slightly below, as these resources also contain a few entities like famous persons or cities. OpenCyc and UMLs comprise both concepts and entities, while GeoNames is a typical entity-oriented resource. It appears more to the right, as GeoNames also allows volunteers to improve and update the data sets.

Further to the right appear the resources that were created by a crowd of volunteers. Such crowds usually include laymen, which means that the overall quality of such resources might be slightly below resources created by a selected group of experts. WikiSaurus and OpenThesaurus are concept-oriented while WikiData and Freebase rather focus on entities. ConceptNet is a special resource which was partly manually created, and partly

[46] `http://swoogle.umbc.edu/`

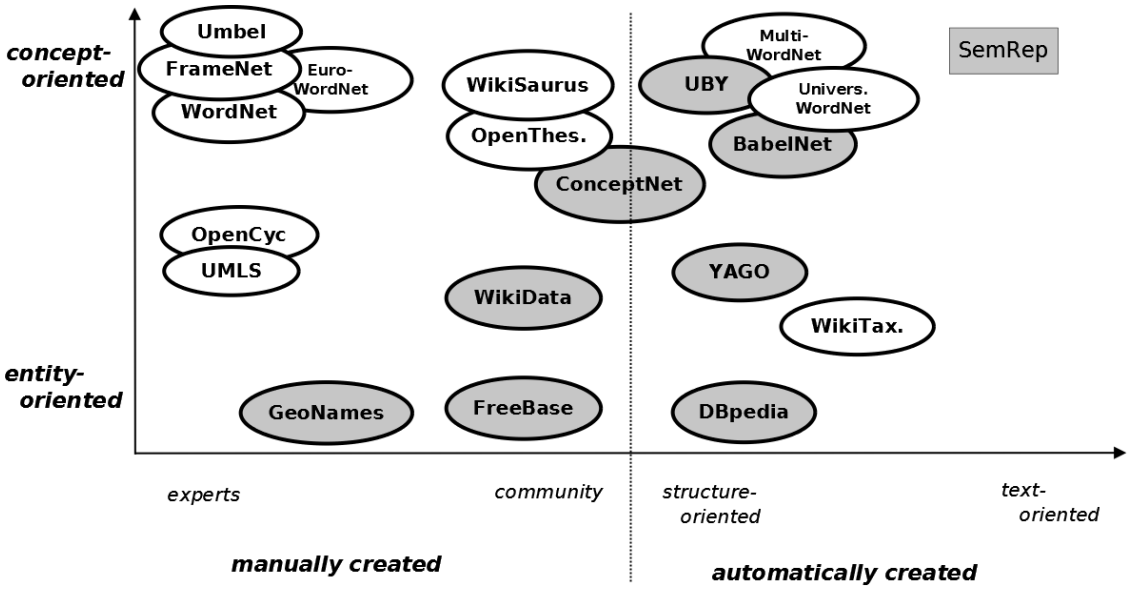

Figure 4.1: Classification of selected background knowledge approaches.

automatically by connecting the knowledge of further resources. It is rather concept-oriented, but also contains facts about entities.

Still further to the right appear the resources that were automatically or semi-automatically built. UBY and BabelNet are rather concept-oriented, just as MultiWordNet and Universal WordNet. YAGO combines both conceptual knowledge and entity data. WikiTaxonomy also contains entity and concept data, but since Wikipedia, the resource of WikiTaxonomy, contains much more instance data than concept data, it tends to be more entity-oriented. Since the approach exploits both syntactic strategies (e.g., parsing the category tree of Wikipedia) and string-pattern strategies, it is best situated between the structure- and text-oriented resources. Eventually, DBpedia is a typical entity-oriented approach exploiting structural information of Wikipedia pages (info boxes, headings, URLs, etc.).

In the top-right corner is the repository SemRep, which is an essential part of this thesis and will be described in detail in Part III. So far, no successful conceptual background knowledge resource has been built by means of NLP or entirely text-oriented techniques, although there generally exist such approaches, which have been introduced and discussed above.

To conclude this section, Table 4.2 provides a full overview of the presented resources. It is oriented towards the previously introduced classification and thus describes the way how each resource has been developed (column 2), the area that is covered (column 3), the kind of data that prevails (column 4), the supported link types (column 5), the supported languages (column 6), statistical information about the comprehensiveness (column 7) and lastly information about the availability (column 8). For comparison, SemRep was also included in this table (last row). Link types can be lexicographic, referring to relation types like synonymy, antonymy and co-hyponymy, or specific, referring to relations like

lives in, *is used for*, *is manager of*, etc. If the concrete lexicographic relations used in a resource are known, they are fully mentioned.

Approach	Develop.	Area	Data	Link Types	Lang.	Size	Avail.
BabelNet	automatic	general purp.	concepts	lexicographic	multi-ling. (271 lang.)	6.4 mio. concepts, 354 mio. lexicographic relations	free
ConceptNet	manual / collab., semi-automatic	general purp.	concepts, entities	lexicographic, specific	multi-ling. (focus on *EN*)	2.7 mio. concepts, 8.7 mio. relations (EN)	free
DBpedia	automatic	general purp.	concepts, entities	specific	multi-ling. (271 lang.)	4.6 mio. concepts, 580 Mio. facts (*EN*)	free
Euro-WordNet	manual	general purp.	concepts	synonyms, hypernyms, meronyms, co-hyponyms, antonyms	multi-ling. (8 lang.)		mostly comm.
FrameNet	manual	general purp.	concepts	specific, partly lexicographic	mono-ling. (*EN*)	800 frames, approx. 10,000 lexical units	comm.
FreeBase	manual / collab.	general purp.	entities	specific	mono-ling. (*EN*)	47.5 mio. topics, 2.9 billion facts	free
GeoNames	manual / collab.	geography	entities, some concepts	lexicographic, specific	mono-ling. (*EN*)	10 mio. terms	free
GermaNet	manual	general purp.	concepts	synonyms, hypernyms, meronyms, co-hyponyms, antonyms	mono-ling. (*DE*)	111,000 terms	comm.
Multi-WordNet	manual, semi-automatic	general purp.	concepts	synonyms, hypernyms, meronyms, co-hyponyms, antonyms	multi-ling. (7 lang.)		free
OpenCyc	manual	general purp.	concepts, entities	lexicographic, specific	mono-ling. (*EN*)	239,000 terms, 2.1 mio. relations	free
Open-Thesaurus	manual / collab.	general purp.	concepts	synonyms, hypernyms, meronyms, co-hyponyms, antonyms	mono-ling. (*DE*)	93,000 terms	free
Princeton WordNet	manual	general purp.	concepts	synonyms, hypernyms, meronyms, co-hyponyms, antonyms	mono-ling. (*EN*)	155,000 terms	free
UBY	automatic	general purp., specific domains	mostly concepts	lexicographic, specific	multi-ling. (*DE*, *EN*)	4.2 mio. concepts, 5.3 mio. relations	free
Umbel	manual	general purp. (top-level)	concepts, some entities	synonyms, hypernyms	mono-ling. (*EN*)	28,000 concepts	free
UMLS	manual	biomedical domain	concepts, entities	lexicographic, biomedical-specific	mostly mono-ling. (*EN*)	707,700 terms, 58 mio. relations	comm.

Universal WordNet	automatic	general purp.	concepts	synonyms, hypernyms, meronyms, co-hyponyms, antonyms	multi-ling. (> 200 lang.)	> 1.5 mio. terms	free
WikiData	manual / collab.	general purp.	entities	specific	mono-ling. (*EN*)	14 mio. terms	free
WikiSaurus	manual / collab.	general purp.	concepts	synonyms, hypernyms, (meronyms), antonyms	mono-ling. (*EN*)	1,400 categories	free
Wiki-Taxonomy	automatic	general purp.	concepts, entities	hypernyms	mono-ling. (*EN*)	> 100,000 is-a relations	not avail.
Yago	automatic	general purp.	concepts, entities	lexicographic, specific	multi-ling. (10 lang.)	10 mio. entities, 350,000 concepts, 120 mio. relations	free
SemRep	automatic	general purp.	concepts	lexicographic	multi-ling. (*DE*, *EN*)	2.7 mio. concepts, 5.6 mio. relations	free

Table 4.2: Full overview of the different approaches.

4.3 Mapping Enrichment and Repair

A first approach for semantic mapping enrichment was introduced in 2004 by Su et al. Instead of enriching mappings (the output of a match task), the authors try to enrich the input ontologies with semantics to facilitate the match process and to obtain better results [141]. They first assign relevant documents to the input ontologies and afterwards build feature vectors for each concept in the ontology. When matching the two ontologies, such feature vectors are taken into account, which provide much more semantics than the sole concept name. A similar approach was introduced in [149], where concepts within an ontology are enriched by meta information to facilitate ontology matching. However, approaches for mapping post-processing are rare and rather focus on mapping repair, in which the mapping is tested for consistency and coherency and, if necessary, is revised to achieve this state [82, 107]. Only a small selection of tools regards mapping repair, e.g., LogMap [81] and Alcomo [98, 97], which use reasoning-based techniques to attain such a mapping coherency. Further approached for mapping repair were introduced in [127], [150] and [136].

In Spicy, an automatic schema mapping tool, a so-called *mapping verification* step is used based on the structural analysis of the input data. Given a correspondence in the match result, alternative correspondences are generated and the transformed data objects are compared with the data objects of the target system. This method called *post-translation check* can discover erroneous correspondences in case that an alternative correspondence leads to a transformation result that is closer to the target data than the transformation result produced with the original correspondence [25]. ASMOV performs a verification step after the match phase and removes correspondences that are unlikely to be in accordance with the information encoded in the ontology. This includes contradictions between the correspondences and the ontology structures, or contradictions between cor-

respondences within the mapping [78]. Another approach conducts mapping repair on biomedical mappings by means of description logic and reasoning. The approach could successfully discover false correspondences caused by a high (misleading) lexicographic similarity, such as *sexual function* and *sexual dysfunction* [33]. These approaches are closer to the STROMA approach, as they perform two separate steps (matching and verification), but STROMA focuses rather on enrichment and not on verification.

At present, there appears to be no approach that conducts mapping enrichment in a two-step approach as STROMA does, except for the above mentioned verification approaches. In the rest of this doctoral work, we will elucidate the gist of such an enrichment approach, the techniques used to determine the semantic relatedness and the general benefits of such an approach compared to 1-step approaches.

Part II

The STROMA System

5 STROMA Architecture and Workflow

In this chapter, we will present the mapping enrichment tool STROMA, which obtains a simple initial mapping and returns an enriched mapping with the relation types assigned to each correspondence. We will start with a general introduction of the tool in Section 5.1 and explain the Stroma architecture and the workflow for mapping enrichment in Section 5.2. Finally, in Section 5.3 we discuss some technical and implementation details.

5.1 Introduction

STROMA (**S**eman**T**ic **R**efinement of **O**ntology **MA**ppings) is a prototype for mapping enrichment, which obtains a pre-calculated mapping as input and returns a semantically enriched mapping as output. Two important forms of enrichment are carried out: First, STROMA determines the relation type that holds between the two matching concepts of each correspondence (relation type detection), and second, correspondences are verified for validity. If there is no evidence for such a correspondence, it is assumed to be a false match and will be removed from the mapping. This step is called *mapping repair* and can increase the original precision of the mapping.

Technically, mapping repair is a form of mapping *improvement*, not a form of mapping *enrichment* in the narrow sense (since nothing is added to the given mapping). However, we will henceforth combine these two techniques, relation type detection and mapping repair, under the term *enrichment*.

The approach presented in this doctoral thesis is thus a two-step approach in which first an initial mapping is calculated (step 1), which is subsequently enriched by STROMA (step 2). Compared to other approaches that try to obtain semantic mappings in one step,

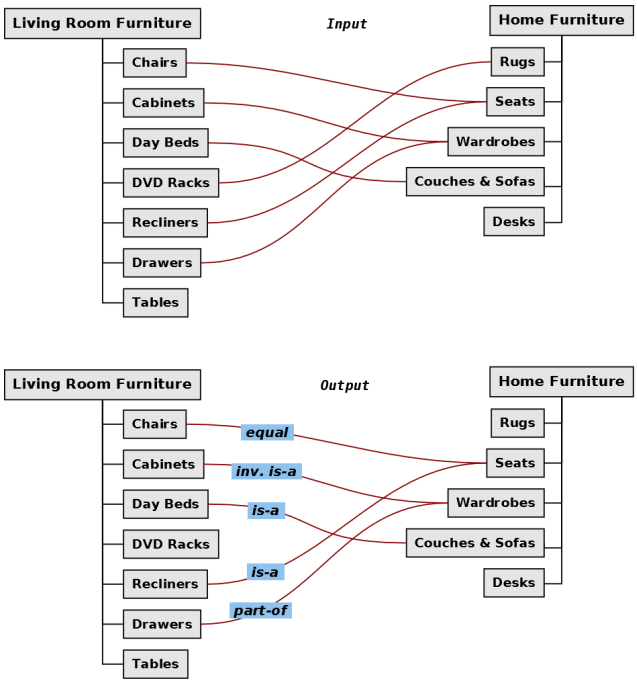

Figure 5.1: Input and output example of a mapping processed with STROMA.

STROMA focuses much more on the relation type detection than related approaches that often determine the type only secondarily.

Fig. 5.1 illustrates the mapping enrichment using a simple example. The above chart shows a representation of the input mapping as provided by a match tool. It consists of 6 correspondences between two furniture taxonomies. One of the correspondences, the match between *DVD Racks* and *Rugs*, is a false correspondence, because there is no semantic relation between the two terms. Additionally, one correspondence is missing in the mapping, namely, the correspondence between *Tables* and *Desks*. The chart below shows a representation of the STROMA output. The tool determined the relation type for each correspondence and correctly removed the false correspondence between *DVD Racks* and *Rugs*. However, as STROMA only works on the input mapping and not on the source ontologies, it is unable to find the missing correspondence *(Tables, Desks)*. Detecting missing correspondences would require to perform a post-matching within the given mapping or, preferably, on the original two taxonomies. The latter form implies that the two source taxonomies have to be parsed and loaded into STROMA before the post-matching could commence. Since this doctoral thesis is about mapping enrichment on a given mapping, not on post-matching, we will not broach this topic here, but will refer to it as possible future work.

One of the key features of STROMA is its relatively high flexibility. It receives nothing but a simple list of correspondences as input, where a correspondence is a tuple *(source path, target path)*. Optionally, a correspondence can be a triple *(source path, target path, confidence)*, but the confidence value is not mandatory, although it is necessary for mapping repair (selection). In contrast to classic match tools, STROMA does not require the source

and target schemas or ontologies, but only works on the given input mapping. There is still much information in such mappings (especially the correspondence paths), which are also taken into account by several implemented strategies.

The simplicity of the input format enables any match tool to use STROMA for enrichment, because any match tool calculates at least a list of correspondences of the form *(source concept, target concept)*, and it can be expected that the translation of an internal mapping into the STROMA input format could be accomplished without much effort. In the context of evaluations, we will use the match tool COMA 3.0 for the first step, but technically, any other tool can be used, as long as its match result is converted into the input format required by STROMA.[47]

However, a drawback of such a two-step approach is its high dependency on the input mapping. If correspondences are missing in the mapping (lower recall), STROMA is unable to find and enrich those correspondences, so its recall is automatically limited. If correspondences are incorrect or irrelevant, any type that STROMA denotes is automatically false, because the correspondence is false in the first place (or at least it is not part of the expected mapping). Mapping repair and related techniques considerably help to deal with false correspondences, but the recall issue cannot be solved this way. Therefore, the overall quality of STROMA also depends on the quality achieved in step 1. Another issue is that many match tools focus on equality-relations, while STROMA attempts to provide mappings of all semantic types. The careful selection and configuration of a suitable match tool for the first step is therefore an indispensable part in mapping enrichment.

5.2 STROMA Workflow

5.2.1 Overview

Fig. 5.2 depicts at a high level the general workflow processed by STROMA. As already illustrated, a match tool calculates a mapping between two given ontologies in step 1, which is the input for the mapping enrichment conducted by STROMA (step 2). The workflow of mapping enrichment consists of two phases: Relation type detection and selection (mapping repair). The two phases are carried out consecutively, i.e., the selection phase only commences when the type detection phase is accomplished. In this thesis, the main focus is on the relation type detection, while mapping repair is a subordinate issue.

In the first step, type detection, STROMA iterates through all correspondences and each correspondence is passed to six different strategies that independently try to determine the relation type of the correspondence at hand. Each strategy returns either a specific relation type, like `is-a`, or `undecided` if no type can be determined. By default, all strategies are enabled, though it is generally possible to disable strategies (e.g., to only use background knowledge). Since the six strategies can return different types, the results have to be analyzed and a final relation type has to be determined, which is carried out by the *Type Computation* component (see Section 5.2.2). The final type is then verified for

[47] `http://dbs.uni-leipzig.de/Research/coma.html`

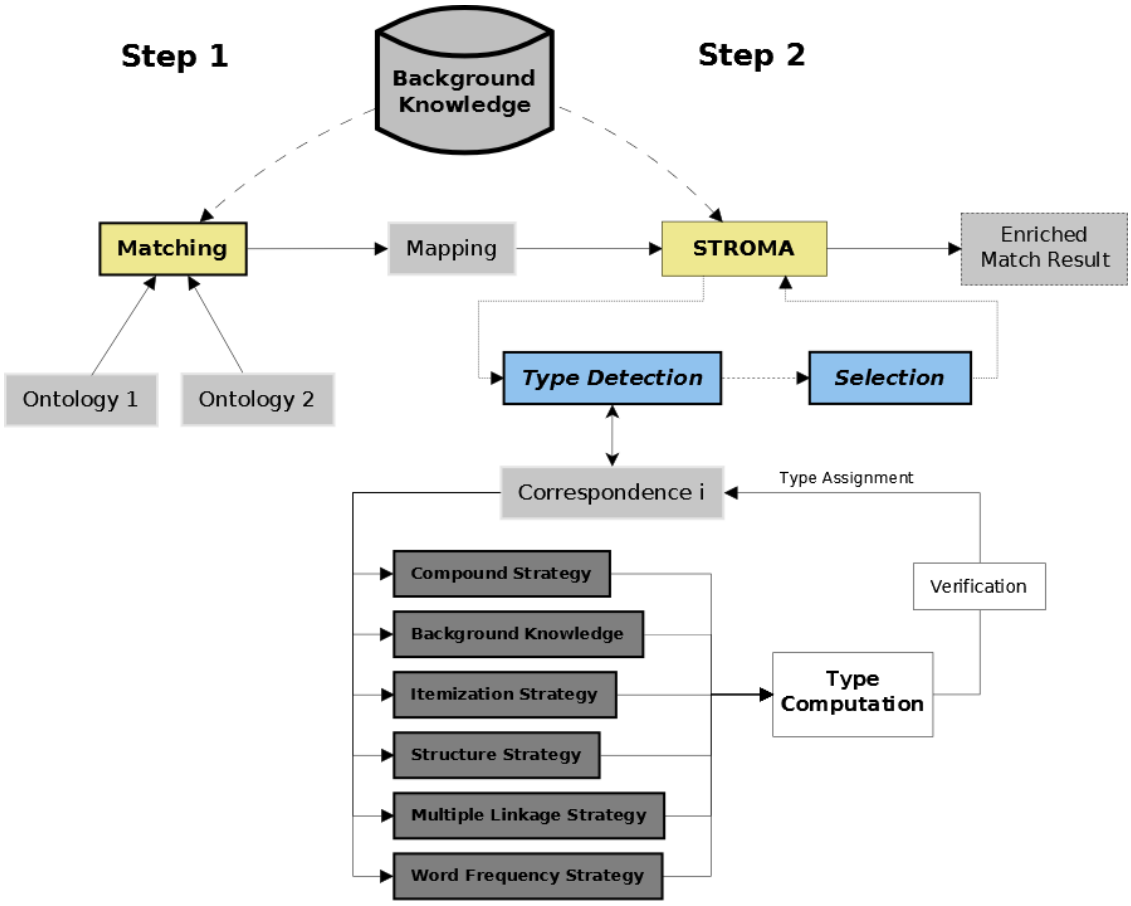

Figure 5.2: Illustration of the two-step approach and the mapping enrichment workflow.

plausibility, i.e., different techniques drawing on contextual information are used to corroborate the type or to reject it (see Section 6.7). The correspondence is then denoted by one specific relation type or by `undecided`.

The second phase is the selection phase, in which all correspondences in the mapping are once again iterated and checked for linguistic plausibility. If the relation type of a correspondence with an already low score could not be calculated, or if there is enough linguistic evidence that a correspondence is not true, it will be removed from the mapping.

In both steps and in all phases, background knowledge like dictionaries, thesauri or domain-specific ontologies can be applied. While this is optional in Step 1, depending on the tool that is used for the initial mapping, STROMA practically exploits background knowledge in all phases and sub-steps it performs.

5.2.2 Type Computation

Let S be the set of strategies used by STROMA, and R the set of relation types STROMA is able to determine (in this subsection we treat `undecided` as a type, too). Each strategy $s \in S$ returns exactly one relation type $r \in R$.

Internally, the result for a specific correspondence is represented as an $S \times T$ matrix, because each strategy can return any of the 7 relation types. To determine the final relation type, it appears natural to use the type returned by the majority of strategies. Though STROMA is generally based on this notion, this approach is too simple, because the different strategies differ slightly in their general reliability. For instance, Compound and Background Knowledge are quite reliable strategies that normally return satisfactory results. The strategy Word Frequency is more heuristic and seems less reliable, though, and its results should have less impact on the overall result determination. Therefore, each strategy s has a specific weight $w(s)$, which specifies how much reliability is assigned to it. By default, STROMA uses the weights 1.0 for Compound and Itemization, 0.9 for Background Knowledge, 0.8 for Multiple Linkage, 0.7 for Structure and 0.6 for Word Frequency. The result matrix would look as depicted in Table 5.1. Compound and Background Knowledge return both `is-a`, and according to the strategy weights, a score of 1.0 resp. 0.9 is achieved for each strategy. The Structure Strategy returns `equal`, but according to its weight, only a score of 0.7 is achieved for this type.

Eventually, the overall score for each type is the sum of the scores produced by each strategy. For `undecided`, no specific score is calculated, because a type is assigned to a correspondence as soon as it has an overall score above 0 – this makes scores for the result type `undecided` unnecessary. In the given example, STROMA would eventually decide on the type `is-a`, because it obtained the highest score.

Strategy	equal	is-a	inv. is-a	part-of	has-a	related	undecided
Compound		1.0					
Background Kn.		0.9					
Itemization							
Structure	0.7						
Multiple Linkage							
Word Frequency	–	–	–	–	–	–	–
Score	0.7	1.9	0.0	0.0	0.0	0.0	

Table 5.1: Sample matrix for a type result of processed correspondence.

Using different strategy weights for each strategy reduces the risk of draws, in which two relation types achieve the same score. Still, a draw is theoretically possible, e.g., Background Knowledge and Word Frequency together achieve a score of 1.5, which is also achieved by Structure and Multiple Linkage. In this very rare and unlikely case where a draw between two types is attained, the type with the highest preference will be applied. In STROMA, the preference of types is defined as follows (descending): `equal`, `is-a`, `inverse is-a`, `part-of`, `has-a`, `related`. This order is based on the distribution of relation types in most mappings. The types `equal`, `is-a` and `inverse is-a` normally predominate, while the other types occur less often. Therefore, STROMA will assign the type which is more likely to hold.

Occasionally, neither of the six strategies is able to calculate any relation type, i.e., each strategy returns `undecided` and each type will have a score of 0. STROMA allows two configurations in this case: It can denote the correspondence with the label `undecided` and let the user decide on the correct relation type (manual interaction), or it denotes the correspondence with `equal` by default. Since many match tools tend to detect equivalence relations, and since this type is often the most frequently occurring one in mappings, `equal` seems to be the most likely type to hold if no type could be calculated. By default, this second configuration is applied, which we call *undecided-as-equal* in the evaluation of mappings. The opposite configuration is *undecided-as-false*, i.e., a correspondence not having any type assignment is treated as falsely typed. A list of all weights and default configurations used in STROMA is also provided in Appendix A.

5.2.3 Selection

STROMA can use mapping repair techniques to remove false correspondences, which can lead to a better mapping precision. However, since STROMA works on a given mapping, there is practically no chance to achieve a better recall. Therefore, it is advised to use relaxed match configurations in step 1, which generally leads to larger mappings containing more correspondences. Such mappings will have a better recall, but a lower precision, which STROMA tries to augment by means of different repair techniques.

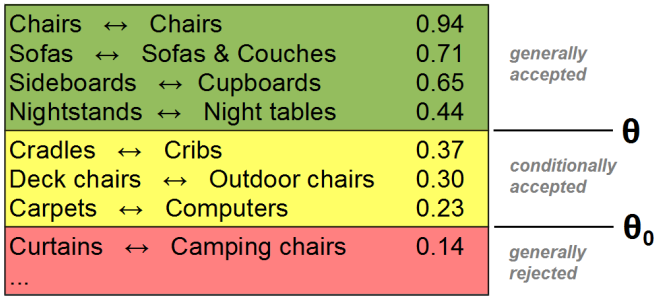

Figure 5.3: Sample mapping with the two thresholds θ and θ_0.

The match tool COMA 3.0 was used in the initial phase for the mappings processed with STROMA. It normally uses a selection threshold $\theta = 0.4$, which means that correspondences between two concepts must achieve a score of 0.4 or higher in order to be accepted for the mapping. In the context of this research, a lower threshold $\theta_0 = 0.2$ is used, which consequently results in larger mappings with a better recall, but worse precision. However, correspondences with a score between 0.2 and 0.39 are only treated as "conditionally accepted". If there is enough linguistic evidence that a correspondence from this range is correct, it is accepted to the final mapping, otherwise it is rejected.

This approach is also illustrated in Fig. 5.3. This initial mapping produced with COMA contains 4 correspondences above the threshold θ (3 of them are correct, while the third one is false), which will appear in the enriched mapping for sure. The 3 yellow correspondences following below are the conditionally accepted correspondences that are further

verified for correctness and are possibly kept in the enriched mapping. They have scores between θ and θ_0, and apparently two of them are valid, while one is false (the third one). Finally, the red correspondences having a score below θ_0 are generally rejected. They will not be part of the enriched mapping. Note that such correspondences are usually not part of the input mapping in the first place and are only shown for illustration. However, there would be no problem to use such an input mapping, as STROMA automatically ascertains the correspondences that are below θ_0 and removes them instantly.

In the example, it can be seen that the lower threshold leads to two further correct correspondences, which increased the mapping recall. There is also a further incorrect correspondence *(Carpets, Computers)*, which STROMA is able to detect, though. Thus, in this simplified example two further correct correspondences are found, but no further false correspondence would be part of the enriched mapping. The relaxed configuration thus led to a better recall without impairing the precision. On the opposite, the precision is even higher now, as 5 of 6 correct correspondences are in the final mapping (83.3 %) compared to 3 of 4 (75 %) correct correspondences if no lower threshold is used.

The threshold θ_0 is freely adjustable. If it holds $\theta = \theta_0$, no additional correspondences are regarded and no mapping repair will be carried out. Given a correspondence with a confidence c so that $c \geq \theta_0$ and $c < \theta$. This correspondence will be finally accepted if these two conditions are fulfilled:

1. STROMA could determine a specific relation type for this correspondence, i.e., the result is not `undecided`.
2. There is no evidence that the correspondence is a result of sloppy lexicographic matching.

The first point is relatively simple, as it seems natural to only accept critical correspondences if a specific relation type could be determined. Unfortunately, a type can also be found for irrelevant correspondences, especially by using background knowledge. If there is a correspondence *(chair, table)*, this might be a false correspondence, because *(chair, seat)* would be the correct correspondence. Still, STROMA could determine a type between *chair* and *table*, which might be `related`. Therefore, a score of at least 1.0 has to be reached to finally accept this correspondence, which entails that either 2 strategies have to return a specific type for this correspondence, or Compound or Itemization (which are quite reliable strategies). This score is called *minimal type confidence for acceptance*.

The second point is more intrinsic. As already mentioned in the introduction of this thesis, most match tools are based upon lexicographic matchers and often disregard linguistic laws. As we have already explained, a similar spelling of words is no hint at all for any semantic relatedness, as long as there is no linguistic evidence like inflection, compounding, derivation or shortening. In all these cases, changes only refer to the end of words, e.g., *(Computers, Computing)* or *(Book Shelves, Book Shelve Ladders)*. Therefore, if the first fragments of the matching concept do not overlap, they are most likely unrelated, as in *(stable, table)* or *(page, cage)*. This mapping repair technique checks whether the first letters of the matching words overlap. If this is the case, the correspondence is accepted, otherwise it is rejected. In the default configuration, the 5 first characters of

the two concepts have to overlap to accept the word. Thus, *(telephone, television)* would be rejected (only the first 4 letters overlap), while *(computer, computing)* will be accepted (6 letters overlap). As in many cases, this default configuration is based on experiences and can be freely adjusted in STROMA. If words are of length 4 or less, they are usually simplex words consisting of only one morpheme. The arbitrariness of language suggests that those words are normally not in any relevant relation and can thus be rejected.

It would be generally possible to extend this second technique by more complex techniques, e.g., by also handling similarly spelled compound words like *(city map, city council)* (mismatch) or *(city hall, town hall)* (match), which is quite a difficult undertaking, though. Currently, open and hyphenated compound words are never rejected, as there is a considerable likelihood that the two matching concepts are somehow related. Since mapping repair is not the primary focus of this thesis, as it is a much too complex research field, only the basic techniques described above are applied in STROMA. As we will show in the evaluation, they still allow further improvements of the mapping quality.

5.3 Technical Details

STROMA was developed in Java and consists of an engine, which comprises the core classes for mapping parsing and processing, as well as a test module to carry out experiments. This module provides a GUI to insert a mapping for enrichment, as well as a gold standard (perfect mapping) for verification. Therefore, the program can automatically determine recall and precision for a processed mapping (by comparing the enriched mapping with the gold standard). Additionally, an API was implemented that allows other tools to easily enrich mappings.

Input mappings are represented as CSV files and will be loaded by a text file parser. Periods are used to mark the different concepts in a concept hierarchy (path). Listing 5.1 shows a valid sample file containing 3 correspondences. Two colons are used as separator, because this character sequence will most likely not appear in any concept path. If no score for the correspondence is applied, the last column of the CSV file remains empty.

```
furniture.living_room.chairs :: home_furniture.seats :: 0.71
furniture.living_room.recliners :: home_furniture.seats :: 0.46
furniture.living_room.day_beds :: home_furniture.couches_&_sofas :: 0.49
```

Listing 5.1: Sample excerpt from a valid input file.

The enriched mapping is exported as a CSV file, where each correspondence is a 4-tuple *(source concept, target concept, confidence, type)*. The output format is identical to the input format, with the exception of a fourth column to store the relation type. Each relation type is encoded by a digit: `equal` by 0, `is-a` by 1, `inverse is-a` by 2, `part-of` by 3, `has-a` by 4 and `related` by 5. Listing 5.2 shows a sample excerpt from a valid export file. The simplicity of the output format allows other tools to further process the mapping, e.g., to carry out ontology merging or to generate a transformation script.

```
furniture.living_room.chairs :: home_furniture.seats :: 0.71 :: 0
furniture.living_room.recliners :: home_furniture.seats :: 0.46 :: 1
furniture.living_room.day_beds :: home_furniture.couches_&_sofas :: 0.49 :: 1
```

Listing 5.2: Sample excerpt from a STROMA export file.

The gist of STROMA is an iteration through all correspondences and each correspondence is passed to the six strategies. Let n be the number of correspondences in the mapping, this part of the workflow has linear costs of $O(n)$. After the type detection is performed, all correspondences are iterated again to remove irrelevant correspondences. This causes linear costs again, so that the overall time complexity is $O(n) + O(n) = O(n)$. Compared to other match tools, this is an excellent time complexity, though STROMA evidently depends on the match tools used for step 1. If a match tool with a complexity of $O(n^2)$ is used for this first step, the time complexity for the overall match process is $O(n^2) + O(n) = O(n^2)$, so STROMA has no negative impact on the time complexity.

STROMA can process mappings of 1000 correspondences within a few seconds, while match tools normally require much more time to calculate such voluminous mappings. Basically, match tools need to compare each concept pair of two input ontologies O_1, O_2. Without blocking, the number of comparisons is $\frac{|C_1| \times |C_2|}{2}$, where $|C_1|$ is the number of source concepts and $|C_2|$ is the number of target concepts. If either ontology consists of 1,000 concepts, 500,000 comparisons are necessary. By contrast, STROMA only needs to iterate through the 1,000 input correspondences which makes it much faster and allows it to exploit more time-consuming strategies like dictionary look-ups. Though the two-step approach is always slower than a first-step approach where STROMA is not used, the additional time needed to determine the correspondence types is usually rather low. Especially in large match scenarios, where the first step may take several minutes, the required time for the second step (enrichment) becomes almost negligible.

STROMA requires about 3 GB of RAM for mapping execution. While the core of STROMA only needs a few MB to store the correspondences, configurations and relation type matrices, most of the main memory is used for SemRep, the background knowledge repository used for the mapping enrichment. This repository also accounts for much of the initialization time of STROMA, which is about 45 seconds on a Windows server possessing two Intel Xeon E5540 CPUs (2.53 GHz). Without using SemRep, the loading time is about 1 second. We will discuss this subject in more detail in Part III.

6 Implemented Strategies

In this chapter, we will describe different strategies used to determine the relation type of a correspondence. Strategies are either independent or depend on other strategies. Some strategies are more generic and can be applied in most situations, while other strategies focus more on mapping- or ontology-specific aspects. Except for the Multiple Linkage Strategy, each strategy has a set of specific pre-conditions that must be fulfilled in order to be ran, and each strategy is able to determine a subset of the relation types provided in STROMA (at least two in numbers). All strategies have in common that if they obtain a *trivial correspondence* (c_1, c_2), for which holds $c_1 = c_2$, they automatically decide on type `equal`. In the following, we will start with the two base strategies *Compound* (Section 6.1) and *Background Knowledge* (Section 6.2), before we come to the more specific strategies *Itemization* (Section 6.3) and *Structure* (Section 6.4), as well as the heuristic strategies *Multiple Linkage* (Section 6.5) and *Word Frequency* (Section 6.6). In Section 6.7, we present different techniques to verify the final type obtained from the strategies. Eventually, we provide a summary and strategy comparison in Section 6.8.

6.1 Compound Strategy

6.1.1 Overview

The *Compound Strategy* is able to determine `is-a` and `inverse is-a` correspondences using the knowledge about the semantics of compounds as introduced in Section 3.1.2. For instance, in a correspondence *(kitchen table, table)*, the compound *kitchen table* matches its head *table* so that an `is-a` relation can be assumed. The implemented Compound

Strategy searches for correspondences between two words A, B, in which A ends with B or vice versa. If such a correspondence is at hand, the Compound Strategy is executed; otherwise, the strategy has no effect on the given correspondence and will automatically return `undecided`.

As illustrated in Section 3.1.2, compounds are a very productive means of word formation. Compound-head correspondences occur astonishingly often in real-world scenarios, and thus the Compound Strategy can discover many `is-a` correspondences in a mapping. However, different issues have to be regarded. First of all, not every correspondence *(A, B)* in which A ends with B or B ends with A expresses a compound-head case, as in *(plant, ant)*. In this case, *plant* is not a compound and *ant* is not its head, though an algorithm is unable to recognize this without further knowledge. We call such forms *pseudo compounds*, which (for a computer program) look like compounds, but are no compounds.

Moreover, not all compound-head correspondences express true `is-a` relations, which refers to the class of endocentric compounds, e.g., *computer mouse* (which is not a mouse) or *butterfly* (which is not a fly). Eventually, the Compound Strategy cannot determine all possible `is-a` relations, because of its strict focus on compound-head matches. For example, correspondences like *(car, vehicle)* or *(computer mouse, gadget)* are `is-a` relations that cannot be discovered by this strategy. For these reasons, the naive notion presented above was somewhat extended to mitigate those problems.

In the following, let $C = mH$ be a word that matches another word x and let m, H be sequences of characters. If C is a compound, it consists of the two words m, H, with m being the modifier and H being the head. Since any common noun, verb or adjective of the English language has a length of at least 3 characters, the following condition is used to tread C as a compound: $lenght(m) \geq 3, length(H) \geq 3$. Therefore, both head and modifier must have a length of 3 or greater. This simple rule is highly effective, because false compound-head matches like *(stool, tool)* or *(string, ring)* can be easily recognized. The specified rule does not hold for open and hyphenated compounds, though, because in this case modifiers can be indeed of length 1 or 2. Examples include *US president*, *e-book* and many technical terms (e.g., a *V-tail* is a specific tail of an aircraft and a *B-pillar* is a specific part of an automobile).

Some pseudo compounds like *(nausea, sea)* cannot be discovered with this rule, as the pseudo modifier *'nau'* has a length of 3. In this case, a dictionary look-up can help to corroborate that C is a compound, or to refute it. In particular, the words m, H have to be checked whether they are listed as separate words in a dictionary or word list. In the example of *(nausea, sea)*, the modifier *nau* is not a word in a dictionary and thus *nausea* would not be treated as a compound.

However, such a dictionary look-up raises new problems. First of all, there exist specific compounds in which the modifier is not an actual word and only appears in the given compound. The modifier is then a cranberry morpheme, and using the dictionary approach means to reject words although they are valid compounds (e.g., a *cranberry* is indeed a kind of berry, even though the word *cran* would not be found in dictionaries). Sometimes, also the modifier has changed its spelling so that it is not a real word any-

more, as in *holiday*. Finally, a dictionary or word list can never be complete and there may be more specific modifiers that are not contained in it. In this case, a correct compound would also be rejected. After executing some sample mappings, first with an English-language word list and subsequently without it, it became clear that such a look-up does not increase the precision of relation type determination, but impairs the recall. Thus, this method is not used in STROMA.

Even without dictionary look-ups, the Compound Strategy works very reliably and leads to a high precision. It is an interesting question, why this strategy usually achieves such reliable results, while there are many pseudo compound matches possible, and while there are many endocentric compounds that do not express an `is-a` relation. The answer to this question is that ontologies to be matched are usually from the same domain, or at least from two similar domains. In endocentric compounds, the head of the compound and the compound itself are usually from quite different domains, and the same holds for pseudo compounds. If a mapping would contain a correspondence *(buttercup, cup)*, a botanical ontology must have been matched with an ontology about furniture or kitchen items, which seems very unlikely. If a correspondence *(nausea, sea)* appears in a mapping, a diseases ontology must have been matched with a geographic ontology, which also seems unlikely. Therefore, most of such erroneous examples are usually prevented. Only in some specific cases they may indeed occur, e.g., if two general purpose ontologies are matched. Then it is possible that a pseudo compound like *nausea* refers to another concept *sea* and the Compound Strategy decides on `is-a`, although the whole correspondence is wrong in the first place.

The implemented strategy works for all kinds of compounds, so for open compound relations like *(high school, school)*, hyphenated compound relations like *(bus-driver, driver)* and closed compound relations like *(daybed, bed)*. Sometimes, it may happen that an attribute like an adjective or participle occurs before a noun. This construct will also be treated as a compound, although it is no compound from a linguistic point of view. Examples are *large-size screen* or *pre-owned vehicles*. Although these examples are no typical compound words, they also hold the `is-a` relation. A large-size screen is a specific screen (there are also mid-size or small screens) and a pre-owned vehicle is a specific type of vehicle (in contrast to a new vehicle). The Compound Strategy would also work in this case, since attributes occurring before a noun automatically make it more specific.

Regarding Further Forms of Word Formation

The Compound Strategy focuses on compound-head matches as illustrated before, but also checks for other compound-correspondences where the type is `equal`. This is the case if a compound matches an acronym or abbreviation that is apparently identical with the compound. For example, the Compound Strategy recognizes that "EU" and "European Union" are most likely identical, just as "NCI" and "National Cancer Institute" are most likely identical. Thus, the Compound Strategy is also able to determine `equal` relations, even though this is not the primary focus of the strategy.

Furthermore, the Compound Strategy regards prefix derivations. Using a manually created list of typical English prefixes, such as *con-*, *de-*, *inter-* or *pre-*, the Compound Strategy

is usually able to distinguish between derivations and compounds. As prefix-derivations are usually not in any `is-a` or `inverse is-a` relation with the base word, the Compound Strategy will return `undecided` for such a correspondence.

6.1.2 Compounds in Different Languages

The Compound Strategy works for the English and German language and was successfully used in English- and German-language mappings. In fact, compound formation follows the same scheme in any Germanic language, which comprises the north-Germanic languages such as Danish, Swedish and Norwegian and west-Germanic languages such as English, German, Dutch or Yiddish.

By contrast, the Romance languages such as Italian, French or Spanish have a different scheme for compound formation. In such languages, the head is often found at the beginning of the word (left), while the modifiers expand to the right. For example, the words *ordinateur portable* (French), *ordenador protátil* (Spanish) and *computer portatile* (Italian) refer to the English compound *portable computer*.

The implemented Compound Strategy in STROMA is unable to handle Romance-language compounds so far, but could be easily adapted to also process mappings of this group of languages. Analyzing compounds is hence a very generic, cross-linguistic technique to discover `is-a` and `inverse is-a` relations in a mapping.

6.1.3 Relations between Compound and Modifier

So far, the relation between a compound and its head has been analyzed, and because of the nature of compounds an `is-a` or `inverse is-a` relation is concluded. One question remains, whether there is a relation between the modifier m of the compound and the compound C itself, as in *(bedroom, bed)* or *(desktop, desk)*. In some cases, there is indeed a `part-of` or `has-a` relationship between modifier and compound, as a roof window is part of the roof or a computer screen is part of a computer.

In fact, there are 2 different cases. It can hold C `part-of` m or C `has-a` m (resp. m `part-of` C). There is linguistic evidence for both cases, though in some cases neither `has-a` nor `part-of` seems to hold. Analyzing the 500 most frequently occurring noun compounds of the English language, which occur among the top 12,800 words of the English language, there were 23.3 % compounds for which `part-of` holds and 30.7 % for which `has-a` holds (see Table 6.1). In 46 %, no reasonable `has-a` or `part-of` relation could be observed, though sometimes the type `related` can be assumed, as in *toothpaste – tooth* or *rainbow – rain*.

As a result of this analysis, there is no clear relation type between modifier and compound, as only in 54 % of all cases a semantic relation exists, and even then it can be either `has-a` or `part-of`. However, it is possible to use this strategy to suggest `has-a` or `part-of` relations within a mapping, which have to be manually checked (and either rejected or confirmed) by a user. As `part-of` and `has-a` types are generally very difficult to be

Case	Distribution	Examples
C `part-of` m	23.3 %	heartbeat, daylight, earring, policeman, moonlight, eyeballs, bathtub, clockwork, horseback, backbone
C `has-a` m	30.7 %	bedroom, motorcycle, railroad, fireplace, bookstore, graveyard, drugstore, sailboat, bowman, snowball
No relation	46.0 %	popcorn, spotlight, headline, billboard

Table 6.1: Examples and distribution of the different compound-modifier match cases.

found, such a semi-automatic approach could be very useful. By default, this compound-modifier-match-strategy is disabled in the standard configuration of STROMA, but can be enabled before mapping enrichment is carried out. It is also possible to select the correspondences where this strategy has determined any result, thus facilitating the manual verification.

6.2 Background Knowledge

Using background knowledge like thesauri is an effective way to determine the relation type between two concept names. STROMA uses multiple background knowledge sources, which contain both proprietary resources like WordNet or UMLS and automatically generated resources (Wikipedia relations). These resources are combined in a semantic repository called SemRep, which is independent from STROMA and works as a black box in the background. The SemRep API can be queried by STROMA at any time and simply obtains two words A, B and the language of the input ontologies. SemRep calculates and returns the relation type that holds between the two words, though it can also return `undecided` if the type cannot be ascertained.

Besides the Word Frequency Strategy, the exploitation of background knowledge is the only non-generic strategy used in STROMA. It can resolve many correspondences where linguistic strategies reach their limits, e.g., *(student, part-of, school)* or *(car, is-a, vehicle)*, but often fails if a specific term in a concept name is not contained in the repository or two related terms are not linked in the repository. Another drawback of background knowledge is its high time complexity, especially if indirect paths between terms are searched. Still, as we will illustrate in the evaluation, background knowledge can improve mappings significantly and can also be effectively combined with other STROMA strategies.

I will discuss the selection, integration and exploitation of background knowledge in the context of STROMA in more detail in Part III. This also includes the issue of generating background knowledge automatically from the web. For now, we will treat the Background Knowledge Strategy as the aforementioned black box, neglecting the details and functionality therein.

6.3 Itemization Strategy

6.3.1 Problem Description

Though many ontology concepts are described by a single term, e.g., *Student* or *Conference Venue*, there are ontologies whose concepts are more complex and are described by several nouns. We call such concepts *itemizations*, and examples include concept names such as *laptops and notebooks* or *motorbikes, scooters and pedelecs*. Itemizations need a special treatment, because they contain more information than their counterpart, which we call *atomic concepts*. The Itemization Strategy is used to determine the relation type of correspondences where at least one concept is an itemization, while it returns `undecided` if a correspondence consists of two atomic concepts. Differentiating between itemizations and atomic concepts is also an important step to avoid false conclusions based on pseudo compounds. For instance, in the correspondence *(cross bikes, bikes)* containing two atomic concepts, the Compound Strategy would correctly suggest an `is-a` relation. However, in the correspondence *(scooters and bikes, bikes)*, the Compound Strategy would be misled. It would squarely suggest an `is-a` relation, which is false, because *scooters and bikes* is no compound word, but an itemization. In point of fact, *scooters and bikes* has a larger semantic scope than *bikes*, because the concept refers to both scooters and bikes. This means that the `inverse is-a` relation holds in this instance.

6.3.2 Approach

To handle correspondences with itemizations, the first step is to build the *item sets* for each concept name. An item set is a complete set of all nouns that occur in the itemization, and it is created by splitting the concept name at the commas, slashes, ampersands (&) as well as the words *and*, *or*. For example, the item set of the concept *motorbikes, scooters and pedelecs* is {motorbikes, scooters, pedelecs}. This simple method practically works in 100 % of all cases, because there are only a few possible separators found in itemizations. Subsequently, the algorithm works as follows: Within four separate steps, it tries to gradually remove items from the two item sets that are either redundant (synonyms) or that are semantically included in another item (hyponyms). For instance, the item set *{laptops, notebooks}* can be reduced to either *{laptops}* or *{notebooks}*, as the two terms are synonymous. The reduced item set has still the same semantic scope as before. In another example, the item set *{computers, laptops}* can be reduced to *{computers}*, without influencing its semantics either. The implemented algorithm tries to remove as many items within the two item sets as well as between the two item sets. Finally, some simple methods of semantic reasoning are applied to determine the item set having the largest semantic scope. Ideally, the Itemization Strategy can reduce an item set to the empty set, which then makes it easy to determine the relation type of the correspondence. Otherwise, advanced techniques have to be used or the relation type cannot be determined.

In detail, the following steps are carried out:

1. **Intra-Synonym Removal.** In each item set I replace the items $i_1, i_2 \in I$ by i_1 if i_1 and i_2 are synonyms.
2. **Intra-Hyponym Removal.** In each item set I remove an item $i_1 \in I$ for which there exists a hypernym $i_2 \in I$.
3. **Inter-Synonym Removal.** Remove each item $i_1 \in I_1$ and $i_2 \in I_2$ if i_1 and i_2 are synonyms.
4. **Inter-Hyponym Removal.** Remove each item $i_2 \in I_2$ if there exists a hypernym $i_1 \in I$ and vice versa.

The first two steps only work within each of the two item sets. In the first step, synonym pairs (i_1, i_2) are replaced by one of the two synonyms, which will be i_1 by default. This makes the item sets smaller without losing any information. For instance, if a concept *computers, laptops and notebooks* is given, step 1 will replace the two synonym concepts *laptops, notebooks* by *laptops*, and hence the item set becomes smaller without impairing its semantics (the item set has still the same semantic scope as before). Synonyms within a concept name occur not very often, but as the above example shows, are generally possible.

The second step removes items in each item set to which a hypernym within the item set exists. This is another form of redundancy, because the hypernym already expresses what the hyponym expresses. In the above example, the item set was already reduced to *computers, laptops*. Now, the algorithm removes the item *laptops*, because the hypernym *computers* already includes the concept *laptops*.

Synonyms and hyponyms within a concept name clearly constitute a form of redundancy and may suggest a bad design of the taxonomy or ontology. However, it is quite natural in ontologies, and especially in product catalogs, to provide different terms for users. For example, a user searching for laptops in an online product catalog may look for this exact term (and might miss the relevant entry if it is labeled *Computers* or *Notebooks*). For this reason, synonyms and hyponyms within a concept name are not in the least exceptional, and the first two steps of the algorithm can already make the item sets much smaller.

In steps 3 and 4, actions refer to both item sets at the same time. In step 3, two items $i_1 \in I_1$ and $i_2 \in I_2$ are removed if they are synonyms. This means that the semantics of each item set is impaired, but since it influences both item sets in the same way, this step has no negative impact on the overall relation type determination. Consider the two item sets *{cars, bikes}* and *{automobiles}*. Inter-synonym removal means that the two items *cars* and *automobiles* are removed, because they are synonymous. Consequently, the item sets *{bikes}* and $\emptyset$ remain and it may already become obvious that the first item set expresses something more than the second item set, which is already empty. This conclusion would lead to the decision that the `inverse is-a` relation must hold.

Finally in step 4, hyponyms of each item set are removed if there is a hypernym in the opposite item set. For example, if there are two item sets *{books}* and *{novels}*, the concept *novels* would be removed from the second item set, as novels are books.

After the four steps have been carried out, each item set has been reduced to its minimal possible size. To determine whether items are in a synonym or hyponym relation, which is a significant part of this strategy, is achieved by aid of the previously introduced strategies Compound and Background Knowledge and on lexicographic comparison (e.g., two items i_1, i_2 are obviously synonyms if they are equally spelled). Therefore, the techniques of the previously introduced strategies are effectively re-used by the Itemization Strategy, but this also means that the quality of this strategy partly depends on the quality of the Compound and Background Knowledge Strategy.

In the following, the Itemization Strategy will determine the relation type based on what item sets remain after the removal steps. There are 5 distinct cases to be regarded:

1. $I_1 = \emptyset, I_2 = \emptyset$
2. $I_1 = \emptyset, I_2 \neq \emptyset$
3. $I_1 \neq \emptyset, I_2 = \emptyset$
4. $|I_1| = 1 \wedge |I_2| = 1$
5. $|I_1| > 1 \vee |I_2| > 1$

Case 1. If both item sets are empty, then both item sets express the same thing and the relation type `equal` is assumed.

Cases 2 and 3. If I_1 is empty, but I_2 is not empty, I_2 expresses something more than I_1. The `is-a` type is concluded. If I_2 is empty, but I_1 is not, the argumentation is analogous and it holds the relation type `inverse is-a`.

Case 4. In this case, two items remain which are obviously not in an `equal`, `is-a` or `inverse is-a` relation. The Background Knowledge Strategy will check whether there is a `part-of` or `has-a` relation between the two items. If no type can be determined, `undecided` will be returned.

Case 5. If no item set is empty and at least one item set contains more than one item, the relation type cannot be calculated and `undecided` will be returned. There is also a different approach that might be followed, in case that it holds $|I_1| > |I_2|$ or vice versa. It could now be assumed that the item set which holds more items is more general than the item set which holds less items, however, this assumption is not always correct. Evaluations of mappings containing various itemizations showed that this heuristic rule is too vague and that it may also hold the opposite case, as in the remaining item sets {Car} and {SUV, Pick-up, Jeep}. In this case, the second item set containing 3 items is still more specific than the first item set containing just the item *Car*. Besides, this approach could not reasonably handle the case $|I_1| = |I_2|$ either. Therefore, this heuristic is not used in STROMA and `undecided` will be returned in case 5.

Analyzing the implemented algorithm more precisely, it becomes obvious that meronyms and holonyms are not regarded. Although this would be generally possible, the algorithm will ignore meronym and holonym relations, mainly for the reason that the laws of set theory do not hold for `part-of` relations. If a *keyboard* is part of a *computer*, it does not

mean that it is a computer and that it can thus be removed without any loss of information, but that the two concepts are rather independent, with the only exception that they often occur together. Removing meronyms can indeed impair the overall result. Consider the two concepts *computers* and *computers & keyboards*. If meronyms are removed just as hyponyms, the result of the algorithm would be two empty sets, as *keyboards* are part of *computers* and are removed, and subsequently the two remaining items *computers* are removed. Therefore, the relation type `equal` would be suggested. However, it seems more reasonable that the concept *computers & keyboards* expresses semantically more than the concept *computers*, which suggests the `inverse is-a` relation (which would be also concluded with the current implementation).

In other cases, there might be a clear dependency between meronym and holonym, as in the example *(door handle, door)*. As a *door handle* is a specific part of a door, removing this item would not influence the semantic scope. In this case, it would be necessary to remove *door handle*, because it is also included in the concept *door*. However, it is completely impossible to decide whether a given meronym depends on its holonym (and thus should be removed) or whether it occurs rather independently (and should not be removed, just as in the *keyboard* example). Therefore, meronyms and holonyms are not regarded by the Itemization Strategy so far.

To conclude this subject, we will provide an overall example that shows how the strategy works in general. The following sample correspondence from the media domain consists of 3 resp. 4 items:

$$\{novels, periodicals, magazines\} \Leftrightarrow \{books, periodicals, e\text{-}readers, e\text{-}book\ devices\}$$

In step 1, the synonyms within each item set are removed. The terms *e-readers* and *e-book devices* are synonyms and will be replaced by the first word, *e-readers*.

$$\{novels, periodicals, magazines\} \Leftrightarrow \{books, periodicals, e\text{-}readers\}$$

In step 2, the hyponyms within each item set are removed. In the first item set, *magazine* is a hyponym of *periodicals* and will be removed.

$$\{novels, periodicals\} \Leftrightarrow \{books, periodicals, e\text{-}readers\}$$

In step 3, synonyms between the two item sets are removed. The two terms *periodicals* are synonymous and are removed.

$$\{novels\} \Leftrightarrow \{books, e\text{-}readers\}$$

In step 4, hyponyms in each item set are removed if there is a hypernym in the opposite item set. The term *novel* is a hyponym of *book*, so that it is removed from the first item set.

$$\{\ \} \Leftrightarrow \{books, e\text{-}readers\}$$

As a result, the second item set is more general than the first item set and the strategy would return `is-a` (case 2).

6.3.3 Two-ways Adaptation

The introduced workflow is a form of bottom-up approach, in which first hyponyms and synonyms within each set are handled and afterwards hyponyms and synonyms between the item sets are handled. There is also a top-down approach, in which first Inter-Hyponym and Inter-Synonym treatment takes place and afterwards Intra-Hyponym and Intra-Synonym treatment.

In its original form, the Itemization Strategy was a sole bottom-up strategy. However, this approach is not optimal, as the following example illustrates:

science fiction novels $\Leftrightarrow$ books and novels

The bottom-up approach starts with the intra-operations. The right-hand concept consists of the two terms *books* and *novels*. Using background knowledge, STROMA discovers that novels are a specific form of books so that the two item sets {*science fiction novels*} and {*books*} remain. However, the term *science fiction novel* is not contained in the background knowledge repository and the two item sets cannot be further reduced. STROMA is unable to resolve the relation between *science fiction novels* and *books* and returns `undecided` for the given correspondence.

It is quite natural that some correspondences are too difficult for STROMA to handle, but a closer look at the above correspondence reveals that there is a compound-head match *(science fiction novels, novels)*. Obviously, the right side expresses more than the left side, because science fiction novels are specific forms of novels.

Now, the top-down approach starting with the inter-operations would immediately remove *science fiction novels* after recognizing the compound-head relation to *novels*. The remaining item sets would be { } and {*books, novels*}, and STROMA would correctly decide on an `is-a` relation.

Since a sole top-down approach is just as inappropriate as a sole bottom-up approach, the Itemization Strategy was adapted as follows: First, the Itemization Strategy is carried out using the bottom-up approach. If it does not yield any specific relation type, the top-down approach is followed. It uses the same input as the bottom-up approach, so the item sets are reset before the strategy is run again, in order to regain the original starting conditions. Using this two-ways-approach, about 5 – 10 % more itemization correspondences can be successfully handled.

6.4 Structure Strategy

The previously introduced strategies only concentrate on the leaf concepts of correspondences, i.e., they do not take the correspondence path into account. Sometimes, determining the relation type between two leaf concepts (two words) is very difficult or even impossible. In this case, it can be very helpful to also regard the parent concepts of the leaf concepts and to use some semantic deduction to determine the relation type. Given

two leaf concepts X, Y, the Structure Strategy is able to derive the relation type indirectly by regarding the parent concepts X' resp. Y' of the two concepts. This implies that the Structure Strategy can be only executed if at least one of the two concepts X, Y has a superior concept (parent concept). However, this is normally the case in taxonomies and ontologies. Additionally, X, Y and X', Y' must be atomic concepts.

Fig. 6.1 illustrates this strategy. Instead of determining the relation between X and Y (a), the Structure Strategy determines the relation between X and Y' (b) or Y and X' (c) and tries to entail the relation type indirectly. The relation type between X and Y is then resolved in the following way:

1. X `equal` $Y' \rightarrow X$ `inverse is-a` Y
2. X `inverse is-a` $Y' \rightarrow X$ `inverse is-a` Y
3. Y `equal` $X' \rightarrow X$ `is-a` Y
4. Y `inverse is-a` $X' \rightarrow X$ `is-a` Y
5. X `part-of` $Y' \rightarrow X$ `part-of` Y
6. Y `part-of` $X' \rightarrow Y$ `part-of` X

These conclusions are based on the assumption that sub-concepts in an ontology are more specific than their parent concepts. The Structure Strategy thus makes advantage of the intra-ontology structure and transitivity of the relation types. In case 1, where X and Y' are equivalent, the relation type `inverse is-a` must hold, because anything under Y' can be assumed to be more specific than Y' and thus more specific than X. The same holds if X is more general than Y', which makes it even more general than a sub-concept of Y' (case 2). These two cases refer to Fig. 6.1 b).

By contrast, if Y is equal to X', it is more general than the sub-concept X and thus the `is-a` relation holds (case 3, also illustrated in Fig. 6.1 c). The same argument holds if there is an `inverse is-a` relation between Y and X' (case 4). Cases 5 and 6 are similar to cases 2 and 4 and refer to the `part-of` relation. For example, if X is part-of Y', it can be assumed that X `part-of` Y' is a typical property of Y' and that it also holds for its sub-concepts. We will discuss such indirect relation type calculation in more detail in Section 9.3.3.

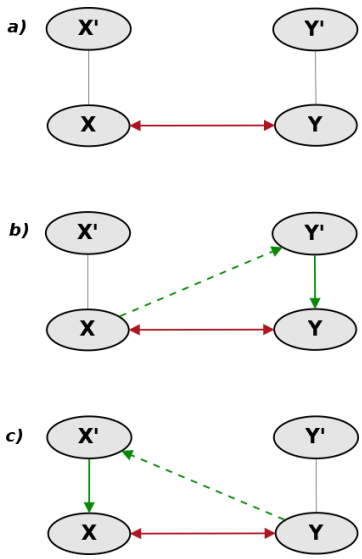

Figure 6.1: The basic notion of indirect relation type determination.

An example is illustrated in Fig. 6.2. Let us assume that there is a correspondence *(furniture, nightstands)* which cannot be resolved by any other strategy. Structure Strategy will now compare *furniture* and *bedroom furniture*, and by aid of the Compound Strategy, conclude *furniture* `inverse is-a` *bedroom furniture*. The sub-concept *nightstands* is obviously more specific than *bedroom furniture* and thus it holds *(furniture, inverse is-a, nightstands)* (case 2).

Thus, the Structure Strategy is able to determine any relation type except for `related`. To determine the relatedness between X and Y' resp. X' and Y, this strategy uses other strategies such as Background Knowledge and Compound Strategy. There are few cases in which this strategy cannot determine any type. These four cases are:

1. X `is-a` Y'
2. Y `is-a` X'
3. X `has-a` Y'
4. Y `has-a` X'

I will only discuss the first and third case, since the other cases are analogous. In the first case, it holds X `is-a` Y' `inverse is-a` Y, and having only these pieces of information, the relation between X and Y cannot unequivocally be determined. In fact, it could hold either of the three cases: X `equal` Y, X `is-a` Y and X `inverse is-a` Y. To illustrate this using an example, let us assume that the concept X is *laptop* and Y' is *computer*, which constitutes a typical `is-a` relation. Now, any of the following concepts could be Y: *notebook*, which is in an `equal` relation to *laptop*, or *netbook*, which means an `is-a` relation, or *personal computer*, which means an `inverse is-a` relation. As the type cannot be determined this way, Structure Strategy would return `undecided`.

In the third case, there is a `has-a` relation from X to Y', e.g., *(university, student)*. However, sub-concepts of *student* are not necessarily part of a university. For instance, the sub-concept *high-school student* is not in a `part-of` relation to university. This makes cases 3 and 4 too vague to entail any relation type.

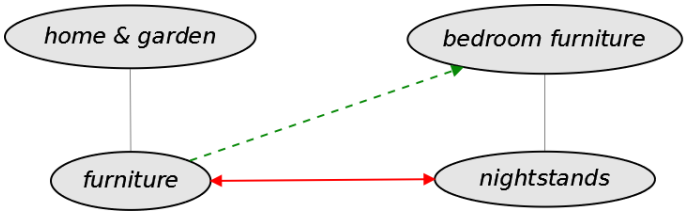

Figure 6.2: Sample correspondence for indirect relation type determination.

6.5 Multiple Linkage

Occasionally, if two ontologies O_1, O_2 are matched, one of the two ontologies is considerably larger than the other, i.e., it contains many more concepts. If the two ontologies refer to the same domain, it can be assumed that one ontology is more specific than the other, so that a high-level ontology was matched with a more specific ontology. STROMA does not possess any information about the input ontologies and their sizes, nor whether they refer to the same domain or not, but such a special situation can be also determined by analyzing the correspondences in a mapping itself. If there is a concept $c \in O_1$ which takes part in correspondences to several concepts $x_1, x_2, ..., x_n \in O_2$, it can be assumed that c is more general than any of the $x_i \in O_2$ and that for each correspondence the `inverse is-a` relation holds.

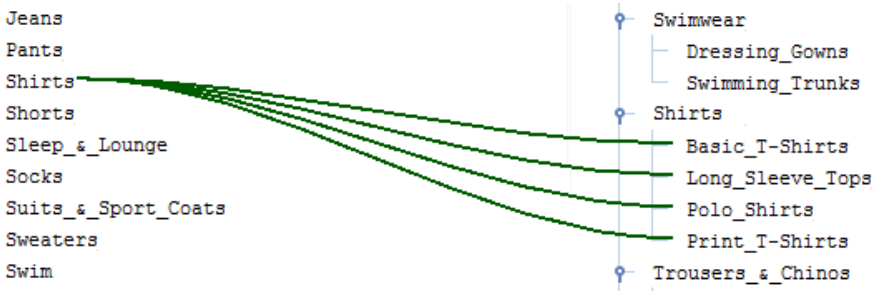

Figure 6.3: Sample mapping where the Multiple Linkage Strategy can be used.

This is a mere heuristic which does not use any semantic knowledge hidden in the mapping, but it achieves good results in most cases. An example is given in Fig. 6.3, where two taxonomies about clothing are matched, the right one (O_2) being much more specific than the left one O_1. Thus, the concept *Shirts* in O_1 is considered to be more general than the concepts in O_2 and each correspondence would be denoted by an `inverse is-a` relation.

Still, there are some issues related to this strategy. First of all, the initial mapping may have a low precision, meaning that c is erroneously connected to the concepts x_i. If it turns out that only one element of the x_i is actually in a correct relation to c, the strategy would be entirely misled and any relation type might be possible then. For this reason, STROMA uses a threshold parameter that specifies under which circumstances Multiple Linkage can be used. By default, a node must be involved in at least 3 correspondences so that this strategy is run, which achieved the best results in different experiments. This parameter is flexible, however. A higher value would potentially increase the precision, but naturally reduce the recall.

Furthermore, the concepts x_i could also be in a `part-of` or `has-a` relation to c, e.g., *(head, face)*, *(head, brain)*, *(head, mouth)*, etc. Though `part-of` relations do not occur very often, Multiple Linkage would return the wrong type in this case. There is no trivial technique to prevent such false conclusions.

Finally, one of the correspondences (c, x_i) could also be an `equal` relation, for instance if this x_i is a parent concept of the other concept x_j $(j \neq i)$. This case is illustrated in Fig.

6.4, where the concept *Shirts* in O_1 matches an additional concept *Shirts* in O_2. Normally, schema and ontology matchers would rather focus on leaf-to-leaf correspondences and omit correspondences between inner nodes (as data is normally stored at the leaf concepts), but such correspondences are still possible and can be the result of an advanced match configuration that also calculates inner correspondences. In this case, Multiple Linkage uses two techniques to mitigate this problem. First of all, the spelling of the two concepts is taken into account, and in the given example Multiple Linkage would quickly decide on an `equal` relation, as both concepts are obviously the same. However, even if the spelling would be different and the two concepts are synonyms (like Tee-Shirt and T-Shirt), Multiple Linkage would discover the `equal` relation, as it also analyzes the structure of the multiple concepts in O_2. In this case, it recognizes that *Shirts* (O_2) is the parent element of all other concepts in O_2. Since there is no other correspondence between *Shirts* (O_1) and the five concepts in O_2, it can be assumed that these 5 concepts have the same semantic scope as *Shirts* in O_1. Consequently, since *Shirts* (O_2) is the top level element of the multiple concepts, it comprises the meaning of all child concepts and its semantics can be assumed to be equal to *Shirts* in O_1. Therefore, `equal` would be assumed.

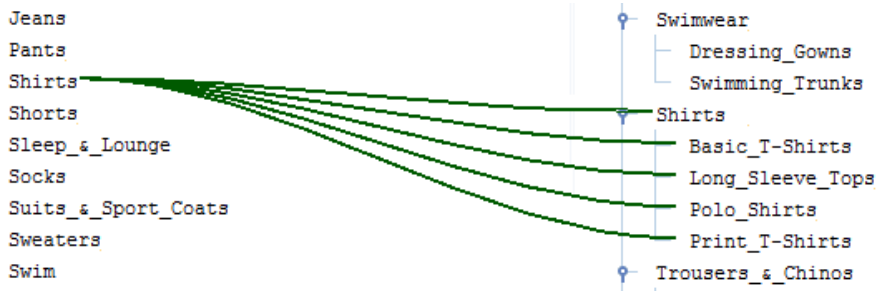

Figure 6.4: Extended sample mapping where one correspondence is of type `equal`.

Although Multiple Linkage achieves generally very good results, a somewhat lower confidence value is used for this strategy, as it is based upon heuristics after all (see Section 5.2.2).

6.6 Word Frequency Strategy

A word frequency list specifies the number of word occurrences within a given text corpus. It is usually ordered by frequency, starting with the most-frequently occurring word, yet it can also be ordered alphabetically. If the text corpus is large enough and is not restricted to any particular domain (like historic novels or scientific publications), it is a representation of the general usage of words in the given language, i.e., it reveals how frequently words are used. The domain of a text corpus used to build word frequency lists is also called *genre*. Typical genres to build word frequency lists are newspapers, novels, lyrics, websites or movie subtitles.

The Word Frequency Strategy implemented in STROMA exploits such word frequency lists as another form of external knowledge and makes the assumption that more frequently used words are more general than less frequently used words. Consider the word *book*. This is a very general term and occurs frequently in present English texts, no matter which genre is regarded. By contrast, the more specific terms *novel* and *cookbook* occur less frequently, as they are more special terms. The gist of the Word Frequency Strategy is to suggest an `is-a` resp. `inverse is-a` relation between given terms a, b if one of the two terms is much more frequent than the other terms. This works in many cases, e.g., in the chains *computer – laptop – netbook* or *artist – musician – composer*. Although there are many examples where this assumption fails (see Section 6.6.2), the Word Frequency Strategy is still able to increase the mapping quality and can suggest the correct type where other strategies fail.

6.6.1 Implementation

STROMA uses publicly available word frequency lists that were built from movie subtitle texts. These lists comprise a vast amount of words, are provided for many languages (including German and English) and can be downloaded free of charge under a Creative Commons License.[48] Those movie-based word lists are rather up-to-date, and because of the diversity of movies, are suitable for almost all kinds of mapping scenarios, except for very specific domains like engineering or chemistry. Further freely available word frequency lists can be found on Wiktionary[49], but they are either too small or not up-to-date anymore. For example, the Gutenberg word list is based on freely available e-books collected by the Project Gutenberg, which were usually published before 1923. Commercial resources are also provided by institutions like the Linguistic Data Consortium[50] or the Evaluations and Language resources Distribution Agency (ELDA)[51], but are typically used for much more specific linguistic research. There are also free corpora for the development of individual word lists, e.g., the IMS Open Corpus Workbench (CWB)[52] or the Opus Corpus presented in [147].

The Word Frequency Strategy contains a frequency list for the English language and for the German language. Both lists are loaded from a text file when STROMA is launched. The strategy first checks whether the input words a and b are identical by stemming both words. If they are identical, `equal` is returned; otherwise, STROMA checks whether the given words are suitable to carry out the Word Frequency Strategy.

The following cases can occur:

1. The words a and b are no open compounds. In this case, the strategy can be run without any preprocessing.

[48] `https://invokeit.wordpress.com/frequency-word-lists/`
[49] `http://en.wiktionary.org/wiki/Wiktionary:Frequency_lists`
[50] `https://www.ldc.upenn.edu/`
[51] `http://portal.elda.org/en/about/elda/`
[52] `http://cwb.sourceforge.net/index.php`

2. The word a is an open compound, but b is no open compound (or vice versa). Then a needs to be preprocessed as described below, while b remains unchanged.
3. Both a and b are open compounds. The strategy will not be carried out then and `undecided` is returned.

Since word frequency lists do not contain open compounds, open compounds need a special form of preprocessing. In this case, the compound word a is reduced to its head a_H. The Compound Strategy then compares a_H and b instead of a and b. This preprocessing is only effective if at least one of the two words is no open compound; otherwise, no result can be determined (case 3).

STROMA is able to determine `is-a` and `inverse is-a` relations in the following way:

- $rank(a)/rank(b) > t$: `is-a`
- $rank(b)/rank(a) > t$: `inverse is-a`

In all other cases, the Word Frequency Strategy only returns `undecided`. It is thus restricted to the two types `is-a` and `inverse is-a`. The threshold $t > 1$, which we call *Minimal Word Frequency Quotient*, can be freely configured. If no preprocessing is used, a value $t = 8$ turned out to be the most effective one. A larger value increases the precision, but reduces the recall. A lower value decreases the precision, but increase the recall.

If preprocessing was used on term a, a lower threshold is necessary, i.e., $t > 8$ is too strict for such a case. Since a is a compound word expressing something more specific than a_H, it can be assumed that a is even then more specific than b if a_H and b have nearly the same rank. In fact, the best results were achieved if a value of 1 was used. For example, the two synonymous words *country* and *state* have a similar rank and a minimal word frequency quotient of 1.43. STROMA would return `undecided` in this case. However, if the two forms *country* and *US state* are used, STROMA returns *US state* `is-a` country, because the index is above the threshold of 1.

The Word Frequency Strategy is also used by the Itemization Strategy and Structure Strategy to find hypernym and hyponym relations. As shown in the evaluation, it can have a considerable impact on the mapping quality.

6.6.2 Obstacles

Because of the high complexity of languages in use, the Word Frequency Strategy remains a mere heuristic. In many cases, top-level concepts exist that comprise many other concepts and are thus very generic. However, since humans often attempt to express themselves specifically, such very generic terms are sometimes used less often than the more specific terms and thus have a lower rank in word frequency lists. For instance, the term *vehicle* is very generic and comprises cars, ships, planes, bikes, trains and many more concepts. However, each of the itemized concepts has a higher rank in the word frequency table than *vehicle*, because *vehicle* is rarely used in everyday language. Thus,

sentences in corpora would rather look like "I caught a train to..." instead of "I caught a vehicle to...". The difference between synonyms can also be pretty big, e.g., plane (#916) is much more frequently used than aircraft (#4662), yet is no specific form of an aircraft, but a synonym.

Word frequency lists are not optimal for the purpose of this research because of three reasons: First, they are not lemmatized, i.e., all inflections of a word can appear in the list (e.g., *watch*, *watches*, *watching*, *watched*), although only information about the total number of noun occurrences are needed, i.e., *watch* including the plural form *watches*, yet without the third-person verb form *watches* and the progressive form *watching*. Secondly, frequency lists do not contain open compound words, e.g., there is no entry for *city hall*, yet there are just the two entries *city* and *hall*. Eventually, a good deal of words within a language can be used in different word classes. For instance, the word *to watch* is a very frequently used verb and has a very high rank. The noun is less frequently used, but as it is identical with the infinitive form of *to watch*, it shares this high rank. In fact, if the words *clock* and *watch* are compared, *watch* has a higher rank and appears to be more generic, although the opposite is the case. The same argument holds for words from specific domains. For instance, in the biological domain the terms *ring* and *chain* describe specific types of *structures*, although those two terms are used frequently in everyday language and for this reason have a much higher rank than *structure*.

Because of the many drawbacks and heuristic assumptions made, the Word Frequency Strategy has the lowest confidence value. That is, other strategies immediately outvote the result of this strategy if they determined any other type. In this configuration, the Word Frequency Strategy is most effective, as it can determine the correct type where all other strategies fail, but cannot impair the results determined by the other strategies.

6.7 Type Verification

After each strategy has been executed and STROMA has calculated a relation type, the final type is once more checked for validity, which is part of the type verification. This verification becomes necessary, because the strategies only regard the relation between the matching path leaf concepts, yet the relation type may also depend on the internal structure of the concepts within the ontology (concept path). According to evaluation results, `is-a` relations are especially susceptible for such erroneous decisions, but also `equal` and `part-of` relations can be erroneous because of the specifics of the ontology structures.

The verification is carried out by the verification module, the so-called verificator of STROMA. Each enriched correspondence is passed to this verificator, which comprises several verification techniques for each correspondence type. Correspondences of type `related` are not further verified, though, as there are no plausible techniques to prove or disprove such a co-hyponym relation.

Verification only becomes possible if at least one of the two matching concepts has a parent concept and the concepts are atomic. Therefore, the verificator has the same preconditions as the Structure Strategy.

6.7.1 Verifying Is-a Relations

In some cases, an obvious `is-a` correspondence like *(Children shoes, `is-a`, Shoes)* turns out to be an `equal` relation if the overall correspondence path is analyzed:

Clothing.Children_shoes $\leftrightarrow$ *Clothing.Children.Shoes*

This correspondence is obviously an `equal` correspondence, because both concepts express the same thing (shoes for children). However, several strategies (e.g., the Compound Strategy) would have suggested an `is-a` relation, because the two concepts *Children shoes* and *shoes* are in an `is-a` relation. The `is-a` verification investigates the parent element of each leaf concept and tries to figure out whether the `is-a` relation seems justified or whether an `equal` relation might be correct.

In a correspondence between concepts X, Y, let X' resp. Y' be the parent concept of X resp. Y. The verificator uses several methods to proof or disproof that either $(X' + X) = Y$ holds or $(Y' + Y) = X$. The operator $+$ can be interpreted as "combined" or "concatenated" and only serves for the illustration of the problem. In the following, we will focus on the case $(Y' + Y) = X$, as in the above example, but the argumentation for the other case is the same.

The first and simplest method is to concatenate Y and Y', put a space, underscore or hyphen in between and check whether it matches X. This would already work in the above examples. It holds $Y' = Children$, $Y = Shoes$ and the concatenation of the two terms, including a space character, yields the term $X = Children\ Shoes$.

There are much more complex scenarios conceivable, though. For instance, let us assume that the source concept is not *Children Shoes*, but *Kids shoes*. In this case, background knowledge is used to discover the equivalence between the two concepts. As it will be shown in Part III, SemRep is able to discover an `equal` relation between *Children shoes* and *Kids shoes* because of the equivalence between *Children* and *Kids*.

Furthermore, let us assume the source concept would not be *Children shoes* or *Kids shoes*, but *Shoes for children*. To some degree, STROMA is able to handle even such a case and recognize the equivalence between the two concepts. In this case, the concepts Y' and Y are concatenated again and a lexicographic matcher calculates a similarity value between $(Y' + Y)$ and X. This matcher uses the similarity measures *Trigram*, *Jaccard Distance* and *Jaro-Winkler Dinstance* [133]. If the similarity value is above a specific threshold, the concepts are considered to be equivalent. In the above case, the similarity between *Shoes for Children* and *Children Shoes* would be high enough for STROMA to consider the correspondence as `equal` relation.

Such a verification is quite effective and can boost the mapping quality to some degree. However, it is important to avoid any false conclusions, so to prevent an erroneous type

change of an originally correct `is-a` relation. For instance, the following correspondence is a true `is-a` correspondence, but looks very similar to the correspondence above:

Clothing.Baby_shoes ↔ *Clothing.Children.Shoes*

However, STROMA would recognize that this is no `equal` relation and will not change the type, because SemRep would not confirm any `equal` relation between *baby* and *children* resp. *baby shoes* and *children shoes*. Besides, the lexicographic overlap between *Baby shoes* and *Children Shoes* is too low and no equality relation would be assumed.

6.7.2 Verifying Equal-Relations

Similar to `is-a` relations, `equal` relations can be misleading as well, as the following example shows:

Clothing.Children.Shoes ↔ *Clothing.Shoes*.

Although the relation between the leaf concepts *Shoes* and *Shoes* is apparently `equal`, the concepts are in an `is-a` relation, because the left concept refers to children shoes, while the right concept refers to shoes in general.

To discover such a pitfall is more difficult and more prone to errors than the discovery of false `is-a` relation. Let $X_1, X_2, ..., X_m$ be the concepts in the source path and $Y_1, Y_2, ..., Y_n$ the concepts in the target path with X_i, Y_j being the specific concepts within the path and $X_1 = Y_1$ being the leaf concepts of the ontology. It obviously holds X_1 `equal` Y_1, as the correspondence was denoted as an equal-correspondence.

The implemented approach turns the `equal` type into `is-a` if there are i, j with $i < m, j < n$ so that X_i `equal` Y_j and $i > j$. By contrast, it turns it into `inverse is-a` if it holds $j > i$. This method is called *Common Predecessor*.

In the above example, X_3 and Y_2 are obviously equivalent. It therefore holds $i = 3$ and $j = 2$ and thus $i > j$. The introduced strategy works in this case, as it can be quite naturally assumed that *Shoes* (X_1) must be more specific than *Shoes* (Y_1), because of the additional element *Children* (X_2) between the two *Clothing* elements (X_3, Y_2), which is also illustrated in Fig. 6.5 a). However, this technique fails in many cases and can both improve and reduce the overall result. An counterexample is depicted in Fig. 6.5 b), where the `equal` relation is correct. The strategy is impaired by the additional element *Shirts* in the left taxonomy, which is more fine-granular than the right taxonomy.

Experiments showed that the `equal` verification leads to worse average results. It is therefore not used in the default configuration, but can be enabled and disabled at any time.

6.7.3 Verifying Part-of relations

`part-of` relations are less prone to such errors than `is-a` or `equal` relations, though they are also possible, as the two following examples illustrate:

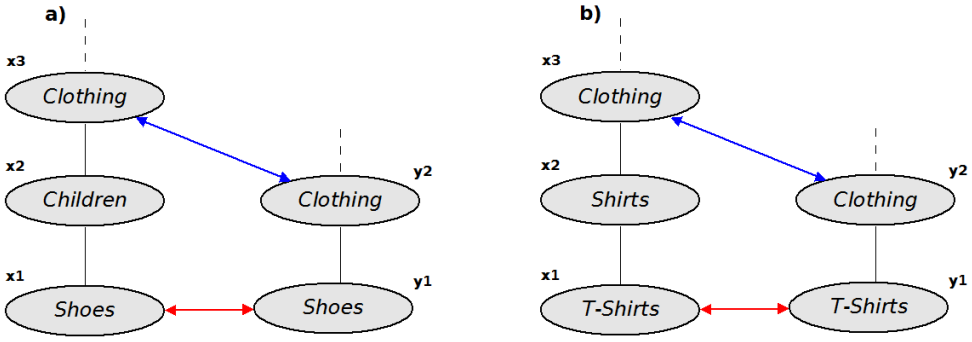

Figure 6.5: Two examples where the *Common Predecessor* technique works correctly (a) and where it fails (b).

$$\textit{Clothing.Zips} \leftrightarrow \textit{Clothing.Zips.Pants}$$

$$\textit{Home.Door\_handles} \leftrightarrow \textit{Home.Handles.Door}$$

Both correspondences are actually of type `equal`, i.e., they both express zips for pants resp. door handles, but only regarding the concepts, STROMA would decide on `part-of` relations. Such false `part-of` relations become possible if a specific concept can be part of different objects, as a zip can be part of pants, jackets, cardigans, etc. If the taxonomy does not follow the natural part-of hierarchy (*clothing – pants – zip* or *home – door – handle*), STROMA may decide on a `part-of` relation although it is an `equal` relation.

Since `part-of` relations occur less frequently than `is-a` or `equal` relations, such cases hardly occurred during the evaluations. Still, this problem was also addressed and requires a similar strategy as in the `is-a` verification. Instead of combining the last two concepts of each concept path, which would yield terms like *Zip Pants* or *Handles Door*, the terms are swapped, i.e., leaf concept and parent concept are concatenated instead of parent concept and leaf concept. Then again, the verificator tries to figure out whether the concepts are equivalent by using match techniques, as well as the Compound Strategy and Background Knowledge Strategy.

6.8 Strategy Comparison

STROMA consists of the six strategies presented in the previous sections, yet since some strategies have specific preconditions, not all strategies are executed for a given correspondence. For instance, Itemization Strategy and Structure Strategy mutually exclude each other, which means that at most five strategies can return a result. Table 6.2 provides an overview of the preconditions of each strategy. Correspondences where at least one concept is an itemization can only be processed by the Itemization Strategy, which, however, draws on the Background Knowledge and Compound Strategy. Correspondences containing atomic concepts can be run by all strategies except Itemization, but may have further preconditions. Multiple Linkage is the only strategy that has no for-

mal preconditions, but will only return a type if the correspondence at hand is part of an $(n:1)$-correspondence, with $n \geq 3$ (default configuration).

Strategy	Preconditions
Compound	Compound-head-match, no itemizations
Background Kn.	No itemizations
Itemization	At least one of the concepts is an itemization
Structure	No itemizations, at least one of the concepts has a parent element
Multiple Linkage	
Word Frequency	Both concepts must be contained in the word frequency list

Table 6.2: Overview of the preconditions of the strategies.

Fig. 6.6 shows the relatedness of the six implemented strategies. On top of the chart are the three general enrichment techniques background knowledge, lexicographic analysis and structural analysis (ellipses), as also used in the matching and mapping domain. Below are the six strategies (boxes) and their dependencies among each other. An arrow indicates that a strategy can also be used by another strategy, just as the Itemization Strategy uses the Compound Strategy and Background Knowledge Strategy to determine synonyms and hyponyms in item sets.

Background Knowledge and Compound Strategy are the so-called base strategies that can be universally used (either directly or indirectly by another strategy). The other strategies are advanced strategies that are used in more particular situations.

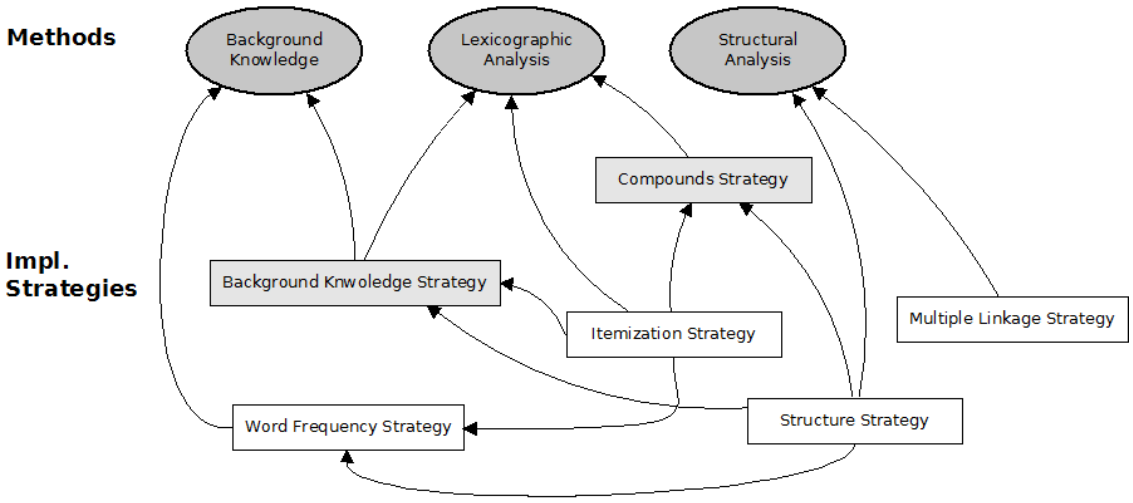

Figure 6.6: Dependency graph of the six implemented strategies.

Table 6.3 shows the different types that can be determined by each strategy. The Background Knowledge and Itemization Strategy are able to return any possible relation type. Though the Itemization Strategy disregards meronyms and holonyms and thus usually returns `equal`, `is-a` or `inverse is-a`, there is a specific case in which both remaining item sets consist of exactly one concept. Using background knowledge for these two items can

then lead to a relation type `has-a`, `part-of` or `related`, so that the Itemization Strategy can practically return all kinds of relation types. Structure Strategy can determine any type but `related`. The strategies Compound, Multiple Linkage and Word Frequency are limited to `equal` and `is-a` types. Nevertheless, these are the most frequently occurring types in common mappings.

Strategy	**equal**	**is-a + inv. is-a**	**has-a + part-of**	**related**
Compound	X	X		
Background Kn.	X	X	X	X
Itemization	X	X	X	X
Structure	X	X	X	
Multiple Linkage	X	X		
Word Frequency	X	X		

Table 6.3: Overview of the different types determined by each strategy.

7 Evaluation

In this chapter, we will evaluate the quality and performance of STROMA. Given an input mapping M and the enriched mapping ME, the primary focus of this evaluation is to gauge how many correspondences in ME have been correctly typed. Such an evaluation is very useful to assess the overall quality of STROMA and to outline the strengths and weaknesses of the implemented strategies. In Section 7.1, we will explain how the match quality of common match tools is usually evaluated, while we explain in Section 7.2 how these techniques can be extended to also evaluate the quality of enriched mappings. Such an evaluation is more intricate, because it has to regard both the match quality and relation type quality. In Section 7.3, we present the different gold standards (reference alignments) used to evaluate STROMA. In Section 7.4, we evaluate STROMA using perfect (correct) input mappings, allowing a more appropriate assessment of the general quality provided by the different strategies implemented in STROMA. By contrast, we use authentic (real) input mappings in Section 7.5 which shows the quality achieved within the full two-step approach (including matching *and* enrichment). We will discuss the time performance of STROMA in Section 7.6 and conclude this chapter with a comparing evaluation between the semantic match tools S-Match, TaxoMap and STROMA in Section 7.7.

7.1 Evaluating the Match Quality

In ontology matching, the quality and effectiveness of match tools is usually gauged by matching two input ontologies O_1 and O_2 and comparing the resulting mapping M with a benchmark mapping B, which is also referred to as *perfect mapping*, *gold standard* or *reference alignment*. Such benchmarks are generally user-defined, as no automatic tool can

reliably determine a perfect mapping. Simpler and smaller benchmarks can be usually developed by laymen within a limited amount of time, while domain-specific benchmarks often need the aid of domain experts. Crowd sourcing can be effectively used to create larger benchmarks which are more effective to assess the match quality [104]. Comprehensive overviews of mapping evaluation are presented in [16] and [47].

Given the mapping M and the benchmark B, two measures are used to estimate the quality of the match tool.

$$r = \frac{|M \cap B|}{|B|} \tag{7.1}$$

$$p = \frac{|M \cap B|}{|M|} \tag{7.2}$$

Equation 7.1 defines the recall r, which specifies the completeness of the mapping, i.e., how many of the correspondences in the benchmark were detected by the match tool. Equation 7.2 specifies the precision p of the mapping. The precision specifies how many of the detected correspondences are correct, i.e., how many of them occur in the benchmark. Correspondences in $M \setminus B$ thus diminish the precision while correspondences in $B \setminus M$ diminish the recall (see also Fig. 7.1).

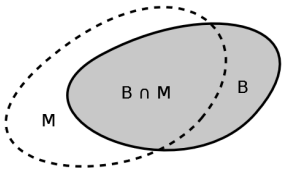

Figure 7.1: Illustration of the two overlapping mappings M and B.

Precision and recall are normally in an inverse relationship. If one measure is increased by some technique, the other measure usually sinks or at least remains unchanged. Therefore, tools may either score at a good precision or a good recall, or they may have an average recall and precision. Since both precision and recall are crucial parameters to judge the match quality of a tool or approach, a third measure is used that includes both values. This so-called F-measure or F_1 score is the harmonic mean of recall and precision, and is defined as in Equation 7.3.

$$f = \frac{2 \cdot r \cdot p}{r + p} \tag{7.3}$$

Determining the quality of the semantic enrichment of ontologies is more intrinsic than determining the quality of a mapping without semantic information. Given a correspondence $c \in M$, two aspects can now be evaluated:

1. The correspondence itself (is it contained in the benchmark or not?)
2. The correspondence type (is it correct or not?)

7.2 Evaluating Semantic Mappings

7.2.1 Problem Description

STROMA obtains an input mapping M and extends it to an enriched (semantic) mapping ME. If no mapping repair is carried out, it always holds $|M| = |ME|$, otherwise it holds $|ME| \leq |M|$. Additionally, there are benchmarks B, BE, with B containing the expected correspondences and BE containing the expected correspondences with their expected type (enriched benchmark). It always holds $|BE| = |B|$.

In evaluations, there are two forms of input mappings that STROMA can obtain: a perfect mapping which is complete and correct, or an "authentic" mappings that was precalculated with a match tool (and which can be assumed to be incomplete and partly incorrect).

If a perfect mapping is used as input, it holds $M = B$. This makes the evaluation easier, since only the analysis of the correspondence type of each correspondence $c \in M$ becomes necessary; the correspondence itself is always correct. However, if an authentic mapping is used and it holds $M \neq B$, two important questions arise: First, how are false correspondences treated that are not in B? Let us assume that there is a correspondence *(cars, `is-a`, vehicle)* in ME, which is not in BE, as BE only contains the more specific correspondence *(cars, `equal`, automobiles)*. First of all, STROMA obviously retrieved the correct type (cars are vehicles), but as the correspondence is not part of the benchmark, the evaluation would require to treat the type as false, as the correspondence itself is already false. As a matter of fact, STROMA cannot tell whether a given correspondence is true or false, at least not if some strategies actually corroborated a specific relation type. In this case, the precision of the type evaluation would decrease, although STROMA may have denoted several correspondences correctly.

The second question asks, how correspondences are treated that are in B, but not in M? For instance, there might be a correspondence *(pedelecs, bikes)* in B, which was not detected by the initial match tool. Since M does not contain this correspondence, STROMA will not determine any type and the recall decreases, even though STROMA might have determined the correct type if the missing correspondence was in the initial mapping.

This problem description clearly shows the differences between using a perfect mapping and an authentic mapping as input. The perfect mapping is suitable to assess the sole mapping enrichment quality. In fact, it is suitable to judge the quality of each single strategy, of the use of background knowledge and of STROMA in general. However, such an idealized form of evaluation does not show how STROMA would behave in real-world mapping tasks, where it would not obtain perfect mappings as input. Using authentic mappings as input shows the effectiveness and behavior of STROMA as part of the more complex workflow (two-step approach), but insufficiently shows the actual quality of the mapping enrichment strategies.

For the evaluation of STROMA, we will use both evaluation methods, with a stronger focus on perfect mappings as input. For authentic input mappings, different ways of recall and precision calculations become possible, which we will explain in Section 7.2.3.

7.2.2 Measures for Perfect Mappings

If a perfect mapping is used as input, each relation type can be evaluated by the standard recall and precision measures. Since the mapping ME contains the same correspondences as BE, the evaluation only focuses on the relation types. Given a correspondence $c \in ME$, its type can be either correct or false. Simply speaking, this means that anything which is not correct is false and vice versa. This binary nature implies that recall, precision and F-measure are not only interrelated, but that they are always the same. Therefore it holds:

$$r = p = f \tag{7.4}$$

This on first sight uncanny rule is actually quite natural, because any correspondence with a false type is a correspondence whose correct type was not determined. Given a recall and precision of 1, a falsely typed correspondence thus reduces the precision by $\frac{1}{n}$ with n being the number of correspondences in ME resp. BE. At the same time, the recall is reduced by $\frac{1}{n}$, as the correct relation type of the given correspondence could not be found. Let ME_t be the correspondences in ME of a specific relation type t, and BE_t the correspondences in BE of a specific type t. To evaluate the relation type t, we use the type recall r_{type} and type precision p_{type}, which are defined as follows:

$$r_{type} = \frac{|ME_t \cap BE_t|}{|BE_t|} \tag{7.5}$$

$$p_{type} = \frac{|ME_t \cap BE_t|}{|ME_t|} \tag{7.6}$$

Equation 7.4 does not hold if specific relation types are regarded. For instance, there could be 10 `is-a` relations in the benchmark and STROMA could have found 9 `is-a` relations with 2 being falsely typed. In this case, it holds $ME_{(is-a)} = 9$, $BE_{(is-a)} = 10$ and $ME_{(is-a)} \cap BE_{(is-a)} = 7$. The recall is then $r_{(is-a)} = 7/10 = 0.7$ while the precision is $p_{(is-a)} = 7/9 = 0.63$.

STROMA is able to determine six relation types. A further question is how to handle correspondences for which no type could be determined and which are thus denoted as `undecided`. Each of such correspondences would be automatically falsely typed. In the default configuration of STROMA, correspondences denoted as `undecided` are turned into `equal` relations once the enrichment has been accomplished, which is referred to as the *undecided-as-equal* mode. This default type is used since input mappings often have a tendency to `equal` correspondences and this type seems to be the most likely one to hold. Thus, in default configuration each correspondence in ME has exactly one assigned relation type. Another possible configuration is the *undecided-as-false* type, in which any correspondence whose type could not be determined remains of type `undecided` and is thus automatically treated as a falsely typed correspondence. In the evaluation, we will show the results for both configurations.

7.2.3 Measures for Authentic Input Mappings

Given the enriched mapping ME and the enriched benchmark BE, there are further subsets that need special attendance for the type evaluation in case that it holds $M \neq B$. As illustrated in Fig. 7.2, let β be the typed correspondences of a specific relation type r in ME (blue set) and α be the typed correspondences of type r in BE (red set). The correspondences $\alpha \cap \beta$ are correspondences correctly typed by STROMA that are also in BE. For a better illustration, we denote further portions of α and β as follows:

$$\beta^+ = \beta \setminus BE \tag{7.7}$$

$$\beta^* = \beta \setminus (\beta^+ \cup (\alpha \cap \beta)) \tag{7.8}$$

$$\alpha^+ = \alpha \setminus ME \tag{7.9}$$

$$\alpha^* = \alpha \setminus (\alpha^+ \cup (\alpha \cap \beta)) \tag{7.10}$$

Hence, β^+ comprises correspondence of type r that only occur in M, but do not occur in B. Analogously, α^+ comprises correspondences that occur in B, but not in M. These two subsets are the result of the authentic mapping being not complete (missing correspondences in α^+) and containing false correspondences (correspondences in β^+). By contrast, β^* contains correspondences to which the type r was erroneously denoted (lower precision for this type), because the correspondence has a different relation type in the benchmark; α^* contains correspondences of type r that STROMA erroneously denoted by a different type (lower recall for this type).

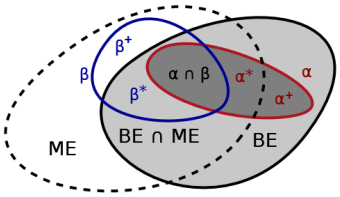

Figure 7.2: Illustration of the two overlapping mappings ME and BE and the overlapping subsets α and β containing correspondences of a specific type.

All correspondences in $\alpha^*, \alpha^+, \beta^*, \beta^+$ are treated as falsely typed, but an important question is which of those false correspondences were caused by STROMA, i.e., for which correspondences STROMA could have been able to denote the correct type. This refers only to the two sets β^* and α^*, where β^* consists of correspondences where STROMA denoted a false type and α^* consists of correspondences of a specific type not determined by STROMA. However, STROMA is not responsible for correspondences in β^+, which are false correspondences in the mapping, as well as for missing correspondences of a specific type in α^+, which are correspondences that are missing in the input mapping.

On the basis of the predefined variables, we will define two types of measures: effective measures and strict measures. The strict and effective recall measures are defined as follows:

$$r^s_{type} = \frac{\alpha \cap \beta}{\alpha} \tag{7.11}$$

$$r^e_{type} = \frac{\alpha \cap \beta}{\alpha \setminus \alpha^+} \tag{7.12}$$

The strict recall r^s treats correspondences of type t as not correctly determined that are not in the benchmark, so which cannot be denoted by STROMA. It is suitable to assess the overall recall of STROMA after the two steps have been carried out (matching and enrichment). The effective recall r^e disregards correspondences that are not in the benchmark. It is better to judge the enrichment strategies implemented in STROMA. The strict and effective precision are defined analogously:

$$p^s_{type} = \frac{\alpha \cap \beta}{\beta} \tag{7.13}$$

$$p^e_{type} = \frac{\alpha \cap \beta}{\beta \setminus \beta^+} \tag{7.14}$$

The strict precision p^s denotes correspondences of type t as falsely denoted if they are not in the benchmark (it does not matter which type was assigned then). The effective precision p^e disregards such correspondences and once again is a better means to assess the quality of STROMA.

Since the effective recall and precision disregard correspondences in $M \setminus B$ and $B \setminus M$, it holds the same as for perfect input mappings:

$$r^e = p^e = f^e \tag{7.15}$$

The strict measures can never be greater than the effective measures, because missing resp. false correspondences in M automatically decrease p^s resp. r^s, while they have no effect on r^e resp. p^e. Thus it always holds:

$$r^s \leq r^e, \quad p^s \leq p^e, \quad f^s \leq f^e \tag{7.16}$$

Only in the rare case in which a 100 % correct and complete mapping has been determined ($M = B$) it holds:

$$r^s = r^e, \quad p^s = p^e, \quad f^s = f^e \tag{7.17}$$

Finally, the strict measures can never be greater than the match recall r resp. the match precision p. Thus, it also holds:

$$r^s \leq r, \quad p^s \leq p, \quad f^s \leq f \tag{7.18}$$

In the very special case that all correspondences have been correctly typed, it holds:

$$r^s = r, \quad p^s = p, \quad f^s = f \tag{7.19}$$

7.3 Evaluation Data Sets

Table 7.1 provides an overview of seven data sets that are used to evaluate STROMA. The first three columns provide information about the data set in general (number, name and language), while the following three columns provide some information about the original data sources: the number of nodes (concepts) in the source ontology, the number of nodes in the target ontology and the number of correspondences the data set contains. The last 4 columns specify the number of correspondences for each relation type within the data set.

No.	Name / Domain	Lang.	#Src. Nodes	#Trg. Nodes	#Corr.	*equal*	*is-a / inv. is-a*	*has-a / part-of*	*related*
G_1	Web Directories (W)	DE	741	1892	340	278	52	5	5
G_2	Diseases (D)	EN	2795	1131	395	354	40	1	0
G_3	TM Taxonomies (T)	EN	328	256	762	70	692	0	0
G_4	Furniture (F)	EN	144	24	136	13	111	11	1
G_5	Groceries (G)	EN	293	56	169	29	127	2	11
G_6	Clothing (C)	EN	128	29	142	10	124	8	0
G_7	Literature (L)	EN	40	154	83	14	69	0	0

Table 7.1: Overview of evaluation scenarios and data set mappings.

Web Directories (G_1) is a gold standard between the Yahoo and Google Web taxonomies (product catalogs of e-shopping platforms) which is in German language and contains many itemizations. This gold standard was especially used to evaluate STROMA for German-language mapping scenarios where background knowledge has a relatively little impact, since only few German language resources are available. The gold standard was also used in our previous publications, e.g., [95, 109, 3].

Diseases (G_2) is an extract of a diseases mapping (English language) between Google and DMOZ[53]. It consists of many domain-specific terms and itemizations.

Text Mining Taxonomies (G_3) is a gold standard developed by means of crowd sourcing and was provided for research issues by SAP Research [110]. It consists of two text mining taxonomies (OpenCalais and AlchemyAPI) that were matched. Unfortunately, SAP Research could only provide the typed mapping, but not the original source files. To use this data set in the evaluation of authentic mappings, which requires the matching of the two source files, those files were automatically reconstructed from the mapping. As the

[53] `http://www.dmoz.org/`

mapping may not be complete, the generated source files may not be fully equivalent to the two original taxonomies, though.

In addition to those already existing data sets, some new mappings were developed for the evaluation of STROMA. **Furniture (G_4)** is a gold standard between the furniture categories of *amazon.com* and *ebay.com*. Analogously, **Groceries (G_5)** is a gold standard between the categories *Groceries* and *Food & Beverages* and **Clothing (G_6)** is between the *clothing* categories. All these gold standards were manually developed and are in English language. Amazon categories are typically much more specific than Ebay categories, so that the Amazon taxonomies are generally more comprehensive than the Ebay categories. To keep the development and mapping effort at a moderate level, only the top 2 category levels of Amazon were regarded.

Finally, **Literature** (G_7) is a gold standard between two literature taxonomies (goodreads and BIC). In detail, the gold standard was created between the goodreads top-level genre classification[54] and between the subject categories and qualifiers scheme of the BIC (Book Industry Communication), which is a standard classification for the UK book trade.[55] As the goodreads classification is quite limited to fiction, while the BIC classification is much more comprehensive, several parts of the BIC classification could not be mapped to the goodreads classification. Thus, only 83 correspondences were created.

Sometimes, even for humans it is difficult to find the correct relation type. In gold standard G_3, which was created by a crowd of volunteers, different falsely labeled correspondences were discovered. For example, the majority decided that *automobile* and *vehicle* are synonyms, though most linguists would see vehicle as a clear hypernym of automobile. The correspondence *(road, street)* was also denoted as `inverse is-a` correspondence, though WordNet defines streets as a specific kind of road, which means that an `is-a` relation may be more appropriate. There were even examples which were inversely labeled by the crowd, e.g., the majority assumed that a *director* is a specific form of a *film director*, though the opposite is the case (or at least `equal` has to be chosen).

As illustrated in Section 3.2.4, it is sometimes also difficult to distinguish between `is-a` and `part-of` relations. The different point of views on semantic relations thus have to be regarded when carrying out experiments and judging the quality of STROMA. As a matter of fact, given gold standards were not rectified in the evaluation, e.g., if the gold standard defines an `equal` relation for *(Automobile, Vehicle)*, the correspondence *(Automobile, `is-a`, Vehicle)* is regarded to be falsely typed, even if a dictionary proves differently.

Some of the data sets have already been used in our previous publications, e.g., in [5] and in [6]. Due to the constant evolution of STROMA and the SemRep system those results are no longer up-to-date. In the context of this doctoral thesis, all experiments have been carried out again, with the most current version of STROMA and its present default configuration.

[54] `https://www.goodreads.com/genres`
[55] `http://editeur.dyndns.org/bic_categories`

	recall	precision	F-meas.
W	46.9	51.7	49.2
D	70.7	35.8	47.5
T	87.1	89.2	88.1
F	75.6	87.7	81.2
G	77.1	78.8	78.0
C	87.9	90.6	89.2
L	69.5	72.7	71.0
Avg.	73.5	72.4	72.0

Table 7.2: Evaluation results for non-equal relations.

7.4 Evaluations Based on Perfect Input Mappings

7.4.1 Evaluating the Full STROMA System

First, we will show the results for the perfect input mapping using the full STROMA system, with all strategies (including SemRep) being enabled. Table 7.2 shows the results for all correspondences different from `equal`. Such correspondences possess a key role in this research, because they are the kind of correspondences other match tools usually do not detect.

As an interesting result, the average recall and precision are very similar, with the recall (73.5 %) being only 1.1 % above the precision (72.4 %). It might occur odd that the F-measure (72.0 %) is below these two values, but such a situation is generally possible, since in some experiments the recall is above the precision while in other experiments the opposite is the case. Calculating the average can subsequently lead to the peculiar situation in which both average recall and average precision are above the average F-measure. There is no difference between the configurations undecided-as-equal and undecided-as-false in this experiment, as the different configurations only influence the results for equality correspondences.

The recall is between 46.9 and 87.9 %. Precision is more dynamic, ranging between 35.8 and 90.6 %. The F-measure is between 47.5 and 89.2 %. These results show that the obtained quality depends much on the given mapping. The diseases experiment (D) and the German experiment (W) led to rather poor results, while the other experiments obtained far better results.

As `is-a` relations predominate in most data sets, Table 7.3 shows the results for the `is-a` type. Comparing those results to the previous table, it can be seen that the results are constantly better and that the F-measure of 76.2 % is more than 4 % above the result for all non-equality correspondences. In fact, the detection of `is-a` relation is more accurate than the detection of `part-of` or `related` relations. This remains an open issue, because none of the 27 `part-of` relations in the data sets have been found, and only very few `related` correspondences have been found. Apparently, the exploited background knowledge does not contain enough meronym and holonym relations so that `part-of` re-

	recall	precision	F-meas.
W	54.5	51.7	53.1
D	72.5	37.2	49.2
T	87.1	91.1	89.1
F	82.9	91.1	86.8
G	84.3	81.7	82.9
C	93.5	92.8	93.2
L	69.6	92.3	79.3
Avg.	77.8	76.8	76.2

Table 7.3: Evaluation results for is-a relations.

	Undecided-as-false			Undecided-as-equal		
	recall	precision	F-meas.	recall	precision	F-meas.
W	81.1	79.4	80.2	90.2	88.2	89.1
D	75.9	85.6	80.4	86.1	97.1	91.2
T	85.7	69.7	76.8	91.4	74.4	82.0
F	61.5	26.6	37.1	76.9	33.3	46.5
G	82.7	75.0	78.6	82.8	75.0	78.7
C	100	71.4	83.3	100	71.4	83.3
L	78.6	64.7	70.9	78.6	64.7	70.9
Avg.	80.8	67.5	72.5	86.6	72.0	77.4

Table 7.4: Evaluation results for equal relations.

lations cannot be found by STROMA. By contrast, the detection of `is-a` relations, which often predominate, is very successful.

The results for `equal` correspondences are shown in Table 7.4. Now, there is a noticeable difference between the two configurations for handling `undecided` correspondences. The average recall for the configuration undecided-as-false is 80.8 %, while the precision is 67.5 % and the F-measure is 72.5 %. In the configuration undecided-as-equal, the recall for detecting `equal`-correspondences increases, because some `equal` correspondences that could not be determined before are now automatically typed "`equal` " and are thus correctly determined. However, only in 5 of the 7 experiments the recall got increased (W, D, T, F, G). At the same time, the precision increases. Altogether, the average recall increased to 86.6 % (+ 5.8 %), and the precision increases to 72.0 % (+ 4.5 %). The F-measure increases by 4.9 % to 77.4 %. Thus, the configuration undecided-as-equal leads to much better results and is the default strategy of STROMA. However, in experiments the configuration undecided-as-false seems more appropriate, as it can better illustrate the actual quality and effectiveness of the implemented strategies.

Comparing the results with the results of non-equality types (Table 7.2) reveals that the recall for equality correspondences (86.6 %) is much better (compared to the 73.5 %), while the precision (72.0 %) is almost identical (compared to the 72.4 %), which leads to a better F-measure for `equal`-correspondences (+ 5.4 %). In fact, detecting `equal` corre-

	Undecided-as-false	**Undecided-as-equal**
W	75.1	82.5
D	75.8	85.0
T	87.0	87.5
F	74.3	75.7
G	78.1	78.1
C	88.7	88.7
L	71.0	71.0
Avg.	78.6	81.2

Table 7.5: Evaluation results for all correspondences.

spondences is generally easier, as some `equal`-correspondences are matches between two identical concepts, whose type is quite obvious.

Eventually, Table 7.5 shows the overall results for all correspondences, i.e., without focusing on a specific type. In this case, it holds $r = p = f$. It can be seen that the configuration undecided-as-equal leads to the best overall results. The F-measures are in a rather tight range between 71.0 and 88.7 %, which shows that STROMA provides a rather constant quality considering the diversity of the different data set. The average F-measure is 81.2 %, which means that about 81 correspondences out of 100 correspondences are correctly typed. Regarding the high complexity of languages, this can be considered a very good result.

If the configuration undecided-as-false is used, STROMA achieves still very good results, with a slightly worse F-measure of 78.6 % (- 2.6 %). However, a different result can only be observed in benchmarks W, D, T, F, while in the other three benchmarks the two configurations led to the same result (these are data sets containing only very few `equal` relations).

7.4.2 Evaluating Selected Strategies

Table 7.6 shows the F-measure achieved by each single strategy, with the best results in each experiment marked bold. In this test, only one strategy has been enabled, so either Compound, Background Knowledge, Structure, Multiple Linkage or Word Frequency. Note that the Itemization Strategy must always be enabled to achieve any results for itemizations. To focus more on the strengths and weaknesses of the different strategies, the configuration *undecided-as-false* is used.

The **Compound** Strategy allows F-measures between 18.9 and 65.9 %, with an average of 44.9 %. It is usually a very convenient technique, but has a generally restricted scope, as it can only be used in correspondences where a compound word matches its head. **Background Knowledge** leads to constantly good results between 49.1 and 73.5 %, with an average of 64.0 %. This is the best result of all strategies and is caused by the multiple resources imported in the SemRep system that allow a correct relation type determination for many correspondences. In 3 of the 7 experiments, the best results have been achieved

	Compound	Background Knowledge	Structure	Multiple Linkage	Word Frequency
W	62.7	61.5	59.4	50.6	**75.4**
D	65.9	**73.5**	58.7	57.5	64.4
T	18.9	68.4	7.7	**78.0**	50.4
F	58.8	**63.2**	23.5	4.4	55.1
G	23.1	49.1	24.6	**71.6**	38.4
C	47.1	69.0	28.9	**84.5**	71.1
L	38.0	**63.4**	61.9	61.9	49.3
Avg.	44.9	64.0	37.8	58.4	57.7

Table 7.6: Evaluation results (F-measure) for single strategies.

by using background knowledge (D, F, L). The **Structure** Strategy can both lead to good results as in scenarios W, D, L, but can also lead to poor results as in scenario T. With an average of 37.8 % it is rather a special strategy useful for some specific cases where complex concept paths exist. The heuristic strategy **Multiple Linkage** has an even larger range. In experiment F, which apparently does not consist of many $(1 : n)$ or $(n : 1)$ correspondences, only 4.4 % F-measure was obtained. By contrast, an F-measure of 84.5 % was achieved in experiment C, which is very close to the F-measure achieved by using all strategies (see Table 7.5). The average F-measure is 58.4 %, which is the second-best result. In 3 of the 7 data sets this strategy even achieved the best results (T, G, C), which is an impressive result for a strategy mostly based on heuristics. Eventually, the **Word Frequency** Strategy achieves results between 38.4 and 75.4 %. The average F-measure of 57.7 % is very similar to Multiple Linkage, but the smaller range indicates that Word Frequency works more steadily. In one experiment (W) this strategy returned the best result.

It can be observed that background knowledge and heuristic strategies are complementary strategies. In W, T, G, C, heuristic strategies led to the best results while the Background Knowledge Strategy led to somewhat lower results. In D, F, L, the opposite was the case. Thus, using the more profound linguistic strategies as well as heuristic strategies seems to be an ideal configuration.

The Compound Strategy achieves only the fourth-best result, as it is exceeded by the generally good Background Knowledge Strategy and the relatively good heuristic strategies Multiple Linkage and Word Frequency. Still, this strategy is very important, because it can find `is-a` and `inverse is-a` relations that cannot be found by other strategies. Thus, this strategy does not achieve the best results if used alone, mostly because of a limited recall (there are many `is-a` relations which are not expressed by compounds), but in the overall STROMA system allows considerable improvements.

Table 7.7 shows the F-measures for an inverted experiment, where all strategies but one are selected. Thus, one specific strategy was disabled in each test, and each column shows the results achieved in such a configuration. The worst results are achieved if Multiple Linkage is disabled (73.9 %), which shows that this strategy has the largest impact on the overall result. Word Frequency and Background Knowledge have also a considerable im-

	Compound	Background Knowledge	Structure	Multiple Linkage	Word Frequency
W	75.7	76.0	75.1	75.1	64.2
D	74.8	68.7	76.3	75.8	75.8
T	86.3	81.7	87.0	82.2	86.6
F	72.0	71.3	73.5	74.2	68.3
G	77.5	79.8	76.3	59.7	78.6
C	88.1	87.4	88.0	83.8	89.4
L	71.0	68.6	66.2	66.2	72.2
Avg.	77.9	76.2	77.5	73.9	76.4

Table 7.7: Evaluation results (F-measure) for of STROMA if a specific strategy is disabled.

	Without heuristics	Without background knowledge	With all strategies
W	63.8	76.0	75.1
D	75.2	68.7	75.8
T	77.2	81.7	87.0
F	68.4	71.3	74.3
G	52.7	79.8	78.1
C	83.1	87.4	88.7
L	66.2	68.6	71.0
Avg.	69.5	76.2	78.6

Table 7.8: Evaluation results for specific combinations of strategies.

pact (76.4 % resp. 76.2 %), while disabling the Structure Strategy (77.5 %) and Compound Strategy (77.9 %) reduced the general F-measure only slightly (which is 78.6 %). There are two important insights: First, that the impact of a single strategy is very low and only ranges between an F-measure loss of 0.7 % (Compound) and 4.7 % (Multiple Linkage). This substantiates that the generally good quality of STROMA (78.6 % in undecided-as-false mode) is based upon the combination of the several strategies and not on a specific strategy alone. Second, that the deactivation of any strategy leads to a loss in F-measure, which proofs that any strategy has a positive impact on the mapping quality.

If no strategies were used at all, the F-measures would be 0 % in all benchmarks in the mode undecided-as-false. In the mode undecided-as-equal, the F-measures would be identical to the number (percentage) of `equal` relations in the benchmark.

Finally, Table 7.8 shows the F-measures for combinations of selected strategies. The first column shows how STROMA performs if no heuristic strategies (Multiple Linkage and Word Frequency) are used. The second column shows the results if all strategies except Background Knowledge are used. For comparison, the third column shows the result achieved by all strategies, which was already shown in Table 7.5.

If no heuristic strategies are used, the F-measure ranges between 52.7 and 83.1 %, with an average of 69.5 %. Thus, without heuristics the results are about 9 % worse compared to the results achieved with all strategies. This shows that the heuristic strategies are very useful after all. If no background knowledge is used, STROMA achieves results between 68.6 and 87.4 %, with an average of 76.2 %. This result shows that STROMA can even achieve very good results if no background knowledge is used.

As a matter of fact, the best results (78.6 % average F-measure in undecided-as-false mode) are obtained if all strategies are used. However, in experiments G and W the best results were obtained if no background knowledge was used. In this case, some relation types gained from SemRep were obviously incorrect. Since background knowledge has a rather high strategy weight, the incorrect result has sometimes outweighed the result of other (such as heuristic) strategies and thus led to a worse result in this experiment. One example is the rather simple correspondence *(Vinegar, Sauce)*. *Vinegar* seem to be best organized under the concept *Sauce*, so vinegar is some kind of sauce (`is-a`). Multiple Linkage correctly detects this relation, but Background Knowledge returns `related`, as in SemRep vinegar and sauce are both sub-concepts of the concept *flavoring*, which leads to the imprecise relation type `related`. In this case, Multiple Linkage outperforms Background Knowledge and leads to a better result, though it is ignored as Multiple Linkage has a lower weight.

7.4.3 Evaluating the Verificator Strategies

In the default configuration of STROMA the `is-a` and `part-of` verificator is used, while the equal-verificator is disabled. In this section, we will show the results for two different configurations: First, we will enable the `equal` verification, so that all verificators are used. Second, we will disable all verificators and show the results that are achieved if no verification is used. We show the results for the mode undecided-as-false, just as in the above experiments, but the results are practically equivalent to the undecided-as-equal mode.

	Default config.	**With all verificators**	**With no verificator**
W	75.1	59.7	73.1
D	75.8	72.7	69.7
T	87.0	87.0	87.0
F	74.3	75.7	74.3
G	78.1	78.1	78.1
C	88.7	88.7	88.7
L	71.0	71.0	71.0
Avg.	78.6	76.1	77.4

Table 7.9: Evaluation results for specific combinations of verificators.

Table 7.9 shows the F-measures for the three configurations. The first column shows the results obtained in the default configuration and is thus identical to Table 7.5. The second

column shows the results if all verificators are used. It can be seen that the average F-measure decreases by 2.5 %, implying that the `equal` verificator is more impairing than useful. It could improve the results in experiment F by 1.4 %, but also reduced the results in other scenarios by up to 15.4 %, as in experiment W. Therefore, the `equal` verificator is too vague for most experiments and is not enabled in the default configuration.

Finally, the third column shows the results if no verification is used at all. In this case, the average F-measure decreases by 1.2 %, although effects can only be observed in experiments W and D. Thus, verification has only effects on specific scenarios, but is still a good means to further increase the overall results.

7.5 Evaluations Based on Real Input Mappings

In this section, we present the results for authentic mappings. These mappings were generated using COMA 3.0 in its default settings. STROMA was also used in its default configuration to enrich the input mapping (undecided-as-equal mode). Table 7.10 shows the number of correspondences in each benchmark, the number of correspondences obtained by COMA 3.0 and the number of correct correspondences obtained by COMA 3.0 (overlap between M and B).

	Corresp. in B	**Corresp. in M**	**Corresp. in $M \cap B$**
W	340	343	223
D	395	486	270
T	762	163	63
F	133	185	95
G	169	35	17
C	144	19	12
L	83	17	17

Table 7.10: Statistics about the mapping M generated with COMA 3.0.

Table 7.11 shows the result for a selection threshold $\theta_0 = 0.2$, which means that selection has been carried out. The first main column shows the match quality of COMA, where only the correctness of correspondences was evaluated. The second main column shows the effective recall, precision and F-measure while the third main column shows the strict recall, precision and F-measure. With respect to the general match quality, STROMA (resp. COMA 3.0) achieves an average recall of 35.8 %, an average precision of 60.3 % and an average F-measure of 37.8 %. There is an enormous variance, though, as the recall ranges from 8.2 to 69.8 %. The precision is generally better and ranges between 38.6 and 100 %. In most cases, the recall is much lower than the precision (as in experiments T, G, C, L), while occasionally the opposite is the case (as in experiment F). These variations lead to a result in which average recall and average F-measure are almost identical (with only 2 % variance), while the precision of 60.3 % stands out against these two values.

	COMA Match Qual.			Effective Measures	Strict Measures		
	r	**p**	**f**	**r/p/f**	**r**	**p**	**f**
W	65.5	65.0	65.2	91.5	60.0	59.4	59.6
D	68.3	55.5	61.2	88.9	60.7	49.3	54.4
T	8.2	38.6	13.5	93.7	7.7	36.1	12.6
F	69.8	51.6	59.3	85.3	59.5	44.0	50.5
G	10.0	48.5	16.5	88.2	8.8	42.8	14.5
C	8.4	63.1	14.8	83.3	7.0	52.6	12.3
L	20.4	100	33.8	94.1	19.2	94.1	31.8
Agv.	35.8	60.3	37.8	89.3	31.8	54.0	33.7

Table 7.11: Evaluation results for real input mappings ($\theta_0 = 0.2$).

The effective recall, precision and F-measure are constantly very high, ranging between 83.3 and 94.1 %, by an average of 89.3 %. This is an even better result than in the evaluation of perfect mappings, where only an average of 81.2 % could be achieved. However, the effective measures are only calculated on the overlapping set of correspondences in the mapping and the data set. As we will show later in the comparison of STROMA with other match tools, such measures can quickly tend to very high values, as the overlapping part may be very small and may contain very obvious (trivial) correspondences. For example, the generated mapping in gold standard L contains only 17 overlapping correspondences, so that the effective F-measure of 94.1 % may not be very representative.

The strict recall ranges between 7.0 and 60.7 %, with an average of 31.8 %. The average strict precision is between 36.1 and 94.1 %, by an average of 54 %. The average strict F-measure is between 12.3 % and 59.6 %, by an average of 33.7 %. As a matter of fact, these values cannot exceed the match quality recall, precision and F-measure depicted in the first main column. This shows the actual dependency of STROMA towards the input mapping. Since those mappings have partly a very low score, like T and G, the overall result is automatically low, too.

The results presented above are the results for using COMA 3.0 in relaxed configurations and selecting correspondences having a critical confidence (as described in Section 5.2.3). This adaptation leads to a better result than using COMA 3.0 in its default configuration, where it holds $\theta = \theta_0 = 0.4$. Table 7.12 shows how the values change if this default configuration is used and no selection would be carried out. Naturally, the recall is lower in such a configuration where the acceptance threshold is 0.4 and not 0.2. The recall sinks from 35.8 to 30.7 % (- 4.9 %). The precision increases slightly, from 60.3 to 61.1 % (+ 0.8 %), but does not completely compensate the exacerbated recall. Thus, the average F-measure decreases by 3.2 % from 37.8 to 34.6 %. This corroborates the general importance of the selection strategy used in STROMA. It considerably improves the recall and also leads to an F-measure improvement of 3.2 % in absolute numbers, and even an improvement of 9.2 % in relative numbers.

	COMA Match Qual.			Effective Measures	Strict Measures		
	r	**p**	**f**	**r/p/f**	**r**	**p**	**f**
W	54.7	67.3	60.3	93.5	51.1	63.0	56.4
D	65.3	55.7	60.1	88.7	57.9	49.4	53.3
T	8.2	38.6	13.5	93.7	7.7	36.1	12.6
F	56.6	54.6	55.5	81.8	46.3	44.6	45.4
G	8.8	48.3	14.8	86.6	7.6	41.9	12.8
C	8.4	63.1	14.8	83.3	7.0	52.6	12.3
L	13.2	100	23.3	100	13.2	100	23.3
Agv.	30.7	61.1	34.6	89.7	27.3	55.4	30.9

Table 7.12: Evaluation results for real input mappings ($\theta_0 = 0.4$).

Evidently, the relaxed configurations of COMA and the selection made in STROMA lead to better initial results. This also leads to a better strict F-measure, which increases from 30.9 % (base configuration) to 33.7 % (relaxed configuration), which is an improvement of 2.8 %. The effective F-measure almost remains the same, which is natural, because the effective measures do not depend much on the initial match quality and are not expected to change much if the match quality changes.

7.6 Time Measurement

Table 7.13 shows the average execution time per correspondence for each experiment. The first column shows the execution times for STROMA without using SemRep while the second column shows the execution times for using STROMA with SemRep. Altogether, the execution times range between 12.4 and 57.7 ms per correspondence, with an average of 34.4 ms. This means that STROMA is able to process about 30 correspondences per second. Using SemRep requires some additional time compared to the time required by STROMA alone. About 12.4 ms (36 %) of the overall execution time are needed by SemRep. The mappings have been executed on a Windows Server 2008 having 72 GB memory and containing two Intel Xeon E5540 CPUs (2.53 GHz). The time was measured using the internal system time.

7.7 Comparing Evaluation

In this section, we want to compare STROMA with different related semantic match tools. Such a comparison is generally difficult, because STROMA follows a completely different approach (two-step approach vs. one-step approach) and has a different focus (relation type detection vs. matching + relation type detection). Still, such a comparing evaluation seems very useful to demonstrate the quality improvement gained by the novel two-step-approach and linguistic strategies. In the following evaluation, we want to com-

	without SemRep	with SemRep
W	16.8	18.5
D	7.5	12.4
T	8.2	15.5
F	27.3	52.9
G	35.0	57.7
C	26.3	39.5
L	33.0	44.2
Avg.	22.0	34.4

Table 7.13: Average execution time for one correspondence (in ms).

pare STROMA with TaxoMap and S-Match, which are the two most relevant, publicly available semantic match tools. We will both compare the general match quality and, especially, the quality of relation type detection. Compared to STROMA, these tools are one-step-approaches that carry out matching and relation type determination at the same time. Thus, it always holds $|M| = |ME|$, while in STROMA it is $|M| \geq |ME|$ because of the selection (mapping repair) applied in step 2. This divergence has no influence on the evaluation of match quality and type detection, though, and all tools have practically the same preconditions.

No.	Name / Domain	Lang.	#Src. Nodes	#Trg. Nodes	#Corr.	*equal*	*is-a / inv. is-a*	*has-a / part-of*	*related*
G_1	Ebay – Amazon	EN	24	144	136	13	111	11	1
G_2	Ebay – Wikipedia	EN	24	174	87	3	83	0	1
G_3	Wikipedia – Amazon	EN	144	174	138	16	114	7	1

Table 7.14: Overview of evaluation scenarios and data sets in the comparing evaluation.

A first comparison has been conducted and published in [5], between STROMA and the match tool S-Match. Such a comparison is generally very difficult, because S-Match does not allow OWL files as input format, but requires CSV-like or XSD files. To carry out the experiment, the data sets W (Web Directories) and D (Diseases) were converted into this S-Match specific input format. In the first experiment (gold standard W), S-Match returned only 4 correspondences, which have been all incorrect. Though the gold standard is in German language, it remains unclear why some correspondences between near-identical concept names have not been found. Apparently, S-Match draws strongly on WordNet, which is a completely inconvenient resource for a German-language mapping. In the second experiment (gold standard D), S-Match returned almost 20,000 correspondences, although the reference mapping contains less than 400 correspondences. S-Match did not calculate the most relevant correspondence for a concept, but multiple correspondences for each concept. Though these correspondences may have been all semantically correct, it was impossible to manually check those correspondences. Comparing the result with the data set thus led to a very low precision close to zero. The recall was about 3 %, which is extremely low given the huge number of correspondences that was retrieved.

Seeing that both data sets were inappropriate for a comparison, as S-Match follows a different way of mapping generation, a new gold standard between Wikipedia, Ebay

	Match Quality			Effective Measures	Strict Measures		
	r	**p**	**f**	**r/p/f**	**r**	**p**	**f**
STROMA/COMA	69.8	51.6	59.3	85.3	59.5	44.0	50.5
S-Match	74.4	24.2	36.5	84.5	63.1	20.5	30.9
TaxoMap	20.6	62.2	30.9	100	20.6	62.2	30.9

Table 7.15: Experimental results for gold standard G_1.

and Amazon was developed. In each resource, the furniture taxonomy was manually extracted and three gold standards were created: Ebay–Amazon (G_1), Ebay–Wikipedia (G_2) and Wikipedia–Amazon (G_3). Note that the gold standard G_1 is identical with the furniture gold standard (F) used in previous evaluations. An overview of the 3 benchmarks is provided in Table 7.14.

In this evaluation, S-Match and TaxoMap were compared to STROMA, which are the most relevant, freely available tools for semantic matching [6]. During the evaluation, the Wikipedia data set has been disabled, as it might have helped STROMA in some of the Wikipedia-based scenarios.

Table 7.15 shows the results for the first data set. The first main column shows the recall, precision and F-measure of the match quality, i.e., without regarding the type. COMA 3.0 (which was used in the first step) and S-Match obtained a very good recall of 70 % resp. 74 %, while TaxoMap achieved only 21 % recall. By contrast, TaxoMap achieved the best precision (62 %), while COMA 3.0 achieved a somewhat lower precision of 52 %. S-Match had the worst precision (24 %). Regarding the F-measure, S-Match and TaxoMap achieved similar results (37 % resp. 31 %) while STROMA achieved 59 %.

In the effective measures, TaxoMap achieved the best results. All correspondences have been correctly typed. This leads to the remarkable result $r = r^s, p = p^s, f = f^s$. S-Match and STROMA achieved almost identical results (84.5 resp. 85.3 %). However, results look different if the strict measures are taken into account. Since COMA achieved a much better match quality, it consequently achieves better results, because less correspondences are missing and less correspondences are false. It reaches a strict F-measure of 51 %, while S-Match and TaxoMap achieve an identical F-measure of 31 %.

This first experiment reveals that S-Match and TaxoMap produce inverse results. S-Match achieves a high recall, but low precision, while TaxoMap achieves a high precision, but low recall. STROMA (resp. COMA 3.0) is between the two tools, but obtains the best F-measure. Regarding the effective F-measure, STROMA and S-Match achieve similar results, which are considerably lower compared to TaxoMap, which achieved 100 %. This is caused by the considerably larger intersection of correspondences in the input mapping and gold standard compared to TaxoMap. For example, many simple correspondences like *(Table, Table)* were discovered in this intersection produced by TaxoMap, which can be easily typed with `equal` or easily resolved with WordNet. STROMA and S-Match also found more difficult correspondences that were not always typed correctly, resulting in a lower effective F-measure. For this reason, the strict measures seem more appropriate

	Quality of Matching			Effective Measures	Strict Measures		
	r	**p**	**f**	**r/p/f**	**r**	**p**	**f**
STROMA/COMA	27.5	36.3	31.2	91.6	25.2	33.3	28.6
S-Match	10.3	2.6	4.1	100	10.3	2.6	4.1

Table 7.16: Experimental results for gold standard G_2.

	Quality of Matching			Effective Measures	Strict Measures		
	r	**p**	**f**	**r/p/f**	**r**	**p**	**f**
STROMA/COMA	18.1	24.8	20.5	88.8	11.5	17.2	13.7
S-Match	39.8	9.1	14.8	36.3	14.4	3.3	5.3

Table 7.17: Experimental results for gold standard G_3.

for an overall comparison. In fact, an effective F-measure of 100 % can be easily achieved if the match tool detects only 1 correspondence between source and target schema and types it correctly (which then indicates a very low match recall). Thus, only the strict F-measure can reveal how good the match quality and type detection work together.

Table 7.16 and Table 7.17 show the results for gold standard G_2 and G_3. These experiments were more sophisticated and STROMA and S-Match achieved far lower results. TaxoMap could not detect a single correspondence between the taxonomies, so that no results were returned. Analyzing TaxoMap closer, it seems that it uses too strict parameters, allowing the previously demonstrated good precision, but considerably reduces the recall (which was finally 0 % in the two experiments).

In both experiments, STROMA achieves the best match quality and the best strict F-measure. In G_2, S-Match achieves the best effective F-measure, while the opposite is the case in G_3. As in previous evaluations, STROMA obtains a very stable effective F-measure above 80 %. The average strict F-measure of STROMA in all three experiments is 30.9 % compared to an average strict F-measure of 13.3 % achieved by S-Match.

The comparative evaluation showed that STROMA achieves the best results in semantic matching. This is quite natural, as it strongly focuses on relation type detection and uses much more complex, linguistic-based strategies than the other tools. Besides, COMA 3.0 was used for the initial match phase, which is a very successful, highly complex match system allowing even good results in more difficult match tasks. STROMA thus draws advantage of two different systems, the schema and ontology matcher COMA and the enrichment strategies presented in this thesis, so its overall good results are certainly quite comprehensible.

Still, both S-Match and TaxoMap can be regarded as very sophisticated and useful semantic match tools, especially in view of the fact that they are among the first systems that ever calculated semantic mappings; they are also among the very few approaches that provide a user-friendly GUI and are available for download. Without this availability, a comparison would not have been possible. There are also situations where S-Match

and TaxoMap achieve very good results. For example, in the first experiment S-Match could achieve the best match recall, while TaxoMap achieved the best match precision.

Part III

The SemRep System

8 Background Knowledge Extraction from Wikipedia

In this chapter, we will present a novel approach to parse Wikipedia definition sentences and to extract semantic concept relations. We start with an introduction and problem description in Section 8.1 and introduce important semantic relation patterns used for this approach in Section 8.2. In Section 8.3, we finally present the workflow for relation extraction from Wikipedia. The subsequent chapters discuss the integration of those relations in a semantic repository and their exploitation in ontology mappings (Chapter 9), the different possibilities of quality improvement (Chapter 10) and the evaluation of the Wikipedia relation extraction, as well as the semantic repository (Chapter 11).

8.1 Introduction

Since high-quality general purpose thesauri are limited, the development of new resources and their integration together with existing resources is a significant step to achieve better results in mapping enrichment. In the context of this research, the English Wikipedia was used as a basis to automatically derive and collect new semantic relations, which are used as additional background knowledge for the already available resources like WordNet and UMLS.

Wikipedia comprises several advantages for background knowledge extraction. First of all, Wikipedia is a large resource that contains more than 4 million articles. It can be assumed that almost any imaginable concept found in an ontology resp. mapping is contained in Wikipedia, with the exception of compound words (there are articles for *towel*

and for *bathroom*, but not for *bathroom towel*). Wikipedia is up-to-date (unlike WordNet) and articles have generally a good text quality. Furthermore, the textual content of Wikipedia is provided under a GFDL and Creative Commons License, which makes it easy to obtain and use the Wikipedia data.[56]

The gist of our Wikipedia approach is as follows: Most articles start with a so-called *definition sentence* that defines the concept (or entity) at hand. In a typical definition sentence, a concept c is defined by its direct hypernym concept c', together with some specific attributes which distinguishes c from c'. This way of concept definition is very intuitive. Someone who does not know what *slippers* are may know what *shoes* are, so it seems natural to define slippers as a form of shoes having some additional properties (e.g., shoes that are primarily worn indoors). The Wikipedia definition of *slipper* follows exactly this notion:

> *Slippers are light shoes which are easy to put on and take off and usually worn indoors.*[57]

Thus, definition sentences usually contain `is-a` relations (*slippers are shoes*) that are valuable pieces of information for a background knowledge resource to build. Since each mental concept is represented in Wikipedia only once, synonym relations also occur in such definition sentences as the following example illustrates:

> *A laptop or a notebook is a portable personal computer with a clamshell form factor, suitable for mobile use.*[58]

In this definition sentence, *laptop* and *notebook* are obviously synonyms and it holds <*laptop*, `equal`, *notebook*>. Additionally, both concepts are related to *personal computer* by the `is-a` relation. Thus, three semantic relations are contained in the first part of the sentence:

1. <*laptop*, `equal`, *notebook*>
2. <*laptop*, `is-a`, *portable computer*>
3. <*notebook*, `is-a`, *portable computer*>

However, there is a further linguistic relation hidden in the sentence, which is indicated by the simple word *with*. The fragment *with a clamshell form factor* indicates a `has-a` relation, i.e., laptops resp. notebooks have a clamshell form factor (which describes the typical form of laptops so that they can be snapped shut). This means that the total number of semantic relations in the sentence is increased by two further `has-a` relations to altogether five relations.

In short, the basic notion of the Wikipedia approach is to parse any Wikipedia article, find specific semantic relation patterns (like "is a"), find the respective concept terms

[56] `https://dumps.wikimedia.org/legal.html`
[57] `http://en.wikipedia.org/wiki/Slipper` (March 2015)
[58] `http://en.wikipedia.org/wiki/Laptop` (March 2015)

that are related by those patterns and thus extract the semantic relations like <*laptop*, `is-a`, *portable computer*> or <*laptop*, `equal`, *notebook*>. The extracted relations are then written into an output file and are available as background knowledge for further processing. This includes the integration of the relations together with additional resources like WordNet into a unified semantic repository, and to use it effectively in the field of semantic mapping. This step is the core feature of the background knowledge repository SemRep and will be explained in detail in Chapter 9.

Relation extraction from Wikipedia has also some obstacles. First of all, Wikipedia contains a large amount of entity data like persons, cities, movies, organizations, as well as scientific articles about mathematical laws, chemical formulas, philosophic theories or historic events, which are rather irrelevant for a repository used in schema and ontology mapping (where only the lexicographic concepts are needed). Secondly, parsing texts is more difficult and prone to errors than parsing syntactically well-defined structures like the info boxes (as done in [11]) or the category tree (as done in [142, 116]). Finally, not all articles follow the rules of definition formation, so that some articles simply do not contain any plausible linguistic relations and cannot be processed. This reduces the recall, i.e., the semantic relations of some concepts cannot be extracted. We will take up this subject in more detail in the evaluation of this approach (Chapter 11.1).

8.2 Semantic Relation Patterns

8.2.1 Overview

A *semantic relation pattern* is a specific sequence of words that expresses a linguistic relation of a certain type. In the Wikipedia approach, where such semantic relation patterns constitute the key elements, `is-a`, `has-a` and `part-of` patterns are exploited. Semantic relation patterns always connect at least two concept terms that appear to the left and right of the pattern, very much like operands in algebra that appear to the left and right of an operator, such as $a > b$. Unlike mathematical relations, textual relations are much more difficult to parse, because many patterns are possible and the related concepts do not necessarily stand directly before or after the pattern.

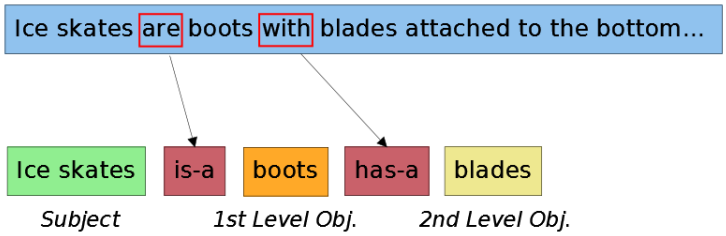

Figure 8.1: Sample sentence containing two semantic relation patterns.

Fig. 8.1 provides a simple sentence that consists of two semantic patterns (`is-a` and `has-a`), with the related concepts standing directly left and right of the patterns. The

Is-a Patterns
is a
is typically a
is any form of
is a class of
is commonly any variety of
describes a
is defined as a
is used for any type of

Table 8.1: Typical `is-a` patterns.

sentence, which is from the *Ice Skates* article, defines the concept *ice skates*.[59] This concept (and all possible synonyms mentioned in an article) are referred to as *subject*, similar to the syntactic subject in a sentence. The concepts to which the subject is related are called *objects*. In the example, *boots* and *blades* are the objects, and to distinguish them (they refer to different patterns after all), we call *boots* the first level object and *blades* the second level object. Any object is in a semantic relationship with the subject; in the ice skates example, *boots* is a hypernym of the subject and *blades* is a meronym.

8.2.2 Is-a Patterns

The most important and most versatile pattern is the `is-a` pattern. It can be quite simple, e.g., *"X is a Y"* or *"X are Y"* (plural), but can also become as complex as *"X is any variety of a Y"* or *"X is commonly any form of Y"*. Such patterns often occur with an additional (time) adverb, like *commonly*, *typically* or *generally*. They also occur with so-called *collectives*, like *a set of*, *a class of* or *a group of*, e.g., *"Penalty methods are a certain **class of** algorithms..."*. They can also occur with so-called *partitives* like *form of*, *kind of* or *type of*, e.g., *"A baker's rack is a **type of** furniture with shelves..."*. They come with different determiners (like *a*, *the*, *any*, *some*) or no determiner at all (as in the ice skates example). They invariably come with a verb, which is often (but not necessarily) the verb *to be*. Table 8.1 contains some typical examples for `is-a` patterns as they appear in Wikipedia definition sentences.

8.2.3 Part-of and Has-a Patterns

Patterns for `part-of` and `has-a` relations are less versatile and occur less frequently in definition sentences. The adverb *within* and the prepositions *in* and *of* indicate `part-of` relations, e.g., *"A CPU is the hardware **within** a computer."* (*CPU `part-of` computer*), or *"Desktop refers to the surface of a desk"* (*desktop `part-of` desk*). A list of `part-of` and `has-a` patterns is presented in Table 8.2. However, these patterns can be misleading, as such simple prepositions can be used in various contexts, as in *"Leipzig University was founded*

[59] `http://en.wikipedia.org/wiki/Ice_skate` (March 2015)

Part-of Patterns	Has-a Patterns
within	consists/consisting of
as part of	having
in	with
of	

Table 8.2: Typical `part-of` and `has-a` patterns.

Synonym Patterns
A, B and C
A, also called B
A, also known as B or C
A, sometimes also referred to as B

Table 8.3: Typical `equal`-patterns.

in *the late Middle Ages."*, which would lead to the not really useful relation <*Leipzig University*, `part-of`, *Middle Ages*>.

Typical `has-a` patterns are indicated by the verbs *to consist*, *consisting of* and *having*, as well as the word *with*. Again, some patterns can be misleading, especially the rather diversely used words *having* and *with*. For example, the sentence *"A screw-propelled vehicle is a land or amphibious vehicle designed to cope* ***with*** *difficult snow and ice or mud and swamp."* is a misleading case, as it can lead to false relations like <*snow*, `part-of`, *screw-propelled vehicle*>.

8.2.4 Equal-Patterns

Equal-relations are normally found in itemizations and are usually indicated by simple words like "and" or "or", e.g., *a bike* ***or*** *bicycle is a specific vehicle...* There are a few more complex constructs as shown in Table 8.3. They belong to the classic Hearst patterns as first introduced in [71]. Outside itemizations, definition sentences might also contain binary patterns like *A stands for B* (in abbreviations), *A is a synonym for B* or *A is a synonym for B*. Such patterns are extremely rare, though, because there is normally only one Wikipedia article per concept. Thus, instead of a definition sentence like *"A car is a synonym for automobile."* articles rather look like *"An automobile, auto car, motor car or car is a wheeled motor vehicle."*.

For this reason, `equal` patterns are not used in the pattern detection phase of the algorithm, but later on, when the respective concepts are determined. They are treated differently than `is-a`, `has-a` and `part-of` patterns, though the outcome remains the same.

8.3 The Extraction Workflow

8.3.1 Overview

The workflow of the Wikipedia relation extraction is illustrated in Fig. 8.2. As a first step, the Wikipedia articles have to be extracted from the Wikipedia dump file (see Section 8.3.2). The extracted and cleaned articles are stored in a document store (MongoDB) where they can be easily accessed in the following steps.

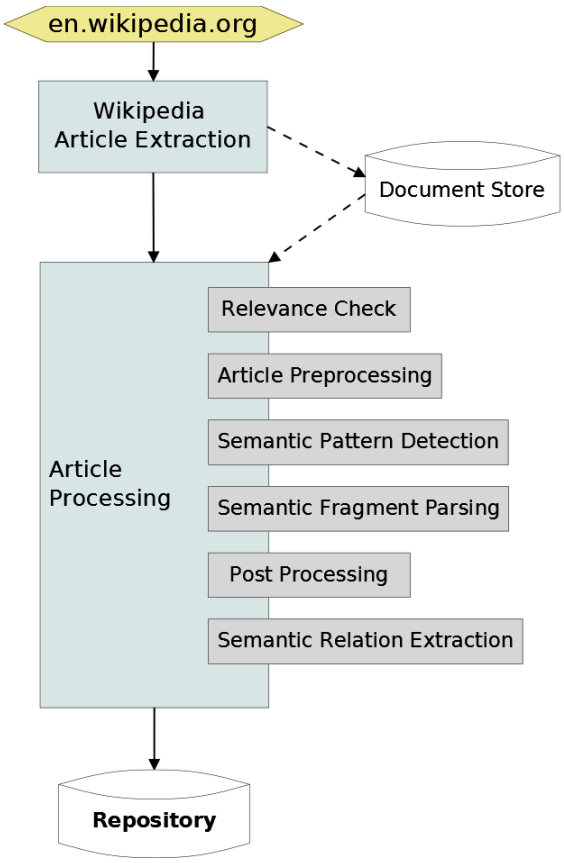

Figure 8.2: Workflow for Wikipedia relation extraction.

After this preparatory phase, the article processing commences. Iterating through each article in the document store, the following 6 sub-steps are carried out:

1. The article is checked for relevance. If it turns out to be irrelevant, the article is skipped (see Section 8.3.3).
2. The article text is preprocessed (see Section 8.3.4).
3. The definition sentence is parsed and the semantic patterns are identified (see Section 8.3.5).

4. The definition sentence is split where the patterns are located and within the resulting fragments the concepts (subjects and objects) are identified (see Section 8.3.6).
5. Some post processing is applied to the extracted concepts (see Section 8.3.7).
6. From the extracted patterns and concepts, the semantic relations are generated and persistently saved (see Section 8.3.8).

Once all articles have been processed, the extracted relations can be imported in the semantic repository (SemRep) as described in Chapter 9. The approach works fully automatically, i.e., no user interaction is necessary. Except for a few manually built resources like a list of English collectives and partitives or a list of anchor terms for the pattern detection, the approach does not rely on any background knowledge resources. In the following, we will illustrate each step and sub-step in detail.

8.3.2 Wikipedia Article Extraction

The first step is to obtain the latest Wikipedia dump file, which is basically a huge XML file containing all Wikipedia pages.[60] For each article item, the article text and some meta-data like the title, page URL or categories are listed. In the context of this research, the dump file from October 2013 was used, which has a size of about 11 GB (packed) and 41 GB unpacked. The file contains about 11 million article items, which is more than twice the number of actual articles contained in Wikipedia (which was about 4.3 million at this time). In addition to the real articles, the dump file also contains large numbers of images, category pages, talk pages and re-directs, which are extraneous for the purpose of relation extraction. Such articles can be easily detected by analyzing the page name. For example, the Wikipedia-internal URI for talk pages invariably starts with "Talk:", e.g., *"http://en.wikipedia.org/wiki/Talk:Computer"* and media files like images start with "File:".

Using an XML parser, the dump file was processed from top to bottom and all irrelevant items were removed until the 4.3 million actual articles remained. From each article text, the first 750 characters were extracted. Since only the first sentence of each article is important (the definition sentence), trimming the article texts to its first 750 characters suffices. There are other possibilities for trimming, e.g., removing any text that appears after the abstract (the first text block of an article that occurs before the table of contents), but not all articles have such abstracts and some articles only consists of a very few sentences.

The extracted text is cleaned from Wiki-specific formatting commands by using the Java Wikipedia API (Bliki engine).[61] Together with the page name and the article categories, the article is stored in a document store for the further processing.

[60] `http://dumps.wikimedia.org/enwiki/`
[61] `http://code.google.com/p/gwtwiki`

Redirects

Redirect pages are not ignored, because such pages represent valuable `equal` relations. For instance, the page "streetcar" redirects to the page "tram", which means that the two terms are synonyms. The Wikipedia approach builds such relations directly when extracting the articles. They are stored separately from the actual relations that are later extracted from the article texts. The redirects form a separate repository called *Wikipedia Redirect Relations*, allowing different configurations of the SemRep system (e.g., with or without the redirects, or with different weights for the redirect relations).

There are different forms of redirects, some that are helpful for the repository to build, and some that are obstructive. Altogether, five different forms of redirects were discovered:

1. **Synonyms,** e.g., the terms *tramcar*, *streetcar* and *trolley car* refer to the Wikipedia article *tram*.
2. **Shortenings,** e.g., the term *Merkel* refers to the Wikipedia article *Angela Merkel*.
3. **Different ways of spelling,** e.g., the term *colour* refers to the Wikipedia article *color*.
4. **Common misspellings,** e.g., the term *libary* refers to the correctly spelled article *library* and *Merckel* refers to *Merkel*.
5. **True references,** e.g., the term *UML Modeling* refers to the article *UML Tool*, which also handles the subject of *UML modeling*.

The fourth type is obstructive, because it means that misspelled (and thus technically not existing) concepts are imported into the repository. This cannot be prevented, though, since no specific information about the redirect type is provided in Wikipedia. The fifth point is also to some degree obstructive, because no actual synonym relation holds. In the example, there is no synonym relation between a UML tool and UML modeling, but the relation is rather "UML tool is used for UML modeling". Still, Wikipedia redirects are generally helpful in schema and ontology mapping and according to experiments some inappropriate redirects do not noticeably impair the mapping quality.

The Wikipedia approach automatically rejects redirects which consist of 5 words or more. These are usually book titles, movies, institutions and the like, which are irrelevant for the repository. Additionally, articles are rejected if their title consists of special characters that are untypical for English concepts, such as numbers, slashes, letters of different languages, etc.

8.3.3 Relevance Check

One serious issue of the Wikipedia relation extraction is to distinguish between entities and concepts. As a lexical resource designed for schema and ontology mapping, entities are not needed in the repository and can make it more comprehensive than necessary and

impair the overall quality of the repository. As an example, the relation <*Albert Einstein*, `is-a`, *Physicist*> is unimportant for the repository, since it is very unlikely to encounter a concept *Albert Einstein* in a schema or ontology. By contrast, a relation like <*Physicist*, `is-a`, *Scientist*> is important for the repository. The *relevance check* is an implemented method to determine upfront whether a given Wikipedia article is about a concept (and thus should be processed) or about an entity (and thus should be discarded).

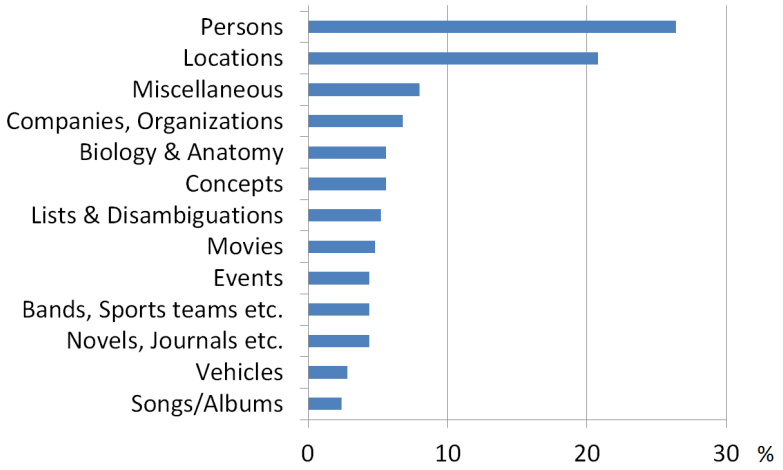

Figure 8.3: Approximate distribution of Wikipedia articles across different categories.

Such a relevance check becomes even more important if the distribution of articles in Wikipedia is taken into account. Fig. 8.3 shows the distribution of 300 randomly checked Wikipedia articles across several domains. As the chart clearly shows, most articles are about persons and locations, which already make up nearly half of all articles (approx. 48 %). Sole concept articles make up only 7 %. With a total amount of 4.3 million articles in the Wikipedia dump, this means roughly 300,000 concept articles. Though the articles from the biologic and anatomic domain can also be considered to be concepts (species, animals, diseases, etc.), and maybe even some selected brands or companies (like *Porsche*, which is both a company and a car, or *Linux*, which is an operating system), the number of remaining articles that are relevant for the repository is small compared to the number of irrelevant articles.

A useful way to detect and discard entity articles is to look at the Wikipedia categories in which a given article appears. For example, most articles about persons are in a category "`Year` births" (like "1879 births"). If they are still living, they often appear in the category "Living people", which currently contains about 700,000 articles. Otherwise, they often appear in a category "`Year` deaths" (like "1955 deaths").

Following a similar technique as used in [142], patterns are defined that indicate an entity article. To cover a broad range of domains and block as many articles as possible, the categories of 1,000 random articles were analyzed. With the gained knowledge, more than 400 regular expressions were built that indicate a category in which only entity articles would be listed.

Let us denote the set of all Wikipedia categories by C_W and the set of categories in which a Wikipedia article occurs by C_A. It holds $C_A \subseteq C_W$. There are altogether four different types of regular expressions:

1. **Exact match expressions (E_1):** An article is blocked if there is a $c \in C_A$ that matches exactly an expression $e \in E_1$, e.g., *"living people"* or *"given names"*.
2. **Starts-with expressions (E_2):** An article is blocked if there is a $c \in C_A$ that starts with an expression $e \in E_2$, e.g., *"towns in *"*, *"airports in *"*, *"studio albums by *"*, etc.
3. **Ends-with expressions (E_3):** An article is blocked if there is a $c \in C_A$ that ends with an expression $e \in E_3$, e.g., *"* companies"* or *"* organizations"*.
4. **Matches expressions (E_4):** An article is blocked if there is a $c \in C_A$ that matches an expression $e \in E_4$. In this case, *match* refers to the matching of a regular expressions, e.g., "[1-2][0-9][0-9][0-9] births".

Altogether, 56 *exact match* expressions, 256 *starts-with* expressions, 43 *ends-with* expressions and 93 *match* expressions were devised. As an example, Fig. 8.4 shows the list of categories in which the article *Paris* occurs. Two of the implemented patterns would trigger in this instance so that the article is automatically discarded: the pattern "Cities in *" and "Visitor attractions in *".

Figure 8.4: Categories in which the article *Paris* occurs.

Blocking articles that are not about concepts yields several advantages:

- The final repository becomes less voluminous, which results in faster query executions.
- Several homonyms can be prevented, which often impair the result returned by the repository. For instance, the term *autumn* is not only a season (concept), but also refers to movies, bands and albums. Blocking such entities, misleading relations like "autumn is a movie" are effectively kept out of the repository.
- The overall execution time of the approach is reduced, because many articles do not have to be processed anymore. Though the category check requires some additional time, it makes the approach still somewhat faster.

However, this filter strategy is unable to filter all entity articles, because some articles appear only in very few categories which may not be handled by any of the regular expressions. Besides, there are many categories which are too specific for any generic filter. For example, the article *Canon PowerShot A*, which describes a camera product by

Canon, is only in one category "Canon PowerShot cameras", but it is entirely impossible to create filters for all such cases (e.g., for all companies, company products, etc.).

The filter technique is rather strict, because an article is discarded as soon as there is a $c \in C_A$ that matches any of the defined regular expressions. Still, experiments showed that the filters work reliably. From the 12.5 million relations that were obtained from the original extraction approach, the number of relations could be reduced to 4.7 million relations, while the filter precision was near 100 %. We will discuss this in more detail in Section 11.3.1.

8.3.4 Preprocessing

Before a sentence is processed, some textual preprocessing is performed. One important step of preprocessing is *sentence simplification*, in which complex expressions occurring in the sentence are replaced by synonymous, simpler expressions. This makes the sentence processing easier, because less special cases need to be regarded in the subsequent steps. For example, the rather bulky expression "is any variety of" is replaced by "is any", which can be easily processed by the approach. In some cases, expressions can be even completely removed, e.g., the expression "means of" as in the example "Boats are means of transportation", which leads to the simplified sentence "Boats are transportation".

Secondly, part-of-speech tagging is used on the sentence, because the subsequent steps require information about the word class of each word in the sentence. A POS tagger from the Apache OpenNLP Library for Java[62] is used to determine the word class of each word in the sentence. Accordingly, a word class key is annotated to all words in the sentence. After this, the sentence "Ice skates are boots with blades attached to it" looks as follows:

> *Ice_NN skates_NNS are_VBP boots_NNS with_IN blades_NNS attached_VBN to_TO it_PRP.*

Table 8.4 gives a clearer representation of the sentence together with the POS tags and their meaning. From a technical point of view, a definition sentence is a list of word objects with a word object being a tuple $(word, pos)$.

8.3.5 Pattern Detection

Pattern detection is the first major step of the relation extraction from Wikipedia definition sentences. Given a sentence S, it aims to find semantic relation patterns within the sentence and to split the sentence at the position of those patterns. The result of this step are sentence fragments $S_1, ..., S_n$ and semantic patterns $P_1, ..., P_{n-1}$ in between.

The approach can discover 1 or 2 semantic patterns within a sentence, so it holds either $n = 2$ or $n = 3$. If no pattern is found, the article cannot be processed and is immediately

[62] `http://opennlp.apache.org/`

Word	POS tag	Word class
Ice	NN	noun (singular)
skates	NNS	noun (plural)
are	VBP	verb (plural)
boots	NNS	noun (plural)
with	IN	preposition
blades	NNS	noun (plural)
attached	VBN	verb (past participle)
to	TO	"to"
it	PRP	personal pronoun

Table 8.4: POS tags of the words in the Ice Skates example.

skipped. Three combinations of patterns are regarded, which cover all combinations one would usually encounter in definition sentences:

1. $S_1 \overset{Hyponym}{\longrightarrow} S_2$
2. $S_1 \overset{Hyponym}{\longrightarrow} S_2 \overset{Hol./Mer.}{\hookrightarrow} S_3$
3. $S_1 \overset{Hol./Mer.}{\longrightarrow} S_2$

Hyponym refers to the `is-a` pattern and *Holonym/Meronym* refers to `has-a` resp. `part-of` patterns. The hooked arrow in case 2 denotes that this pattern links S_1 and S_3, but not S_2 and S_3, because the first fragment contains the subject, while S_2 and S_3 contain the first and second level objects. These objects are always related to the subject, but not necessarily among each other. Examples for the three cases are (patterns are put into brackets):

1. A submarine **[is a]** watercraft capable of independent operation underwater.[63]
2. Ice skates **[are]** boots **[with]** blades attached to it.
3. The Lipothrixviridae family **[consists of]** a family of viruses that infect archaea.[64]

Let S be a sentence consisting of a sequence of words $w_1, w_2, ..., w_m$. To find the semantic patterns, the sentence is parsed word by word, starting at w_1. Two finite state machines (FSM) are used that define how an acceptable pattern may look like. There is an FSM for `is-a` patterns, which is quite complex, as well as an FSM for `part-of` and `has-a` patterns, which is less complex. Each finite state machine uses a set of anchor terms A, which is a list of words that specify whether an initial state of the FSM can be entered. Let the parser be at word w_i. If it holds $w_i \in A$, the FSM is used for further parsing. The FSM will

[63] `http://en.wikipedia.org/wiki/Submarine` (March 2015)
[64] `http://en.wikipedia.org/wiki/Lipothrixviridae` (March 2015)

decide which initial state to choose (given the word w_i) and it defines several transitions to further states. The transitions lead to another word or sequence of words that follow w_i and may have some further conditions like "only if w_i is in plural form" or "only if the word following w_i is not a noun".

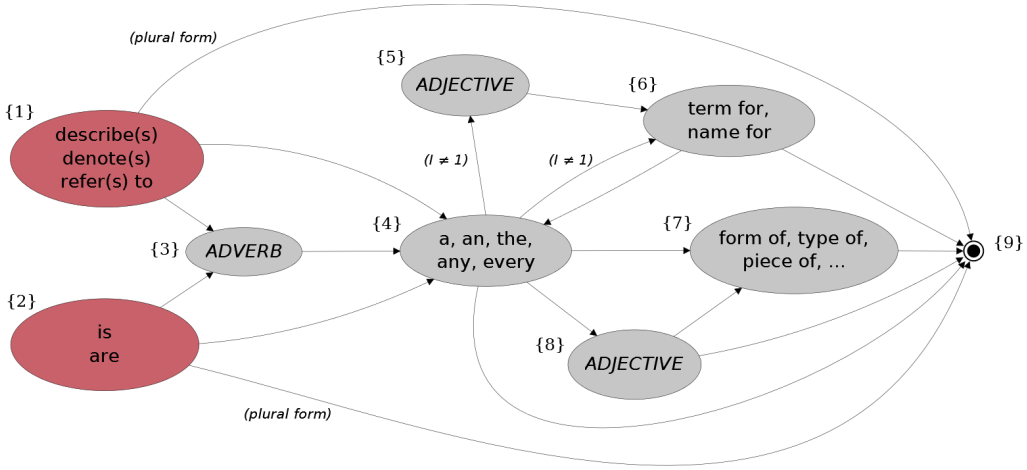

Figure 8.5: A simplified version of the `is-a` FSM.

Fig. 8.5 shows a simplified version of the FSM used to find `is-a` patterns. The two red states (1 and 2) are the initial states and contain the anchor terms that are used to enter the FSM. This means, for instance, if a word "denotes" appears in the sentence, the FSM is entered at state 1. An anchor term invariably indicates the beginning of a possible semantic pattern.

The approach will continue to parse the sentence word-by-word, but with due regard to the FSM. The general goal is to reach the final state 9, which marks the end of a pattern defined by the FSM. If this state is reached at w_j, the pattern has been fully discovered and the word parser continues at position w_{j+1} to possibly find another pattern further up in the sentence.

However, if the final state cannot be reached, the word parser is not within any defined pattern and the FSM is left immediately. The word parser continues to possibly find another anchor term further up in the sentence. In case that the FSM has been entered at word i and the parser is at position j when there is no valid transition to another state, the parser would now continue at position $i + 1$. It is possible that it hold $(i + 1) < j$ and the parser is thus moved back some words. This is absolutely necessary, because an anchor term of the other FSM might have been passed when walking through the present FSM.

The constructed FSMs are deterministic, i.e., for each state there is exactly one transition for each possible input (which cannot be fully seen in the chart because of its simplification). At some points, the parser must consider more than the next word, e.g., in state 4, the next two words must be checked to decide whether a transition into state 7 becomes possible. Additionally, the parser must store some specific attributes of the anchor word, e.g., whether it was in plural and to which initial state it refers. For example, a transition

from state 2 to 9 is only possible if the anchor term was in plural, and a transition from state 4 to 5 or 6 is only possible if the initial state was not 1. Thus, "is a term for" is a valid pattern (states 2, 4, 6, 9), while "describes a term for" is not valid, as it is an uncommon expression.

To illustrate the functionality of the approach, let us consider the following fictional example:

A clock is generally used to measure time.

At the third position in the sentence (w_3), the parser detects that the word "is" is an anchor term of the `is-a` FSM. It thus goes into initial state 2. The following word w_4 is "generally". There is a valid transition from state 2 to state 3, because w_4 is an adverb. The next word w_5 is "used" (an inflected verb). However, there is only one possible transition from state 3 to another state, which could only be followed if w_5 was a determiner like *a*, *an*, *any*, etc. Since this is not the case, no transition can be followed and the final state cannot be reached. The FSM will be left and the word parser continues at w_4, trying to find a semantic pattern further up in the sentence.

By contrast, the following example contains a true `is-a` pattern which would be detected by the FSM:

A clock is generally any instrument to measure time.

In this case, a transition from state 3 to state 4 becomes possible. In state 3, the next word w_5 is "any" (a determiner), allowing to follow the transition to state 4. Then, the following word w_6 is "instrument". Since no other state handles such a noun, the transition from state 4 to the final state 9 is used and the pattern "is generally any" has been discovered. The state route is {2, 3, 4, 9}.

Finding `part-of` and `has-a` patterns works analogously, except that there is a different FSM to handle such patterns. It comes with a separate list of anchor terms, which does not overlap with the anchor terms of the `is-a` FSM. Hence, given a specific word w_i, only one FSM can be entered.

The sentence parsing ends if an `is-a` pattern or `has-a` resp. `part-of` pattern was discovered (case 1 or 3), if an `is-a` and a `has-a` resp. `part-of` pattern was discovered (case 2) or if the end of the sentence was reached. If no pattern was detected, the article is discarded and no relations can be extracted. Otherwise, the sentence is split into fragments at the positions $P_1, ..., P_{n-1}$. The discovered pattern fragments are not part of those fragments.

If a second pattern is discovered, a threshold is used that only accepts this pattern if it is within a certain proximity of the first pattern. This threshold is necessary, as in longer definition sentences a pattern appearing rather at the end of the sentence may not be in any way related to the subject defined in the front part of the sentence. In the approach, a threshold of 7 was used, indicating that at most 7 words may be between the first and second pattern. If the distance is above this threshold, the second pattern would be rejected.

The two FSMs were manually developed based on the analysis of hundreds of randomly selected Wikipedia concept articles. They were systematically refined until most Wikipedia articles could be successfully processed. In order to facilitate the upcoming steps in the workflow, it was important to discover an entire pattern instead of parts of it. For example, in the sentence "A table is a piece of furniture", the correctly extracted pattern would be "is a piece of". If the parser would only detect "is a", the approach could subsequently extract a relation like <*table*, `is-a`, *piece*> or <*table*, `is-a`, *piece of furniture*>, which would be erroneous or at least inaccurate.

8.3.6 Concept Extraction

The result of the Semantic Pattern Extraction is a set of two or three fragments. The first fragment contains the subject(s) and is thus the subject fragment. The second and third fragment are the first resp. second level object fragments. In the following step, the concepts need to be extracted from those fragments in order to build the semantic relations.

To find the concepts that participate in a semantic relation, two further FSMs are used, one to parse the subject fragment, and one to parse object fragments. In many cases, the concepts appear directly left and right from a semantic pattern, which might suggest that the concept extraction can be easily carried out by searching for the first nouns left and right of the pattern. However, there are many examples in which such a naive assumption fails and a much more complex approach for concept extraction is necessary. Two examples are:

1. *A wardrobe, also known as an armoire from the **French**, is a standing **closet**.*[65]
2. *Column or pillar in **architecture** and **structural engineering** is a **structural element**.*[66]

In the first example, a relation like <*French*, `is-a`, *closet*> would be concluded and in the second example relations like <*architecture*, `is-a`, *structural element*>. Both relations are erroneous, because the correct relations would be <*wardrobe*, `is-a`, *closet*> and <*{column, pillar}*, `is-a`, *structural element*>.

The first example contains an additional apposition about the origin of a wardrobe. Such side notes about origin, grammar or pronunciation can frequently occur and must not be ignored while parsing a subject fragment, as erroneous relations may be the result. The second example contains a so-called *field reference*, which specifies to which field or domain the article refers to. Though there is no relation between a field reference and the semantic patterns, there is a relation between a field reference and the subject(s). The field reference suggests that the subjects belong to the specific field or domain and thus constitutes a `part-of` relation. The Wikipedia approach is able to extract such relations; in the second example, it would extract <*{column, pillar}*, `part-of`, *{architecture, structural engineering}*>.

[65] `http://en.wikipedia.org/wiki/Wardrobe` (January 2014)
[66] `http://en.wikipedia.org/wiki/Column` (January 2014)

The FSMs used for the fragment parsing are more complex than the FSMs to find semantic patterns, and they are used in a different way. While the pattern FSMs were entered somewhere in the sentence if a specific anchor term was found, these FSMs are entered at the very first word of the fragment. The final state, that has to be reached, marks the end of the fragment. This means that the FSM is used from the first to the last word of the fragment, not only for specific parts.

There exist special states in the FSMs indicating that a subject (resp. object) or a field reference was found. If such a state is reached, the respective terms are extracted and temporarily stored. If the subject and object fragment FSMs reached the final state, the extraction process is finished, as in this case at least one subject and one object were extracted. If no subject or object could be extracted, no relations can be built and the article is discarded.

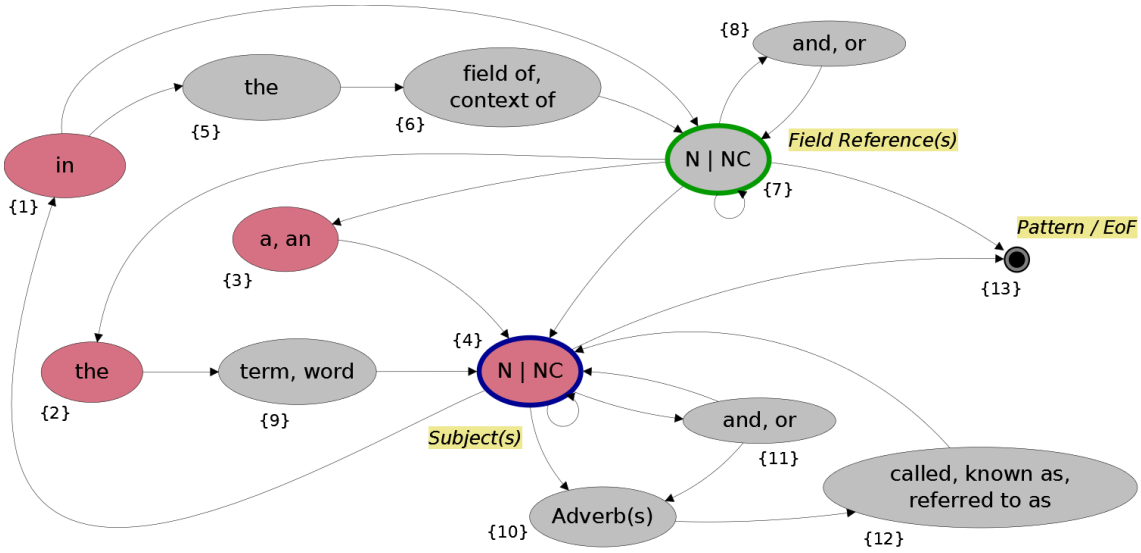

Figure 8.6: A simplified version of subject fragment FSM.

Fig. 8.6 shows a simplified finite state machine to process subject fragments, i.e., the sentence fragment appearing before the first pattern. In this fragment, the field references and subjects have to be extracted. Similar to the pattern FSMs, there are different initial states (red color). Besides, there are the two specified states 4 and 7. State 4 handles the subjects, which can be nouns (N) or noun compounds (NC). Analogously, state 7 handles field references, which also can be nouns or noun compounds. State 13 is the final state, which can be entered if the end of the fragment (resp. the beginning of the semantic pattern) was reached.

To illustrate the FSM, consider the following example:

A spindle, in furniture and architecture, is a cylindrically, symmetric shaft.[67]

[67] `http://en.wikipedia.org/wiki/Spindle_(furniture)` (January 2014)

The subject fragment is obviously "A spindle, in furniture and architecture". The first word "A" indicates that the FSM is entered at state 3. The following word, "spindle", is a noun, allowing a transition to state 4. As this state handles the sentence subjects, *spindle* is considered a subject. The following word "in" allows a transition to state 1. The next word is "furniture", which is a noun and thus leads to state 7. State 7 handles field references, so *furniture* is considered a field reference. The next word, "and", leads to state 8 and the final word "architecture" leads back to state 7. Therefore, *architecture* is again considered a field reference. Now, the end of the fragment is reached and there is a transition to the final state 13. Thus, the fragment was successfully parsed and one subject (*spindle*) and two field references were extracted (*furniture*, *architecture*).

Since there are many special cases that can occur in such definition sentences, the actually implemented subject fragment FSM is much more complex. It consists of about 20 states, 40 transitions and many additional conditions to decide whether a transition can be used or not. Object fragment processing works analogously to subject fragment parsing, though the object fragment FSM is less complex.

Parentheses

Parentheses play an important role in the sentence fragment parsing. Consider the following three examples:

1. *A burl (American English) or bur or burr (used in all non-US English speaking countries) is a tree growth...*[68]
2. *A curio (or curio cabinet) is a predominantly glass cabinet...*[69]
3. *Countertop (also counter top, counter, benchtop, (British English) worktop, or (Australian English) kitchen bench) usually refers to a horizontal worksurface...*[70]

In the first example, parentheses contain additional information about grammar, etymology, word usage and the like, which can highly impair the sentence parsing. As a consequence, one might decide to remove parenthesis expressions entirely. However, in the second example, the parenthesis expression contains a synonym (*curio cabinet* is synonym to *cabinet*), which would not be extracted if this expression was removed. Finally, in the third example, there is a complex parenthesis expression containing both valuable synonyms and obstructive usage information.

For this reason, deleting parenthesis expressions can be disadvantageous, because they can contain important information. On the other hand, parenthesis expressions can become very complex so that the fragment parser cannot process the sentence and is forced to discard the whole fragment (which means that no relations are extracted). Besides this, unreasonable information can be extracted if parentheses are not carefully treated.

[68] `http://en.wikipedia.org/wiki/Burl` (January 2014)
[69] `http://en.wikipedia.org/wiki/Curio_cabinet` (January 2014)
[70] `http://en.wikipedia.org/wiki/Countertop` (January of 2014)

Assuming that parentheses often contain a synonym expression to a given word, the approach might consider *American English* to be a synonym to *burl* in the first example and conclude <*burl*, `equal`, *American English*>.

In the Wikipedia relation extraction approach, parenthesis expressions are handled as follows: First, the fragments are processed without touching the parentheses. A small list of key words is used to distinguish between nouns like *American English* or *French* from actual concept names. If the FSM does not reach the final state, the parentheses are replaced by commas and the processing is tried again. There is also a configuration in which parentheses are turned into an apposition, e.g., *A car (automobile) is a...* is turned into *A car, or automobile, is a...* which can be easily processed by the FSM. Finally, if the fragment still cannot be successfully parsed, the parenthesis expression is removed. In this case, synonymous terms can be missed by the approach, but the approach may be able to successfully parse the sentence after all.

8.3.7 Post-processing

After concepts and relation patterns were extracted, some post-processing is performed on the concepts. First of all, remaining special characters like periods, commas or parentheses have to be removed. Additionally, stemming is used to get a unique representation of each concept. A Java implementation of the Pling stemmer was used to obtain the stemmed concepts.[71] Even though English nouns are very regular and stemming is much easier than in other languages, stemming can very occasionally fail. For example, the word *houses* is stemmed to the not existing word *hous* instead of *house*, which is called *overstemming* in computational linguistics. There is no straight way to discover such overstemming automatically, except with additional background knowledge (e.g., to look whether the stemmed word is contained in a preferably extensive dictionary or word list).

8.3.8 Determining Semantic Patterns

As a result of the relation extraction phase, a set of subjects S, first level objects O_1, second level objects O_2 and field references F has been extracted. In this last step, the semantic relations are built, which are (1:1)-relations of the form $(concept_1, type, concept_2)$.

When building the relations, a few aspects have to be regarded:

1. All subjects are related to each other by an `equal` relation.
2. All subjects are related to all field references by a `part-of` relation.
3. All subjects are related to all objects.
4. Objects are not (necessarily) related among each other.

[71] `http://www.mpi-inf.mpg.de/departments/databases-and-information-systems/research/yago-naga/javatools/`

The last point may be astonishing at first glance. In fact, there are situations where objects are also related among each other, e.g., by an `equal` relation as in *Netbooks are small **laptops** or **notebooks*** (<*laptop*, `equal`, *notebook*>). However, in many cases there is no clear relation between the objects, as the following examples shows:

> *A rocket is a missile, spacecraft, aircraft or other vehicle that obtains thrust from a rocket engine.*[72]

The four objects *missile*, *spacecraft*, *aircraft* and *vehicle* are in different relations to each other. For example, the relation between *aircraft* and *spacecraft* is `related` (co-hyponyms), the relation between *aircraft* and *vehicle* is `is-a`, while there is no clear relation between *missile* and *aircraft*. In contrast to the *netbook* example, no `equal` relations occur between those four objects. Therefore, the Wikipedia approach does not build relations between object terms, as the relation type cannot be unequivocally determined.

Let R be the set of (1:1)-relations gained from the extracted concepts and patterns of a Wikipedia article. Let R_S be the set of synonym relations that were extracted ($R_S \subset R$). It holds:

$$|R_s| \;=\; \binom{|S|}{2} \;=\; \frac{|S| \times (|S| - 1)}{2} \tag{8.1}$$

The number of relations between subjects and objects is $|S| \times |O_1|$ resp. $|S| \times |O_2|$. The number of relations between field references and subjects is $|S| \times |F|$. The overall number of extracted relations is thus:

$$|R| \;=\; \binom{|S|}{2} \;+\; (|S| \times |O_1|) \;+\; (|S| \times |O_2|) + \; (|S| \times |F|) \tag{8.2}$$

In some cases, $|R|$ can become pretty large. Imagine that there are 5 subject terms, 1 field reference, 2 first-level objects and 2 second-level objects. According to Equation 8.2, the number of extracted (1:1)-relations is 35. Though this is a rare case, this example points out the general richness of the approach compared to other work that concentrates on simply linking Wikipedia pages with other resources like WordNet, which usually leads to only one link per Wikipedia article.

The extracted relations are written into files that serve as import files for SemRep. In detail, there is a file for field references, a file for re-direct relations and a file for all remaining extracted relations (which comprises the main data set). In the next chapter, we will explain how the relations are integrated in this repository and how it can be used for mapping enrichment.

[72] `http://en.wikipedia.org/wiki/Rocket` (January 2014)

9 The SemRep System

In this chapter, we introduce the semantic repository SemRep. This repository allows the import and access (querying) of different lexicographic resources, which was especially designed to support mapping enrichment and matching in general. After an introduction (Section 9.1) we will describe the general architecture of SemRep in Section 9.2. In Section 9.3, we illustrate how SemRep is used to determine the semantic relation type between two concepts, which we call *query execution*. This section also includes the efficient search for paths between two concepts, the calculation of the path type in longer (indirect) paths and the scoring of paths (path confidence calculation). Eventually, we will discuss some technical and implementation aspects of SemRep in Section 9.4.

9.1 Introduction

Once the semantic relations have been obtained from the previously presented Wikipedia approach, they need to be integrated together with the relations from WordNet and further resources in a repository that provides a holistic view on the data. This semantic repository, dubbed SemRep, is designed to serve as background knowledge repository for STROMA and is primarily used to answer questions about the relation type holding between two given input words. Unlike other frameworks that combine semantic relations from different resources (like [44]), SemRep was designed for the field of ontology mapping, which led to some specific requirements and necessary adaptations. In particular, there had been the following requirements:

- **Velocity.** The repository has to process queries within a short time in order to be useful for (larger) mappings. This is especially important in interactive mapping

tasks where a user is interacting with the mapping enrichment tool and expects that the processing of the mapping is carried out within some seconds or at most within a few minutes.

- **Extendability.** The repository must be able to integrate further resources.
- **Multilingualism.** The repository should be able to cope with resources from different languages.
- **Simplicity.** Using and extending the repository should be as simple as possible.
- **Correctness and precision.** The repository should answer queries (as far as possible) correctly and should return the most relevant relation type.

Presently, SemRep contains semantic relations from up to five different resources: WordNet, Wikipedia, UMLS, OpenThesaurus (German relations) and ConceptNet. ConceptNet is not used in the default configuration of SemRep though, as its quality is relatively low and could impair the general usefulness of SemRep. Wikipedia consists of the automatically extracted Wikipedia relations, the field references and the separately extracted Wikipedia redirects.

Table 9.1 gives an overview of those resources, including their language, the way they were created and the number of concepts and relations they comprise (after all filters have been applied). Note that "Creation" refers to the way how the relations were built. Though the Wikipedia redirects have been automatically extracted, they were *manually* created (by users). This is an important difference to the automatically detected and created Wikipedia relations and to the automatically created ConceptNet relations that could be imprecise or incorrect. The number of relations refers to the number of links between two concepts, i.e., one relation describes both directions. Thus, the two atomic relations <*car*, `is-a`, *vehicle*> and <*vehicle*, `inverse is-a`, *car*> are treated as one relation and it is possible that there are less relations than concepts in a resource, just as in the *Wikipedia Redirects* case. Apparently, there are many concepts x in this resource, that have only one relation to another concept y. Apart from Wikipedia Redirects, the other resources contain more relations than concepts; WordNet even has a concept-relations ratio of almost 1:10.

Resource	**Lang.**	**Creation**	**#Concepts**	**#Relations**
WordNet	English	Manually	119,895	1,881,346
Wikipedia	English	Automatically	548,610	1,488,784
Wikipedia Field References	English	Automatically	66,965	72,500
Wikipedia Redirects	English	Manually	2,117,001	1,472,117
UMLS	English	Manually	938,527	1,265,703
OpenThesaurus	German	Manually	58,473	614,559
ConceptNet	English	Automatically	90,364	245,320

Table 9.1: Resources used in SemRep.

Though SemRep was implemented as a background knowledge repository for schema and ontology mapping, there are also further fields of application. As a matter of fact,

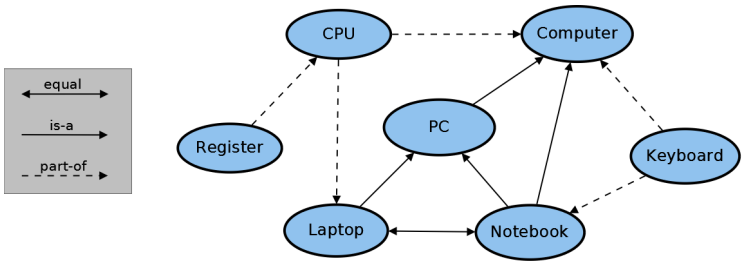

Figure 9.1: Sample excerpt from the repository.

SemRep could also be used for matching, i.e., in the first step of the two-step-approach presented in this thesis. Secondly, it can be used as a repository for (manually) verified, correct mappings. Such an approach is also called *mapping re-use*, where the manually confirmed correspondences of mappings can be re-used to foster new match tasks. Of course, a combination of such mappings and lexicographic resources in SemRep is also possible, and it is even possible to discover potentially incorrect lexicographic relations in other import resources by means of such verified mappings. Eventually, any kind of dictionary, thesaurus or background knowledge ontology can be integrated in SemRep, which would make it also applicable as a knowledge base for specific domains, for entity resolution, for specific linguistic tasks like query answering and related fields.

Similar to STROMA, SemRep uses different configuration parameters, thresholds and weights, and is a fully configurable system. A list of all default parameters used in SemRep is provided in Appendix B.

9.2 The SemRep Architecture

9.2.1 Overview

Basically, SemRep is organized as a simple graph with the nodes being concepts and the edges being the semantic relations in between (see Fig. 9.1). Edges are directed, but as each relation type has a well-defined inverse type, the graph can be traversed in any direction. Each node and edge has a few attributes which are necessary for path calculation and confidence measurement. Node attributes comprise the concept name and the resources in which the concept appears. Edge attributes comprise the semantic relation type and the resources where the relation appears.

SemRep does not distinguish between the different semantics of a concept, nor between the different languages. Given a specific word w, there is exactly one node. For example, *mouse* is represented by one node, although it has different meanings (an animal, an input device, a small person, etc.). The word *gift* resp. *Gift* exists both in German and English, but has different meanings in either language. Still, this word is only represented once in the repository (words are case-insensitive). This means that homonyms cannot be dis-

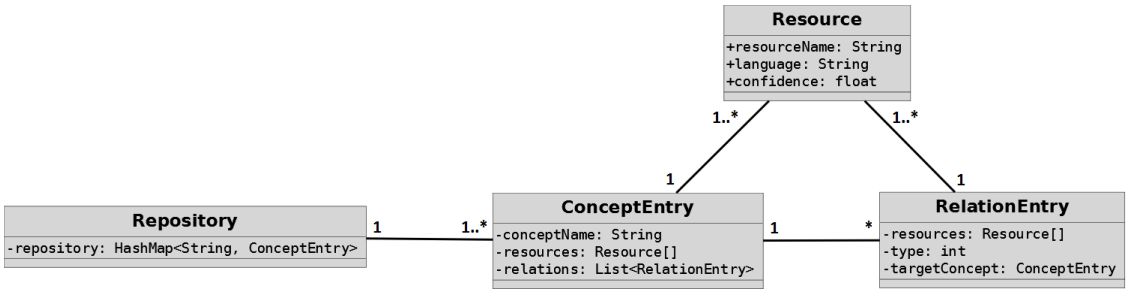

Figure 9.2: Repository infrastructure in UML notation (class diagram).

tinguished in SemRep, which can impair the quality of query execution. We will discuss this point in detail in Chapter 10.

9.2.2 Implementation Details

There are different possibilities to build a repository like SemRep. Intuitively, it seems reasonable to use a graph database, because of the repository's graph structure. In an early implementation, the graph database system Neo4j[73] was used to store the millions of relations from the different resources. Neo4j makes it relatively easy to import those relations and provides different techniques to find semantic paths between nodes (query execution). However, determining paths between two nodes was excessively slow and could take up to 30 seconds, which was an unacceptable execution time for STROMA, as it needs to process hundreds or thousands of correspondences within a mapping.

For this reasons, SemRep is a tailored implementation that utilizes a Java-based hash map structure and is run in main memory. The basic structure is illustrated in the UML model shown in Fig. 9.2. The central class *Repository* contains a *repository* hash map as its primary element, which is a set of concept entries. The hash key is based on the concept name, allowing fast access to a given concept. A *concept entry* has a name, a list of resources where it appears and a list of relation entries. A *relation entry* has a list of resources where it appears, a relation type (encoded by an internal number) and a target concept, which is another *concept entry*. Finally, a *resource* has a name (e.g., *WordNet*), a language (e.g., *English*) and a resource-specific confidence threshold used for path scoring (see Section 9.3.4). As an example, consider the two relations *(car, automobile)* and *(car, vehicle)*. Let us assume that the first relation was provided by WordNet, and the second by Wikipedia. There is one concept *car* with two relations to *automobile* and *vehicle*, which can be visualized in an object diagram as shown in Fig. 9.3.

In SemRep, relations are always directed. Given two concepts x, y, only one relation is stored in the repository. The opposite direction can be easily calculated, because any relation type r has a well-defined inverse relation type r^{-1}. For this reason, there is an $(1 : *)$ cardinality between *ConceptEntry* and *RelationEntry*, as there could be concepts

[73] `http://neo4j.com/`

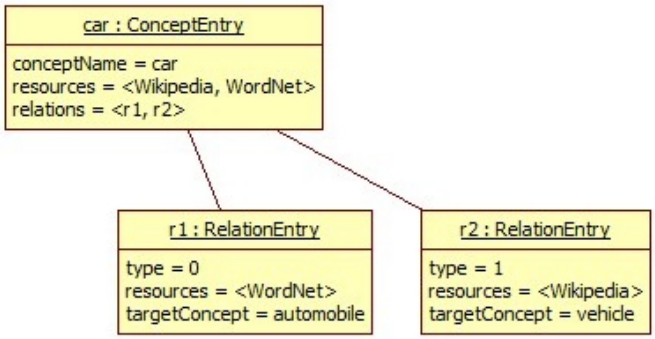

Figure 9.3: Repository infrastructure in UML notation (object diagram).

that have no outgoing relations to other concepts (which makes them leaf concepts in the repository). By contrast, the associations between *ConceptEntry* resp. *RelationEntry* and *Resource* are mandatory, i.e., each concept entry and relation entry must obviously occur in at least one import resource.

As there is only one node for one term, a node may represent concepts from different languages. For instance, the term *oldtimer* exists both in German and English, but it has a completely different meaning in either of the two languages. In English, an oldtimer refers to a veteran or an elderly person in general, while it refers to a vintage automobile in German. Thus, from the node *oldtimer* there are both German and English relations leading to other (German and English) nodes. However, such an implementation does not pose any problem so far, because the language of a concept entry and relation entry can be easily determined by the resource objects it contains (see Fig. 9.2). The language of the repository is English by default, but it can be changed at any time. For example, if German mappings are to be processed, the language has to be switched to German first. Then, if the concept *oldtimer* is queried, SemRep will only follow German relations and ignore relations of all other languages. Thus, there is no impairment by having one node for concepts originating from different languages and SemRep will always make sure that only paths of the selected language are used. Of course, this is a very simple implementation, which could be improved by an advanced repository that creates two nodes for terms of two different languages. For the general purpose of schema and ontology mapping, the current data structure seems sufficient, though.

9.2.3 Data Import

Each resource is a set of triples $(word_1, word_2, type)$, with the type being encoded by a single digit. It holds 0 = `equal`, 1 = `is-a`, 2 `inverse is-a`, 3 = `has-a` and 4 = `part-of`. The type `related` (co-hyponymy) is not supported for data import, because a co-hyponym relation is an indirect relation of the form X `is-a` Y `inverse is-a` Z, yet the repository only accepts direct relations. The triples are stored in a simple CSV-like text file, with two colons serving as separator. Listing 9.1 shows a sample excerpt of such a text file, which

is very similar to the STROMA import format (see Listing 5.1). There are more advanced formats to store triples, like RDF, but as the files solely serve for data import, there was no necessity to use such a more complex format.

```
car :: vehicle :: 1
car :: automobile :: 0
mountain bike :: bike :: 1
bike ring :: handlebars :: 4
```

Listing 9.1: Sample excerpt from an import file

The simple structure of the import file makes it easy to extend SemRep by further resources. The Wikipedia relations were already built in this input format, so that they could be directly imported to SemRep. WordNet, UMLS, OpenThesaurus and ConceptNet had to be converted into this list of triples, though, which had been more laborious. For example, WordNet uses a graph-like structure to organize its synsets, and each synset S comprises a set of synonym words. To convert WordNet into a set of (1:1)-relations, the WordNet tree has to be traversed from top to bottom and all words $w_1 \in S_1, w_2 \in S_2$ to be put into a relationship if S_1 is in any direct semantic relation to S_2. Besides, all w_1, w_2 within a synset have to be put into an `equal` relation if it holds $w_1 \neq w_2$.

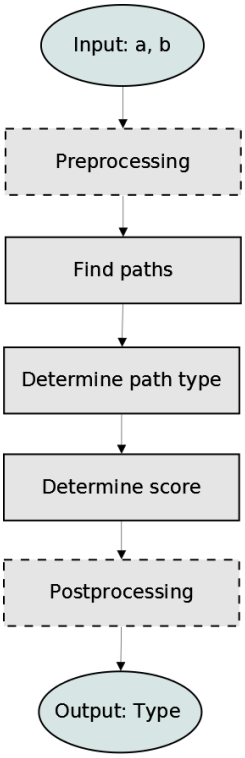

Figure 9.4: Querying Workflow.

9.3 Query Execution

The core feature of SemRep is to determine the relation type between two given words a, b or to return `undecided` if no relation between the two words can be found. This process is called *query execution*, and its primary goal is to find paths from a to b. Fig. 9.4 illustrates the 5 steps that are carried out to determine the type between a, b.

In the first step, some preprocessing is applied to the two input concepts, which is necessary if at least one of the two concepts is not contained in the repository (Section 9.3.1). Secondly, paths are determined between a and b (Section 9.3.2) and the type of each path is calculated (Section 9.3.3). Since there can be generally more than one path from a to b, a score is calculated for each path to determine the most relevant path (Section 9.3.4).

Eventually, some post-processing is applied and the relation type of the highest-scored path is returned (Section 9.3.5).

9.3.1 Preprocessing

To find a path between input concepts a, b is only possible if the two concepts are contained in the repository. If one of the two concepts does not participate in any relation of the input resources, the query cannot be resolved and the repository has to return `undecided`. Even though the repository contains millions of concepts, this case occurs relatively often and is usually caused by compound words. In most cases, such compounds are very simple or self-explanatory words so that there is no necessity to include them in any dictionary or lexicographic resource. For example, the term *chair* is contained in practically any English dictionary. The simple compound *kitchen chair*, which describes a chair usually found in kitchens, is not contained in most resources (including WordNet, Wikipedia, Wiktionary, the Oxford Dictionary[74] and the Free Dictionary[75]). The nature of compound words makes it unnecessary to define such words, because humans can easily determine the meaning of *kitchen chair* if they are familiar with the two words *kitchen* and *chair* (which are contained in the dictionaries). As a matter of fact, it is also impossible to list all imaginable compound words in a resource, considering the huge amount of compound words that could be actually created.

Gradual Modifier Removal

In SemRep, a technique called *Gradual Modifier Removal* (GMR) is used to handle compound words that are not contained in the repository. Given the two input words a, b, let us assume that a is not contained in SemRep, yet b is. If a is an open compound, GMR removes the first (the left-hand) modifier of a and gets the reduced word a'. In the case of kitchen chair, it holds $a' = chair$. Now SemRep checks whether a' is contained in the repository. If this is the case, SemRep takes a' and b as input words. Otherwise, GMR is carried out again, now with a' as input. This way, the modifiers of the open compound are gradually removed until the word is finally found or the compound head is reached. Only if the compound head is not contained in the repository, the preprocessing has failed and `undecided` will be returned.

If GMR was successful, STROMA determines the relation type t' that holds between a' and b. It has to be remembered that the original word has been a and that it invariably holds a `is-a` a', because a is a compound with a' being its head. Thus, it holds a `is-a` a' and $a\ t'\ b$. This is an indirect path and the final type can be determined as shown in Table 9.2.

As an example, let us consider the correspondence *(kitchen chair, seat)*. The repository does not contain the term *kitchen chair*, but *seat*. GMR removes the first (and only) modifier of *kitchen chair* and it holds $a' = chair$. This term is contained in the repository and

[74] `http://www.oxforddictionaries.com/`
[75] `http://de.thefreedictionary.com/`

t'	t
equal	is-a
is-a	is-a
inverse is-a	related
part-of	part-of
has-a	has-a
part-of	part-of
related	undecided

Table 9.2: Calculation of the final relation type t from the preliminary type t'.

it is returned *chair* is-a *seat*. According to Table 9.2, the final relation type is is-a, i.e., it holds *kitchen chair* is-a *seat*. If the term a would be more complex, e.g., *designer kitchen chair*, this technique would work analogously, but two modifiers have to be removed (first *designer* and second *kitchen*).

GMR works reliably for the English language, in which less-established compounds are usually open compounds. In the German language, however, this technique is less applicable, because German compounds are most often closed compounds. For example, the German word for kitchen chair is *Küchenstuhl*, yet not *"Küchen Stuhl"*. To use GMR in the German language, a preferably comprehensive word list is needed to extract the modifiers of the closed compound. However, such a technique requires much more time and is prone to errors because of pseudo compounds and different inflections of the modifiers.

If t' = related, the type cannot be unequivocally determined and undecided will be returned. We will discuss the calculation of relation types in indirect paths more specifically in Section 9.3.3.

Same Heads or Modifier Removal

If both a and b are open compounds and if both words are not contained in the repository, there are two further cases that can be observed:

1. Two compounds have the same modifier, e.g., *kitchen chair* and *kitchen seat*.
2. Two compounds have the same head, e.g., *apple cake* and *fruit cake*.

In the first case, the two modifiers can be removed without impairing the semantics of the relation type that holds. Thus, SemRep compares *chair* and *seat* instead of *kitchen chair* and *kitchen seat*. The result is once again *chair* is-a *seat*. As the input words had the same modifier, this is the final relation type and no further type calculation is necessary. This technique is called *Same-Modifier-Removal* (SMR).

In the second case, the two heads of the compounds are removed and the relation type t' between the remaining modifiers is calculated. This case is a little more complicated, because a relation type can only be determined if t' = is-a or t' = equal. For example,

the relation between *apple* and *fruit* is `is-a` and thus an *apple cake* is a *fruit cake*. It does not work in other relations like `part-of`, though, as there is no relation between a *station hall* and a *city hall* (although it might hold *station* `part-of` *city*). This technique is called *Same-Head-Removal* (SHR).

Effectiveness

In most cases the three strategies (GMR, SMR, SHR) make it possible to process queries containing terms that are not contained in the repository. The techniques can also be combined and proved to be very successful in experiments. However, non-compound words not being in the repository cannot be handled with those strategies. In this case, the word may be too specific to be listed in any input resource and SemRep will return `undecided`. Besides these specific preprocessing techniques, word stemming is applied to the input words, according to the language in which SemRep is used.

9.3.2 Path Search

SemRep uses a form of breadth-first search to find paths from a to b. This approach is motivated by the fact that short paths are generally more relevant and more likely to be correct than longer paths. Besides, shorter paths can be calculated much faster than longer paths, as the number of paths increases exponentially with the path length. The repository uses a constraint l that defines the maximal path length calculated by SemRep. It holds $l \in \{1, 2, 3, 4\}$, i.e., SemRep can calculate paths up to a length of 4. Such a constraint is necessary to keep the execution time low, since calculating all possible paths from a specific node would require too much time, given the huge number of nodes and edges the repository consists of.

If valid paths from a to b are found within a distance of d, there seems to be no reason to calculate further paths of length $d + 1$, even if it holds $(d + 1) \leq l$. This default configuration is called *First Paths*, so the breadth-first search stops as soon as paths of length d are found. Another configuration is *All Paths*, in which all paths up to a length of l are calculated. The two configurations are referred to as *termination mode*, as they specify when the path search has to stop.

Let us assume that the configuration is $l = 3$ and the relation type between CPU and PC has to be determined, as shown in Fig. 9.5. Starting from the concept *CPU*, SemRep calculates all paths of length 1. Since the target node PC cannot be found, SemRep calculates all paths of length 2. There are two results now:

p_1 = CPU `part-of` Laptop `is-a` PC
p_2 = CPU `part-of` Computer `inverse is-a` PC

If the configuration is *First path*, SemRep would stop at this point. Only if the configuration is *All Paths*, it would continue to calculate paths of length 3 and find the following paths:

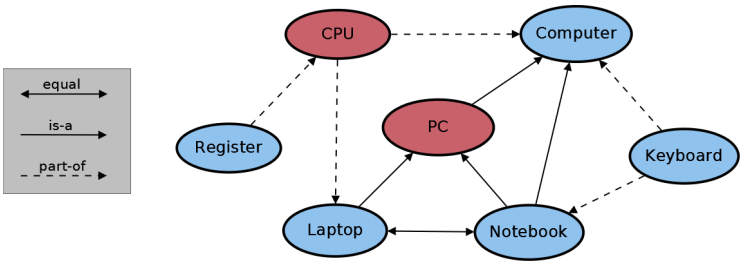

Figure 9.5: Sample query for *(CPU, PC)*.

p_3 = CPU `part-of` Laptop `equal` Notebook `is-a` PC
p_4 = CPU `part-of` Computer `inverse is-a` Notebook `is-a` PC

SemRep would stop at this point, because of the restriction $l = 3$. If l was 4, SemRep would go on to calculated paths of length 4.

Implementation

The classic breadth-first search, as applied in the above example, would start at node A and calculates all paths of length $1, 2, ..., n$ until it finds the target node B. Such an implementation is not optimal, though, because of the generally high node degrees. Common concepts found in mappings have a node degree G of about 100, although more general concepts can have some thousands of outgoing edges and less general concepts can have just a few outgoing edges. Let us assume that all paths up to length 4 have to be calculated. Theoretically, the number of paths is $G^p = 100^4$ (100 million). In practice, this number is much lower, because cyclic paths are ignored and longer paths can quickly reach more specific areas of the repository where the average node degree is lower. Still, the number of paths of length 4 can be several millions, depending on the input concept, which results in long, intolerable execution times.

Therefore, the first breadth search was extended to a bidirectional search. Instead of calculating all paths of length p from A to B, SemRep calculates all paths of length $\frac{p}{2}$ from A and from B, resulting in two sets of paths P_A, P_B. Subsequently, for each $p \in P_A$ and $p' \in P_B$ it is checked whether the target node of p is the target node of p', which we call *connector node*. If this is the case, the two paths p, p' express a path from A to B.

An example of this approach is illustrated in Fig. 9.6. Let us assume that a path from node A to E is searched and that it holds $l = 4$. Thus, A is the start node and E the target node. Starting from both nodes, paths of length 2 are calculated, resulting in $p = A - B - C$ and $p' = E - D - C$ as illustrated in sketch a). Node C is the connector node that allows to combine these two paths to the final path $p_{final} = A - B - C - D - E$. In order to obtain a correct path object, all relation types in Path p' have to be inverted, which is indicated by the double-arrows in sketch b).

Using this techniques, all possible paths from a to b can be determined with only $2 \times G^{p/2}$ calculations, which would reduce the number of comparisons from some millions

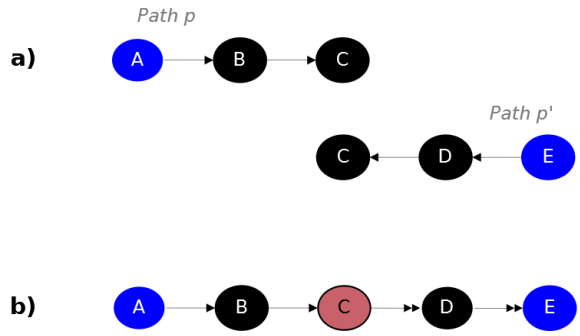

Figure 9.6: Path combination after bidirectional breadth first search.

to $2 \times 100^2 = 20,000$. However, since the connector nodes have to be determined, for each $p \in P_A$ and each $p' \in P_B$ it has to be checked whether (p, p') share such a node. Assuming that is holds $|P_A|, |P_B| = 10,000$, this step requires $10,000^2$ (100 million) comparisons, which is equivalent to the theoretic number of comparisons if the original breadth-first search was used. In this case, the bidirectional would not reduce any effort.

To circumvent this issue, all target nodes of $p \in P_A$ are stored in a hash set H. To find the connector node, all target nodes of $p' \in P_B$ are iterated and it is only checked whether H contains b. If this is the case, the respective path object is retrieved and the full path is built. In the above example, this means that $10,000$ contains-operations have to be carried out.

The contains-operators of hash sets is extremely fast. Experiments showed that the operator requires only fractions of milliseconds to determine whether H contains a specific node concept, even if H contains some $10,000s$ of concepts. As a consequence, this adaptation makes the bidirectional search much faster than the original approach.

In detail, the algorithm to find paths up to length 4 works as follows:

1. All direct paths of concept A are calculated and stored in a set P_A. The target concepts of each path are stored in a hash set $H(A)$. If it holds $B \in H(A)$, a path of length 1 was found.

2. All direct paths of node B are calculated and stored in P_B. The target concepts of each path are stored in a hash set $H(B)$. If there is a concept node C for which holds $C \in H(A), C \in H(B)$, there is a path of length 2.

3. All outgoing paths of length 2 are calculated for A and added to P_A. The target concepts of each path is stored in a hash set $H'(A)$. If there is a concept C for which holds $C \in H'(A), C \in H(B)$, there is a path of length 3.

4. Finally, $H'(B)$ is calculated. If there is a concept C for which holds $C \in H'(B), C \in H'(A)$, there is a path of length 4.

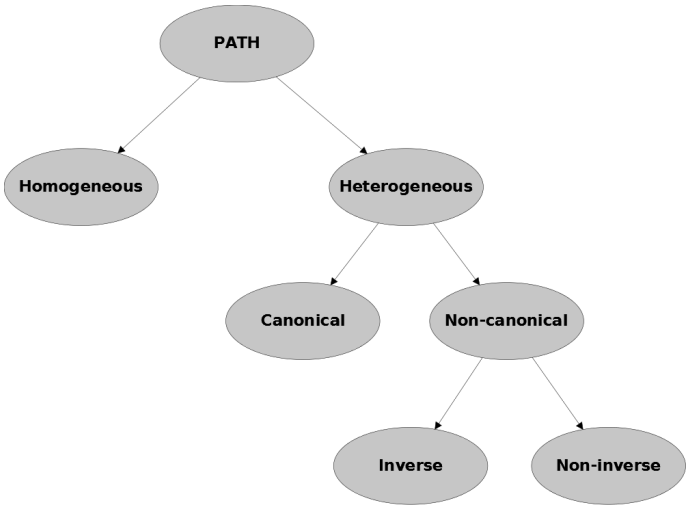

Figure 9.7: Overview of the different path types and the used terminology.

9.3.3 Path Type Calculation

One important part of SemRep is to calculate the relation type of paths consisting of more than one edge. Let us assume there is a path of length 2 between nodes X, Y, Z with two relations r_1, r_2 in between so that the path looks as follows: $X\ r_1\ Y\ r_2\ Z$. Given these pieces of information, the difficulty of the path type calculation is to determine the path type r. This calculation is not always trivial, as it can hold $r = r_1$, $r = r_2$ or $r \neq r_1 \neq r_2$, depending on the type of r_1 and r_2. In [62], the authors present a simple approach for indirect path type resolution, though they do not regard the types `has-a` and `part-of` and do not provide any proofs or reasons for their deductions.

In the following section, we will discuss how the path type r can be determined from the relation types r_1, r_2 and under which circumstances the type cannot be determined. Given a relation type r, we will denote the inverse relation type by r^{-1} for some illustrations. To facilitate the discussion of the different cases that need to be regarded, we define several path categories and use the terminology which is depicted in Fig. 9.7.

Homogeneous Path

If it holds $r_1 = r_2$, p is called a *homogeneous path*. Since all relation types are transitive, it holds $r_1 = r_2 = r$. Thus, homogeneous paths are the simplest paths w.r.t. path type calculation. A typical example is a path leading up the lexicographic taxonomy.

$$\text{SUV } \texttt{is-a}\ \text{Car } \texttt{is-a}\ \text{Vehicle} \rightarrow \text{SUV } \texttt{is-a}\ \text{Vehicle}$$

If it holds $r_1 \neq r_2$, p is called a *heterogeneous path*. There are two forms of heterogeneous paths: canonical and non-canonical paths.

Canonical Path

If it holds $r_1 = \texttt{equal}, r_2 \neq \texttt{equal}$ (or vice versa), p is called a *canonical* path. The type `equal` can be seen as the identity (neutral) element in a path. Any relation type r combined with `equal` leads to r, i.e., `equal` does not change the path type. Thus, it holds $r = r_2$, as the following examples shows:

$$\text{Automobile } \texttt{equal } \text{Car } \texttt{is-a } \text{Vehicle} \rightarrow \text{Automobile } \texttt{is-a } \text{Vehicle}$$

If it holds $r_1 \neq r_2 \neq \texttt{equal}$, we call p a *non-canonical* path. There are again two forms of non-canonical paths: non-inverse paths and inverse paths.

Non-inverse Path

If p is a non-canonical path and it holds $r_1 \neq r_2^{-1}$, we call the path *non-inverse path*. This means that one type is `is-a` resp. `inverse is-a` and the other type is `has-a` resp. `part-of`. Generally speaking, the path consists of one generalization and one aggregation relation. In the field of semantics, it holds that aggregation has a higher binding strength compared to generalization and that the aggregation type prevails. Thus, r is identical to the aggregation type and is either `has-a` or `part-of`. The following examples illustrates this:

$$\text{Laptop } \texttt{is-a } \text{Computer } \texttt{has-a } \text{CPU} \rightarrow \text{Laptop } \texttt{has-a } \text{CPU}$$

One important question is why the aggregation type prevails against the generalization type and whether this assumption always leads to sensible results. As already shown in Section 3.2.1, concepts are defined by different properties and a `has-a` or `part-of` relation is such a typical property. The `has-a` relation is thus used to define (or specify) the concept c and since properties are inherited by all sub-concept c', the `has-a` relation holds for all sub-concepts as well. Consider the example between *Laptop* and *CPU* as shown in Fig. 9.8. The concept computer is defined by the property "has-a" CPU. Any sub-concept below computer, such as laptop, inherits this property so that the `has-a` relation holds as well. By contrast, the generalization type cannot hold, since the `has-a` relation leads to another branch of the taxonomy as illustrated in the picture. The two branches are independent, i.e., CPU and Computer do not necessarily share any properties. For this reason, there is no semantic foundation to infer an `is-a` or `inverse is-a` relation between *Laptop* and *CPU*, but only the aggregation type holds.

However, an important problem arises if the path leads to a super-concept of c, which defines the `has-a` or `part-of` relation. Consider the following, slightly adapted example:

$$\text{Machine } \texttt{inverse is-a } \text{Computer } \texttt{has-a } \text{CPU} \rightarrow \text{Machine } \texttt{has-a } \text{CPU}$$

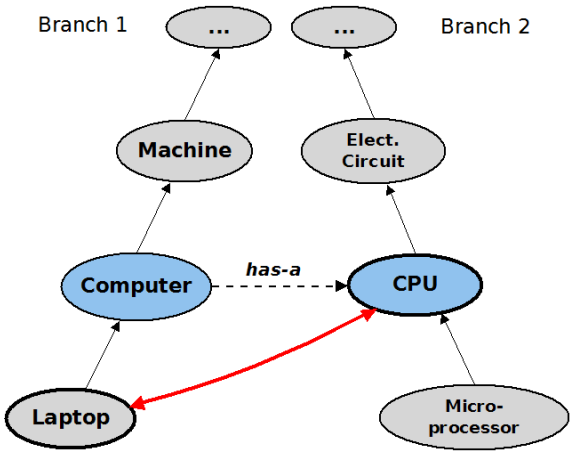

Figure 9.8: Path type calculation across branches.

According to the previous argumentation it holds <*Machine*, `has-a`, *CPU*>. This statement is no longer universally valid, because the "has-CPU" property is only defined for computers, not for concept being more general than computer. The statement is not completely false either, as some machines contain a CPU, but it should be rather expressed like "There are machines that have a CPU (e.g., computers)". We call such a case *Loss of Generality* and as we will illustrate further below, it occurs in 4 of 8 possible combinations of r_1 and r_2. Loss of Generality is regarded in the confidence calculation, i.e., paths are scored lower if this phenomenon occurs. Still, it has to be remarked that most simple `part-of` and `has-a` relations are not generally valid in the first place (as described in Section 3.2.3). For instance, the relation <*cellar*, `part-of`, *house*> already suffers from Loss of Generality, since there are houses without a cellar and vice versa.

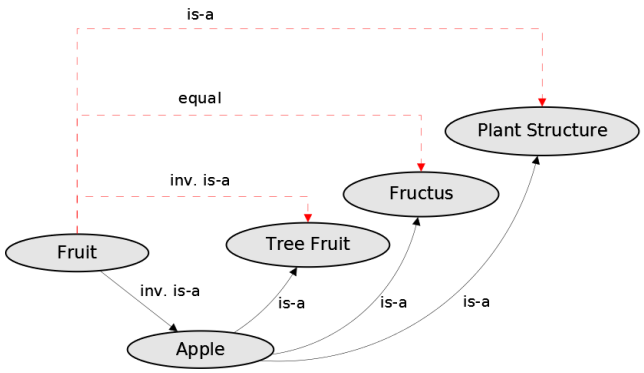

Figure 9.9: Illustration of the specific case `inverse is-a` + `is-a`.

Inverse Paths

Eventually, if it holds $r_1 = r_2^{-1}$ we call such a path an *inverse path*. There are 4 forms of inverse paths:

1. X `is-a` Y `inverse is-a` Z

2. X `inverse is-a` Y `is-a` Z
3. X `part-of` Y `has-a` Z
4. X `has-a` Y `part-of` Z

Case 1 is the classic co-hyponym case, in which X and Z are siblings. They share the properties of the parent node Y, but have some additional, distinct properties. The relation type is `related`. Let us assume that there is a root concept C at the hierarchy level 0 and Y is situated at level n. The greater n is, the greater is the semantic overlap and the greater is the similarity between X and Z.

Case 2 is the inverse of Case 1, but the relatedness between X and Z cannot be determined in this case. It holds obviously $Y \subset X$ and $Y \subset Z$, but the relation between X and Z cannot be determined by means of semantic deduction. As also depicted in Fig. 9.9, the following examples lead to different results:

Fruit `inverse is-a` Apple `is-a` Plant Structure $\rightarrow$ Fruit `is-a` Plant Structure

Fruit `inverse is-a` Apple `is-a` Tree Fruit $\rightarrow$ Fruit `inverse is-a` Tree Fruit

Fruit `inverse is-a` Apple `is-a` Fructus $\rightarrow$ Fruit `equal` Fructus[76]

Since the type cannot be uniquely determined, SemRep returns `undecided` in this case.

Case 3 is similar to case 1, but Y does not describe common properties that X and Z inherit. Instead, Y can be interpreted as a place where X and Z may co-occur. SemRep concludes the `related` type in this case, but in contrast to case 1 this decision seems less confidential. The following example shows a case where `related` seems to be an appropriate decision:

Student `part-of` School `has-a` Teacher $\rightarrow$ Student `related` Teacher

However, in the slightly modified example this decision seems much more vague.

Student `part-of` School `has-a` Janitor $\rightarrow$ Student `related` Janitor

The fact that two independent concepts may occur at the same place does not always suggest a sensible `related` relation. In the student-janitor-example, "unrelated" would be the best decision, as there is no sensible relationship between a student and a janitor.

Finally, **case 4** is similar to case 2. It cannot be uniquely resolved, so SemRep returns `undecided`.

[76] Fructus is the Latin name for 'Fruit'.

		r_2				
		equal	**is-a**	**inv. is-a**	**part-of**	**has-a**
r_1	**equal**	*equal*	*is-a*	*inv is-a*	*part-of*	*has-a*
	is-a	*is-a*	*is-a*	*related*	*part-of*	*has-a*
	inv. is-a	*inv. is-a*	–	*inv is-a*	*part-of*	*has-a*
	part-of	*part-of*	*part-of*	*part-of*	*part-of*	*related*
	has-a	*has-a*	*has-a*	*has-a*	–	*has-a*

Figure 9.10: Relation type matrix for paths of length 2.

Conclusions

In this subsection, we have gradually described the different combinations of r_1 and r_2 within an indirect path, and what semantic path type r can be concluded. The results are the work of a profound semantic study of concepts within a lexicographic hierarchy. An overview of the input types and respective results is illustrated in Fig. 9.10.

Dark green states contain the relation types of homogeneous paths and light green states the relation types of canonical-heterogeneous paths. Non-inverse path types are marked yellow and orange, with orange describing path types with Loss of Generality. Finally, inverse types are marked blue, while two of the four combinations are undefined.

To resolve paths of length greater than 2 works analogously. As an example, consider the following path of length 3: (*SUV* `is-a` *car* `equal` *automobile* `is-a` *vehicle*). In this case, the first two relations are resolved just as illustrated in Figure 9.10. Thus, the two relations <*SUV*, `is-a`, *car*> and <*car*, equal, *automobile*> lead to the relation <*SUV*, `is-a`, *automobile*>. Now, the original path is reduced to (*SUV* `is-a` *automobile* `is-a` *vehicle*). This path of length 2 can be resolved again using the illustrated approach. According to Fig. 9.10 it can be directly concluded: <*SUV*, `is-a`, *vehicle*>.

Combinations with type `related` are not handled by the approach and are generally not allowed, except for the type `equal`. In this case, it holds `related` + `equal` $\rightarrow$ `related`; otherwise `undecided` is returned, because no type can be reasonably concluded then.

9.3.4 Confidence Calculation

Since SemRep is able to find more than one path from a start node to a target node, path objects need to be scored in order to designate the most relevant path. SemRep uses a scoring function that calculates a path score within the interval $[0, 1]$. The path having the highest score is the result path, which is used to answer the query.

The scoring function considers four parameters:

1. The path length.

Parameter	Value
c(r) = WordNet	0.97
c(r) = Wikipedia	0.80
c(r) = Wikipedia Redirects	0.94
c(r) = Wikipedia Field Ref.	0.80
c(r) = UMLS	0.98
c(r) = OpenThesaurus	0.95
c(t) = equal	0.97
c(t) = is-a / inv. is-a	0.95
c(t) = has-a / part-of	0.86
INV	0.20
LoG	0.12
WP	0.10

Table 9.3: Values for the types and resource confidences used in the scoring function.

2. The resource of each relation within the path.
3. The relation type of each relation within the path.
4. Some path-specific features.

The notion for path scoring is as follows: Short paths are always more relevant than longer paths and should always have a higher score. Relations from manually created resources like WordNet are more likely to be correct than from automatically generated resources (Wikipedia relations) and should have a higher confidence. The same holds for the types `equal`, `is-a` and `inverse is-a`. According to experiments, these types are more likely to hold than `part-of` and `has-a`, and should have a higher impact. Finally, inverse paths are generally more prone to errors and less precise than other types. Therefore, the score of an inverse path should be reduced by some constant factor.

Let r be a relation within the path, $c(r)$ be the confidence of the resource where the relation comes from and $c(t)$ the confidence of the specific relation type of r (it holds $c(r), c(t) \in [0, 1]$). The resulting scoring function is:

$$s = c(r_1) \cdot c(t_1) \cdot c(r_2) \cdot c(t_2) \cdot ... \cdot c(r_n) \cdot c(t_n) - INV - LoG + WP \tag{9.1}$$

In the scoring function, *INV* is a parameter used for inverse paths and *LoG* is a parameter used if Loss of Generality occurs. WP is used if the path is a sole "WordNet path", i.e., if it only contains relations from WordNet. Such a path is considered to be more reliable, as WordNet is a high-quality resource and the path score is therefore increased.

The default values used in the repository are shown in Table 9.3. They proved to achieve the best results in the evaluation. In the following, let us assume there is a path of length 2: *car* `equal` *automobile* `is-a` *vehicle*. For the sake of simplicity, let us as-

sume that both relations originate from Wikipedia. The score is calculated as follows: $s = (0.8 \cdot 0.97) \cdot (0.8 \cdot 0.95) = 0.59$.

There is a minimal score that has to be achieved so that the path will be accepted. By default this value is 0.4. If a path reaches a score greater than 1, which is possible because of the WP-factor, the score is trimmed to 1.

9.3.5 Post-processing

In some cases, SemRep does not find any path from the source concept to the target concept, although a semantic type could actually be discovered. One example is the correspondence *(wine region, location)*. Both concepts are contained in the repository, but no path could be found to determine any relation type. The correct type would be `is-a`, as *location* is more general than *region* (which is again more general than *wine region*).

Since *wine region* is an open compound, SemRep performs Gradual Modifier Removal on the concept in case that the query could not be answered. This is a step of post-processing in which the same technique is used as in the preprocessing step. Post-GMR gradually removes the modifiers of the open compound and executes the query again. In the case of *wine region*, the word is reduced to *region* and SemRep returns *region* `is-a` *location*. Applying the same reasoning as in the preprocessing step, it can be concluded that *wine region* `is-a` *location*, as *wine region* is more specific than *region*. Thus, in some cases this post-processing step can correctly resolve correspondences even if no path from source to target concept can be found.

If still no valid path between source node and target node can be determined, a heuristic method is used that compares the degree of the two nodes. If it holds $degree(source)/degree(target) > 1.5$, the source node is assumed to be more general than the target node and `inverse is-a` is concluded. If it holds $degree(target)/degree(source) > 1.5$, the source node is assumed to be less specific and `is-a` is returned. The value of 1.5, which we call *Minimal Node Degree Quotient*, is a configurable parameter in SemRep. This technique can slightly increase the quality of SemRep, though no sensible conclusion can be made for two nodes that have a similar degree. Unlike the Word Frequency Strategy introduced in the STROMA part, which is quite a similar approach, this technique is less error-prone and in most cases a node of high degree is indeed more general compared to a node of lower degree.

9.4 Technical Details

SemRep is written in Java and provides an API for query execution as well as a terminal-based UI for more specific tasks like testing, evaluation, statistical analyses and configuration. The repository data can be loaded in one step (bulk load) or selected resources can be loaded manually. For examples, if a medical mapping is processed, it is thus possible to only load the UMLS data.

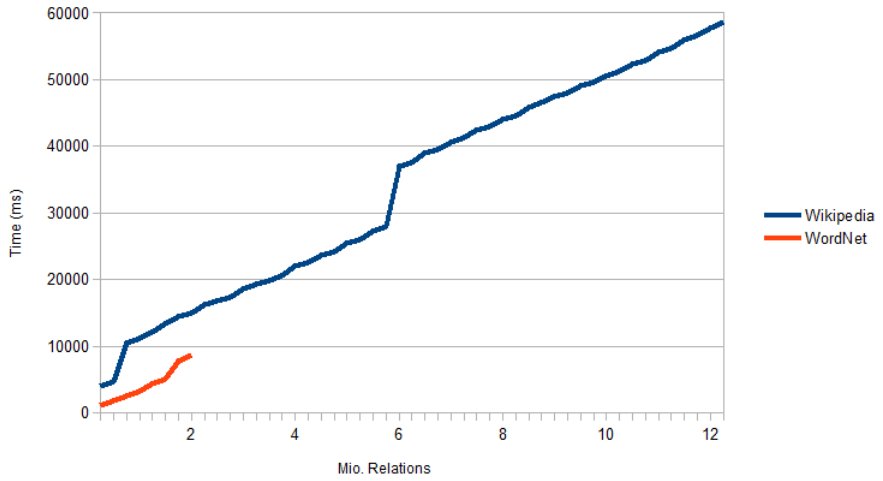

Figure 9.11: Loading time to load a specific number of relations into SemRep. The used interval was 250,000 relations.

SemRep is rather independent from the data that is imported. Though preprocessing, filtering and post-processing is applied to the import data, these techniques can be disabled at any time. Technically, this makes SemRep also applicable for data sets other than lexicographic-semantic relations, e.g., taxonomies about genes, proteins or chemical formulas, where concepts may be simply represented by a numerical id. Since SemRep is strongly tailored to the field of schema and ontology mapping, further adaptations may be necessary to use SemRep in such more specific domains, though.

Reading the complete data sets from the input files takes about 45 seconds on a Windows 2008 server possessing two Intel Xeon E5540 CPUs (2.53 GHz). Loading relations into the repository has a near-linear time complexity. Fig. 9.11 shows the loading time in ms to import different numbers of relations from Wikipedia (the original data set containing more than 12 million relations) and WordNet (1.9 million relations). At some points, there is a noticeable delay in both resources which cannot be simply explained by the possible occurrence of some specific or complex relations (such as long words, words with many outgoing relations, etc.). Instead, it seems that the system has to allocate more memory or re-organize the hash map which requires a considerable amount of processing power and thus interrupts the reading process. Increasing the heap space or specifying the expected number of relations when the hash map is declared does not change this behavior in any way and the interrupts have to be accepted as part of the import process. Still, the overall time complexity is near-linear, which facilitates the inclusion of further resources.

To run SemRep, about 3 GB of main memory are required. The required space depends highly on the amount of data that is loaded. Without loading any resource data, only a few MB of RAM are required.

10 Quality Improvement

Integrating a large number of relations from multiple resources into a relatively simple graph-based repository does not come without any difficulties and complications. First of all, there are irrelevant and potentially erroneous relations in the Wikipedia data set so that the existence of a fully correct repository cannot be assumed. Secondly, homonyms occurring in SemRep can impair the query execution. The effect is that even very simple queries may fail, i.e., that a false relation type is returned. In this chapter, we will discuss several issues referring to the repository quality, as well as different possibilities to solve or at least alleviate them. We will first discuss the problem of homonyms in SemRep (Section 10.1). Subsequently, we will discuss problems caused by so-called universal concepts (Section 10.2) and entities (Section 10.3). In Section 10.4, we illustrate how false relations could be determined in SemRep and what obstacles and difficulties have to be overcome. Finally, in Section 10.5 we sum up important insights gained in this chapter.

10.1 The Homonym Issue

As already described in the previous chapter, a given term x is represented by exactly one node. Therefore, homonyms (the different meanings of such a term) cannot be distinguished this way. They are an important issue of SemRep, because homonyms often express completely different things, as a *mouse* in computer technology is something completely different compared to a *mouse* in biology. However, such a disambiguation between two equally spelled, semantically different concepts is extremely difficult to achieve if various resources are combined, which do not use a unique classification of their concepts or no classification at all. Consider the concept *mouse (animal)*. Technically, SemRep could use a manually defined category `<Biology>` to distinguish biological

terms from other categories. However, nothing in the mouse article of Wikipedia indicates that *mouse* belongs to the category of `<Biology>`. Only by extracting the categories of this article and walking the Wikipedia category tree upwards there might be a chance to reach a category named `<Biology>`, indicating that this article refers to the field of biology. In WordNet, however, nothing indicates a relation to the biological field either. The WordNet gloss defines a mouse as a small rodent, which is similar to the Wikipedia definition. There is no easy way to determine that mouse refers to the biological domain and not to one of the many other imaginable categories.

Further on, to which category should a *computer mouse* belong? *Computing* might be suitable, or *devices* or *hardware*. But again, how can we know when we parse such an article or import it from WordNet, that it refers to this category? And let us assume that we have to deal with much more specific terms, like *table* in the field of databases. This concept might fit best in a category like *abstract*, but again it seems rather unlikely that the Wikipedia article contains any plausible hint indicating that a database table belongs to the category `<Abstract>`.

Therefore, it seems pointless to devise an internal classification system to distinguish homonyms. Another possibility is to simply store the concepts redundantly, e.g., to create different mouse nodes for each Wikipedia mouse article, and for each WordNet sense of *mouse*. However, if we encounter the term *mouse* in a mapping, how could we possibly decide which mouse concept is meant? And even if we knew it was a biological mapping, how can we know which of the nodes in SemRep refers to this biological context?

Some homonyms even occur within a specific category, although they express quite different things. Consider the term *car*, which originates from the word *carriage* and has different meanings in the field of *transportation*. According to Wikipedia, there are 3 conceptual senses related to this field, namely, motor car (automobile), railroad car and trolley car.[77] This means, that a concept node *car (transportation)* still comprises 3 different senses and would not help much to solve the homonym issue in this case.

Eventually, if we had a near-perfect classification and assignment of concepts, and if we would know to which domain a given mapping refers to, this feature impairs the extendability of the approach. If a new resource is integrated, which would be a simple set of triples $(word_1, type, word_2)$, how can we know to which category each relation refers? We cannot expect this from the resource, since some resources do not provide such information, and even if they would, they do not necessarily need to match the classification used in SemRep.

Thus, there is no easy way to handle this homonym problem. It would be possible to add some semantic structuring to SemRep, e.g., by exploiting the WordNet senses or Wikipedia categories. As an example, for each term t, nodes $t_1, t_2, ..., t_n$ could be created if the term occurs in n different WordNet synsets. New resources which would be integrated in SemRep have to be matched against the already existing concepts then and could be also integrated in this semantic structure. Instead of a repository consisting of nodes and edges, an advanced data structure might be also useful, e.g., a data structure being similar to WordNet. This means that SemRep would consist of interlinked clus-

[77] `http://en.wikipedia.org/wiki/Car_(disambiguation)`

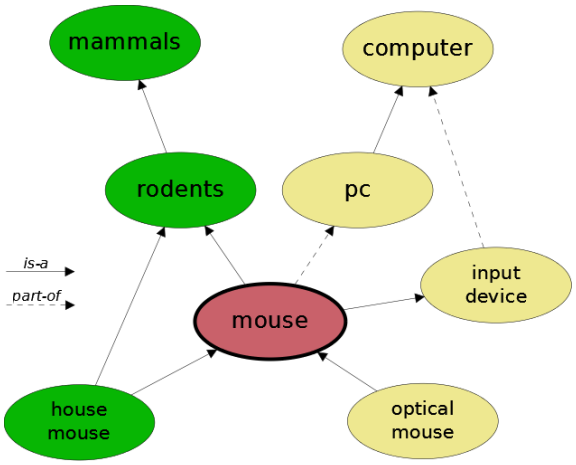

Figure 10.1: Illustration of the homonym *mouse* with relations to different semantic categories.

ters, which comprise a set of synonymous terms. Integrating new resources implies the correct assignment of the terms they contain to the existing synsets, or the automatic creation of new synsets. Though it is not impossible to deal with homonyms in general, it is certainly impossible to obtain a 100 % correct and reasonable organization of concepts, just as it is in the manually developed WordNet. Whether such an attempt of homonym handling can really improve the overall quality of SemRep is thus difficult to decide and should be investigated in future work.

Right now, SemRep is unable to distinguish any form of homonyms. However, since SemRep is designed for the matching and mapping domain, the absence of word sense disambiguation is far less impairing as it may appear at first glance. Consider the example depicted in Fig. 10.1. The homonym node *mouse* is colored red and refers both to the biological domain (green nodes) and computing domain (yellow nodes).

Let us assume that the repository is asked which relation holds between *house mouse* and *input device*. There is a path from *house mouse* to *input device* across the *mouse* concept and the repository would erroneously return that house mice are input devices (`is-a` type). However, in the schema and ontology mapping domain such a query seems unlikely, because it would entail that there was a correspondence *(house mouse, input device)*. This means that a biological ontology must have been matched with an ontology about computers or input devices, which seems unlikely. In the vast majority of all cases, ontologies are matched from the same domains, so either two biological ontologies or two technical ontologies. Consequently, the homonym problem normally does not pose any serious problem. If a correspondence *(mouse, mammals)* is handled, it does not matter that *mouse* is related to concepts of different domains, as long as SemRep provides a semantic path from *mouse* to *mammals* that does not lead through non-biological domains (which normally is the case).

Of course, if two general purpose ontologies are matched, that contain concepts from versatile domains, homonyms can indeed impair the query result. This is a special situation in which SemRep may produce errors. As illustrated above, there are even homonyms within a specific domain (as the word *car*), which can also lead to errors in typical match tasks. In the evaluation (see Chapter 11), erroneous results were usually not caused by homonyms, though, but by the absence of relevant links between concepts, and by imprecise results caused by indirect paths (e.g., SemRep suggesting the type `related`, though `part-of` was correct).

To conclude this discussion, the homonym issue cannot be ignored when integrating large amounts of relations without knowing to which semantic category they belong, but there is no simple solution to this problem. Experiments showed that in most mapping scenarios the number of erroneous results impaired by homonyms was relatively little, so that the homonyms do not cancel out the overall benefits gained from SemRep.

10.2 Universal Concepts and Related-Types

Each language consists of very general concepts that comprise a large number of sub-concepts. Such concepts are also found in lexicographic resources and in the extracted Wikipedia relations. They usually form the root concepts of these resources or are within a close distance to them (e.g., direct sub-concepts). We call such concepts *Universal Concepts*, though there is no clear demarcation between a universal concept and an usual concept. Typical examples are *action*, *artifact*, *condition*, *entity*, *event*, *fact*, *object*, *structure*, *thing* or *unit*. In lexicographic resources like WordNet, artificial concepts are also found, mostly to allow a more detailed structuring of the lexicographic data. For instance, WordNet contains concepts like *imaginary place*, *physical entity* or *whole thing*.

Such universal concepts are normally meaningless, i.e., they are general enough that almost any concept could be part of them. For example, a statement like *car* `is-a` *object* carries only little (if any) semantic information. At the same time, universal concepts can seriously impair the query result. If two concepts are linked by a universal concept, imprecise or false relation types can be returned. For example, the relation *building* `is-a` *object* `inverse is-a` *car* would lead to the result that *building* and *car* are `related`.

In fact, universal concepts often lead to the type `related`, which is the most unspecific of all types. Though this type may not always be false, it is usually not the most relevant type. Given the two sample concepts *door* and *house*, which constitute a typical `part-of` relation, SemRep returns the type `related` if the maximal path length is set to 2:

door `is-a` *construction* `inverse is-a` *house*

Only if the path length is set to 3, the correct type is determined:

door `part-of` *wall* `part-of` *building* `inverse is-a` *house*

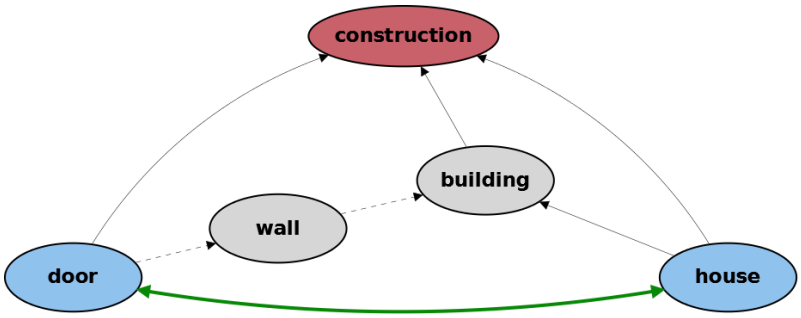

Figure 10.2: Repository extract to illustrate the determination of the relation type between *door* and *house* (green arrow).

Fig. 10.2 illustrates the respective extract of SemRep. To determine the relatedness between *door* and *house*, SemRep will only find a path across the universal concept *construction* if the maximal path length was 2. This results in the type `related` (straight arrows indicate `is-a` relations and dashed arrows indicate `part-of` relations). Only if the maximal path length is increased to 3, the `part-of` relation is found.

SemRep tries to keep the number of such `related` results low by reducing the score of cohyponym paths, however, `related` paths are often the only path that can be determined if the configuration is set to a maximal path length of 2, which is caused by the absence of shorter, more expressive paths in the repository. In order to find the correct path, a higher configuration has to be chosen, but experiments showed that the results are generally worse than in a lower configuration, since longer paths have an increased risk to become erroneous (e.g., because of homonyms).

To alleviate this problem, SemRep is able to block relations containing universal concepts when importing the data. It exploits a manually created list for typical WordNet and Wikipedia universal concepts that is mostly based on tests and experiences. This list contains about 80 terms that seem irrelevant and potentially impairing. Any relation in which such a concept participates will be rejected. This universal concept filter reduced the number of concepts in the repository only slightly. The number of relations can be considerably reduced, though, since universal concepts are usually nodes of a high degree.

Some concepts are right between a universal concept that must not be part of the repository and a common concept that should be part of the repository. Examples are *location*, *method*, *person*, *process* or *technique*. For instance, the concept *location* appears in some gold standards that were used to evaluate STROMA, so removing it from SemRep would be disadvantageous. Obviously, taxonomies can contain some kind of universal objects that need to be matched and for this reason the number of excluded concepts was kept rather low. In turns, this means that erroneous or imprecise results cannot entirely be prevented. In the above *door–building* example, it was decided to do not exclude the term *construction*, as it is not general enough to be removed from the repository. SemRep can only answer this query accurately if a maximal path length of 3 or higher is used.

10.3 Entity Filtering

Although most entities from Wikipedia relations were filtered during the relation extraction (category filtering), SemRep still contains many entity concepts that are extraneous for the schema and ontology mapping domain.

There are different possibilities to further reduce the number of entities in the repository. First, SemRep uses regular expressions to filter words that are not in accordance with the rules of English morphology (word formation). English concepts can only consist of letters (a – z), hyphens and spaces. There are a few exceptions, like 'é' as in rosé or cliché, but other letters will not be found in common English words. Consequently, a given concept word must not contain any numbers, typographical characters like periods, commas and braces, or foreign letters and symbols. We call this filter technique *Morphological Analysis*. Using such a filter can already block many entities like *C64*, *São Paulo* or *A.C.G.T.* However, some entities stick to all rules of English morphology, like *Paris*, *Leipzig University* or *Casio*. Besides, many words are concepts and entities at the same time, e.g., the word *car* also refers to villages, communes, music albums and movies.[78]

Since the Wikipedia relation extraction approach is case-insensitive, which facilitates the sentence parsing and processing, the case of extracted words cannot be analyzed either. Besides, words extracted from the beginning of a sentence always start with a capital letter. In the article "Leipzig is a city in the federal state of Saxony...", such a method could not determine whether Leipzig is a sub-concept of *city* or an entity (instance).[79]

Another option is to compare the extracted concepts with another resource. Such a resource is only useful if it is restricted to concepts (like a dictionary) and if it is very comprehensive. In the context of this research, concepts were compared with Wiktionary, which is a typical multilingual dictionary.[80] All concepts from the Wikipedia data sets were iterated and each concept was considered an entity if...

1. the concept was a single word and was not contained in Wiktionary,
2. the concept was an open compound word and one of the words it consists of was not contained in Wiktionary.

If a word was confirmed to be an entity, all relations of the Wikipedia data set in which it participated were removed. The processing time for all concept terms was about 60 hours and was achieved by checking whether a Wiktionary URL page exists for a specific concept. To reduce the execution time and URL requests, WordNet was used for a first lookup before Wiktionary was asked. If the word to be checked was already contained in WordNet, it was automatically considered to be a concept.

This entity filtering approach could reduce the number of entities by about 65 %, though about 3 % of concepts were also removed. These were specific words that are not contained in Wiktionary. Additionally, a considerable amount of biological terms like species

[78] `http://en.wikipedia.org/wiki/Car_disambiguation`
[79] `http://en.wikipedia.org/wiki/Leipzig` (March 2015)
[80] `www.en.wiktionary.org/`

was removed, which are also not contained in Wiktionary. Such species can be considered both concepts and entities. The remaining relations form a new resource *Wikipedia-Filtered*, which co-exists in parallel to the original *Wikipedia* data set. SemRep enables both users and match tools to work with either of the two sets. However, evaluations showed that if the Wikipedia-Filtered set was used, the mapping quality did not change in any way and both recall and precision remained the same in each experiment. Though the loading and execution time was somewhat reduced, there was no qualitative effect. This insight encourages the assumption that entities in SemRep, just as homonyms, do not seriously impair the query processing.

10.4 False Relations

False relations like *car* `has-a` *vehicle* are the most serious problem of SemRep. They are usually the result of the automatic Wikipedia relation extraction, which cannot achieve a 100 % precision. There are, however, also imprecise relations resulting from WordNet. For example, WordNet contains the two relations *brain* `equal` *head* and *brain* `part-of` *head*. Only the second relation seems sensible, while the first one is erroneous. Given the high number of relations, it is impossible to validate them manually. However, there are hardly any feasible methods to decide whether a given relation is semantically correct, i.e., whether the two concepts are indeed related, and if so, whether the relation type is correct. Thus, a relation can be completely false (e.g., *head* `is-a` *tree*) or possess the false type (e.g., *head* `equal` *brain*).

In this section, we will first introduce an approach to verify relations using a search engine (Section 10.4.1). In Section 10.4.2, we discuss the technical feasibility of this approach and carry out a small experiment illustrating the problems related to this approach in Section 10.4.3. In Section 10.4.4, we condense important insights of this section.

10.4.1 The Search Engine Approach

One possible technique to confirm the correctness of a relation is to translate it into a natural language sentence, apply it to a search engine and count the number of results that are returned. As some search engines have practically indexed the entire world wide web, it seems likely that the search engine would return some results if this sentence is sensible, as it might then occur somewhere among the billions of websites. This approach was already used in [65], while a similar approach sends the words A, B and "A, B" to a search engine and calculates the likelihood that A and B are related (which is not a convenient method to verify the relation type, though) [59].

Google seems to be the most powerful search engine for such an approach. It allows to search for exact matches (phrases), it has the highest number of indexed web pages and it provides the number of results to the user. For example, the expression *"car is a vehicle"* returns many results, while *"car is part of a vehicle"* returns no results. This suggests that *car* and *vehicle* are in an `is-a` relation.

For such an approach, each relation has to be converted into a natural sentence. This is no easy undertaking, because relation types can be transformed into different expressions. For example, the relation *CPU* `part-of` *computer* can be translated into *"CPU is part of a computer"* or *"CPU is part of computers"* or even *"Computers consist of CPUs"*. Some words are more often used in plural, while others are more often used in singular. For example, the most results for the relation between *sweater* and *clothing* are obtained if the first word is used in plural and the second word is used in singular, so *"Sweaters are clothing"*. Note that the first word (subject) determines how the verb is conjugated; a sentence like *"sweaters is a clothing"* is grammatically incorrect and returns only few results.

Besides, some expressions need a determiner. The correct relation *"Door is part of a house"* returns 20 results on Google, while the false relation *"Door is a house"* returns 109 results. Thus, it seems that <*door*, `is-a`, *house*> is the correct relation. However, the relatively high number of results for the second query is solely caused by not regarding the context in which the sentence fragments occur. Most results do not express any `is-a` relation between *door* and *house*, but the sentence fragment appears in different contexts, like *"next door is a house that will charm you"* or *"to the right of a front door is a house filled with..."*. Only if the determiner "a" is used (*"a door is a house"*), the number of results remains low, although Google still returns 3 results.

Thus, for each relation type, different expressions have to be used. Sometimes, it is also sensible to use partitives within the sentence, e.g., *"A lowboy is **a kind of** furniture"*. Combined with plural and singular form and different determiners, the number of expressions per relation type can become fairly high.

Given a relation $r = (w_1, t, w_2)$ that is to be verified. If the search engine returns much more results for a type t' ($t \neq t'$), the relation type of r will be changed from t to t'. If a search engine does not return results for any type, r will be considered to be false and removed from the data set. Even though this approach could detect false relations and thus increase the precision, there is a general risk to remove correct relations if they are too specific to be found by a web search engine. This results in a lower recall.

10.4.2 Time Complexity Analysis

Let us assume we want to verify the 2.8 million extracted relations from Wikipedia, so the entire data set without the Wiktionary filtering. Let us assume that we build about 10 different expressions per type. We have 3 basic relation types (`equal`, `is-a`/ `inverse is-a` and `part-of`/ `has-a`), so we need 30 expressions per relation to calculate what type obtains most results. A Google query can be processed within about 0.5 seconds. If we would check all relations, the execution time for the queries would be: $2,800,000 \times 30 \times 0.5s = 42 \times 10^6 s$.

This execution time is not only unrealistic (it is more than a year), there is no search engine that would accept so many queries. In fact, search engines like Bing or Google only allow a few 100 queries per day. Using search engines are thus ineligible to validate the Wikipedia data set, even if a cluster of computers would be used.

An alternative approach is to evaluate only the subset of relations that are needed by the repository to process a given mapping. Let us assume that a given mapping contains 300 correspondences and that about 500 different noun words appear in the mapping (in the source and target concepts). Let us assume that all of these noun words are also contained in SemRep. As a preparatory step, SemRep could evaluate the direct relations of those noun words, as the repository will not access any other relations if the maximal path length configuration is set to 2. Let us assume that the average degree of a node is about 100, so $500 \times 100 \times 30$ relations have to be calculated if we still assume 30 expressions per relation. This number is still too high to make the approach feasible. The execution time would still be several days, which is not even close to an acceptable mapping preprocessing time.

As a result of this analysis, search engines can at most be used to verify a few selected relations, but never all relations of the repository or all theoretically relations needed to process a given mapping. Even if additional techniques would be used, such as the concatenation of queries by or-operators, the number of necessary queries is too high for the proposed undertaking.

10.4.3 Correspondence Evaluation

Correspondence evaluation seems to be the only feasible scenario for the exploitation of web search engines. If the number of expressions is reduced to 5 per type and a mapping consists of 300 correspondences, $3 \times 5 \times 300 = 4.500$ queries have to be formulated which means an execution time of about 38 minutes. This time can also be reduced if obviously correct correspondences like *(car, `equal`, car)* are skipped. There are even search engines that allow such a number of queries, e.g., *Faroo* allows 100,000 queries per month.[81]

In a very simple test, 20 correspondences from two gold standards were used and evaluated with search engines. They were differently labeled and contained both quite general concepts (e.g., composer `is-a` person) and more specific concepts (e.g., *panel screens* `is-a` *room dividers*). All correspondences were correct and expressed true `is-a` relations, i.e., the approach had to confirm each correspondence. Different search engines were used, among them *Google*[82], *Bing*[83], *Yandex*[84] and *Startpage*[85].

The result was rather poor. Google, which obtained the best results, confirmed only 65 % of the correspondences, even though all correspondences were correct. This means that 35 % of correct correspondences would have been rejected, which is unacceptable. As a result of this analysis, web search engines may help to increase or reduce the confidence of a correspondence, but for a full verification the approach is too weak and much more impairing than useful. The absence of linguistic evidence for a specific relation between concepts is no sufficient argument to clearly mark the relation as false.

[81] `http://www.faroo.com/hp/api/api.html`
[82] `https://www.google.de/`
[83] `https://www.bing.com/`
[84] `https://www.yandex.com/`
[85] `https://startpage.com/`

10.4.4 Summary

After different tests and assumptions were carried out, relation verification using web search engines proved to be inconvenient, both for the semantic repository and a given mapping. The reasons are:

- Search engines are too restrictive and allow only a few queries per day. Licenses can be bought for different fees, but are rather expensive and still restrict the number of queries per day.
- Search engines require too much time for the given problems.
- False relations can be confirmed to be correct, since the context of relation sentence fragments cannot be fully regarded (false negatives).
- Many correct relations are too specific (or possibly too obvious) to be found in any indexed resource. The approach would remove a considerable amount of relations that are actually correct (false positives).

At present, there seems to be no other feasible technique to evaluate the large number of relations in SemRep and erroneous relations cannot be filtered so far. The best solution to this issue is to integrate further high-quality resources into the repository to increase its density and to find shorter paths between concepts. Further knowledge from domain-specific ontologies and taxonomies could be helpful in this case.

10.5 Conclusions

In this chapter, four important issues referring to the repository quality were discussed: homonyms, entities, universal concepts and false relations. SemRep uses different techniques to alleviate the impairments caused by entities and universal concepts, while there are no conventional techniques to address the problem of homonyms and false relations. Among these two issues, only false relations seem to be a serious problem for the repository quality, while homonyms have seemingly only little influence. Table 10.1 provides a juxtaposition about the different issues presented in this chapter.

Fig. 10.3 shows the different techniques used for quality improvement (red boxes). The *category filter* is already used in the relation extraction step, so when the Wikipedia resource is built (see Section 8.3.3). The resource *Wikipedia Filtered* was generated by comparing extracted concepts with a different lexicographic resource (Wiktionary) and contains less entities. Both filter techniques had to be executed only once. *Morphological Analysis* and *Universal Concept Filtering* are applied at run time, so when data is imported to SemRep. These filters can be turned off in SemRep so that it is also possible to load the full data sets (e.g., including the universal concepts).

As a repository that contains data from most versatile resources, including automatically generated data, it seems natural that such a system is beneath perfection and that there

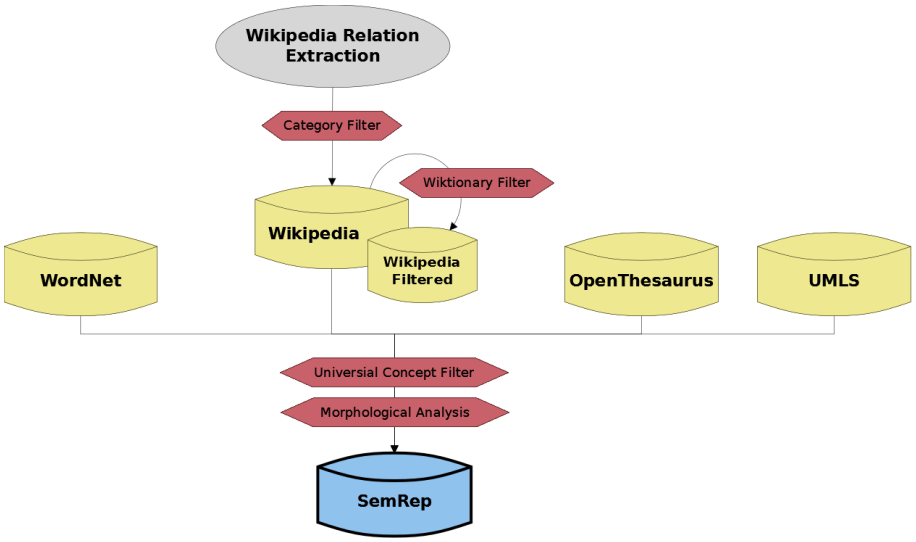

Figure 10.3: The different techniques for quality improvement (red components).

are issues that cannot be resolved so far. As we will show in the evaluation, SemRep is still a powerful tool that can considerably support STROMA in the process of mapping enrichment. Besides, its independence and general applicability for usage in further domains or research fields make it ultimately a very useful system.

Issue	**Description**	**Effects**	**Applied solutions**	**Degree of negative impact**
Homonyms	*Words with different senses are represented as one concept in SemRep*	Concluding false relation types		Low – medium
Universal concepts	*Very general, meaningless concepts are related to more specific concepts*	Concluding imprecise or false relation types	Manual filter list	Medium
Entity concepts	*Unnecessary entity concepts are contained in SemRep*	Increases setup and execution time of SemRep, concluding false relation types	Wikipedia category filter, concept comparison with lexicographic resource, morph. analysis	Low
False relations	*Erroneous relations occur between concepts*	Concluding false relation types		Medium

Table 10.1: Juxtaposition of the different quality issues.

11 Evaluation

In this chapter, we will evaluate the semantic repository SemRep. We will start with the evaluation of the automatically extracted Wikipedia relations, as they constitute a key role in this thesis and in the SemRep system (Section 11.1). In particular, we will evaluate the correct detection of the semantic patterns within the definition sentences, the quality of the term extraction as well as the quality of the final relation creation. Additionally, we will discuss important insights and lessons learned from the relation extraction approach. In Section 11.2, we will evaluate SemRep in general and show its influence on the mapping quality based on the gold standards that have been already used for the STROMA evaluation (see Chapter 7). Finally, we will provide some statistics about the different filter techniques presented in the previous chapter, and about the overall SemRep system in Section 11.3.

11.1 Wikipedia Relation Extraction

11.1.1 Overview

To evaluate the quality of the relation extraction from Wikipedia definition sentences, four data sets were manually created that consist of a complete set of Wikipedia articles of a specific category or of an outline page (a Wikipedia page itemizing a list of related articles), with the exception of "List of"-articles that never contain any semantic relations. Articles from sub-categories were not regarded in order to keep the data set sizes moderate so that they could be created within a feasible amount of time. For each article in the data set, the semantic patterns and the relevant subject terms, object terms and field references were annotated. Having these information, it was possible to calculate for each

processed article how many of the concepts and relations in the articles were discovered (recall) and how many extracted concepts and relations were correct (precision).

Wikipedia categories are very heterogeneous and partly inconsistent, which makes the data sets quite interesting for evaluations. For examples, the Wikipedia category *furniture* currently consists of 201 articles and 24 subcategories like *beds*, *chairs*, *couches*, *desks* and *tables*.[86] However, some articles that actually belong to one of these subcategories also occur in the furniture category, like the articles *Desk* or the articles *Deckchair* and *Windsor chair*. Therefore, this category contains both very general concepts and more specific concepts.

Given a set of Wikipedia articles W, not all Wikipedia articles contain any semantic relation pattern, i.e., they do not follow the classic way of definition formulation. Such articles are not parsable and the extraction approach is unable to extract any semantic relation. I denote the set of parsable articles by W_P and it holds $W_P \subseteq W$. Examples for non-parsable articles are:

- *Anaerobic infections are caused by anaerobic bacteria.*[87]
- *Hutchinson's triad is named after Sir Jonathan Hutchinson (1828 - 1913).*[88]
- *A pathogen in the oldest and broadest sense is anything that can produce disease.*[89]
- *A diving chamber has two main functions: as a simpler form of submersible vessel to take divers underwater and to provide a temporary base and retrieval system [...]*[90]

Table 11.1 provides an overview of the 4 data sets that were developed in October 2013, together with the domain, the number of articles and the number of parsable articles. The data sets *Furniture*[91] and *Vehicles*[92] contain many general domain concepts. The data set *Infectious Diseases*[93] is rather specific and refers to the biomedical domain. The data set *Optimization Algorithms*[94] is even more specific and refers to the domain of mathematics and theoretic computer science.

11.1.2 Article Parsing and Pattern Detection

In this first evaluation, the quality of the article parsing and pattern detection is analyzed. Only articles from W_p are regarded, as other articles can neither be parsed, nor do they contain any semantic pattern.

[86] `http://en.wikipedia.org/wiki/Category:Furniture` (April 2015)
[87] `http://en.wikipedia.org/wiki/Anaerobic_infection` (October 2013)
[88] `http://en.wikipedia.org/wiki/Hutchinson_triad` (October 2013)
[89] `http://en.wikipedia.org/wiki/Pathogen` (October 2013)
[90] `http://en.wikipedia.org/wiki/Diving_chamber` (October 2013)
[91] `http://en.wikipedia.org/wiki/Category:Furniture` (October 2013)
[92] `http://en.wikipedia.org/wiki/Outline_of_vehicles` (October 2013)
[93] `http://en.wikipedia.org/wiki/Category:Infectious_diseases` (October 2013)
[94] `http://en.wikipedia.org/wiki/Category:Optimization_algorithms_and_methods` (October 2013)

Name	**Domain**	W	W_p
Furniture (F)	General	186	169
Infectious Diseases (D)	Medicine	107	91
Optimization Algorithms (O)	Mathematics	122	113
Vehicles (V)	General	94	91

Table 11.1: Data sets with their number of articles W and number of parsable articles W_p.

Let ω be the number of articles that could be successfully parsed, i.e., where at least one semantic pattern, one subject term and one object term could be extracted. Let ω^T be the number of articles where the correct semantic relation pattern was detected. In this experiment, the parsing recall r_{pars} specifies how many of all parsable articles were successfully parsed. The recall for finding the semantic relation patterns r_{rel} specifies how many of all semantic patterns in the articles were detected. Finally, the precision for finding semantic relation patterns p_{rel} specifies how many of the extracted semantic patterns were correct. The three parameters are calculated as follows:

$$r_{pars} = \frac{\omega}{W_p} \tag{11.1}$$

$$r_{rel} = \frac{\omega^T}{W_p} \tag{11.2}$$

$$p_{rel} = \frac{\omega^T}{\omega} \tag{11.3}$$

The results are displayed in Table 11.2. The fifth column r_{pars} shows that between 74 and 96 % of all parsable articles were successfully parsed by the Wikipedia relation extraction approach; between 4 and 26 % of all articles failed, i.e., the semantic pattern was not discovered or the finite state machines could not reach the final state. There seems to be a tendency to better results for more general articles than to more specific articles. For example, articles from the Furniture and Vehicles data sets could be much better processed than the very specific articles from the Optimization Algorithms data set.

The precision of semantic relation pattern extraction (p_{rel}) was 96 % in the furniture experiment and 100 % in all other experiments. Therefore it holds $r_{pars} = r_{rel}$ for the experiments D, O, V, while in F the recall for relation extraction is slightly below r_{pars} as 4 % of the patterns were incorrect. The semantic relation pattern extraction has thus an excellent precision, while the recall is somewhat below, depending on the data set.

11.1.3 Term Extraction

In the second experiment, the quality of the term extraction is analyzed. Each Wikipedia article that was successfully parsed leads to at least two terms (a subject term and object term). Let T^P be the number of terms that occur in the Wikipedia pages parsed

	W_p	ω	ω^T	r_{pars}	r_{rel}	p_{rel}
F	169	148	142	0.88	0.84	0.96
D	91	80	80	0.88	0.88	1
O	113	84	84	0.74	0.74	1
V	91	87	87	0.96	0.96	1

Table 11.2: Evaluation of parsing and pattern detection.

	r				p			
	S.	O.	2L O.	F.	S.	O.	2L O.	F.
F	0.93	0.90	0.66	0.8	0.95	0.87	0.58	0.73
D	0.85	0.87	0.70	1	0.96	0.80	0.57	0.67
O	0.84	0.91	0.81	1	0.88	0.88	0.36	0.92
V	0.83	0.94	0.86	–	0.96	0.94	0.49	–

Table 11.3: Recall and precision for term extraction.

by the extraction approach. Let T_C be the correctly extracted terms and T_F be the falsely extracted terms. To evaluate the quality of the term extraction, recall and precision are used again:

$$r = \frac{T_C}{T^P} \tag{11.4}$$

$$p = \frac{T_C}{T_C + T_F} \tag{11.5}$$

The evaluation regards recall and precision for all kinds of extracted terms, i.e., for subjects (S), first-level objects (O_1), second-level objects (O_2) and field references (F). The results are shown in Table 11.3.

The recall for subject terms (83 to 93 %) and first-level object terms (87 to 94 %) is similarly good, with the first-level objects achieving slightly better results. Second-level objects have a somewhat lower recall (66 to 86 %), since these concepts participate in `has-a` or `part-of` relations and are more difficult to discover.

Likewise, precision for subject terms (88 to 96 %) and first-level object terms (80 to 94 %) is similarly good, with the subject terms achieving slightly better results. The precision for second-level objects is considerably lower (36 to 58 %), which indicates that a certain amount of extracted terms is erroneously combined into `has-a` or `part-of` relations.

Field references occur only scarcely, and only in the data sets F, D, O. The recall is very good (80 to 100 %), while the precision is somewhat lower (67 to 92 %), which means that some terms are falsely regarded to be fields.

Table 11.4 provides the absolute number of extracted concepts and resulting relations. The number of extracted subject terms is invariably the highest, because of the many synonym terms that can be found in the Wikipedia definition sentences. By contrast, def-

	W_P	Subj.	Obj.	2L O.	Fields	Rel.
F	169	200	142	43	4	373
D	91	111	58	26	4	206
O	113	84	66	6	23	137
V	91	138	78	17	0	280

Table 11.4: Number of extracted concepts and relations in each experiment.

	Rel. in W_p	Correct rel.	False rel.	r	p	f
F	497	373	87	0.75	0.81	0.78
D	323	206	67	0.64	0.76	0.69
O	182	137	49	0.76	0.74	0.75
V	413	280	66	0.68	0.81	0.74
$\sum$	1,415	996	269	0.71	0.78	0.74

Table 11.5: Number of relations per data set, correctly extracted relations, falsely extracted relations as well as recall, precision and F-measure.

inition sentences usually contain only one hypernym, so that the number of first-level object terms is lower. The number of second-level object terms is again much lower, since a second pattern occurs only occasionally in definition sentences, and because of the generally lower recall and precision for second-level object term extraction. Field references occur only scarcely, although there are some notable differences between the four data sets. They seem to occur more often in scientific or more specific domains than in general domains. Thus, there were 23 field references in the data set about optimization algorithms, as articles often contain phrases like *"in mathematics"*, *"in the field of graph theory"*, etc. There were no field references for vehicles, though, because there is mostly no reason to mention the field (e.g., transportation, traffic) to which a vehicle article belongs.

11.1.4 Relation Extraction

In this section, the quality of the relation extraction is analyzed. Semantic relations are built directly from the extracted patterns and terms, i.e., they completely depend on the quality of pattern and term extraction. As the relations are the outcome of the Wikipedia relation extraction approach, this experiment reveals the actual quality of the overall approach.

The results are shown in Table 11.5. The second, third and fourth column show the number of relations contained in the parsable Wikipedia articles, the number of correctly extracted relations and the number of falsely extracted relations. Having these values, which had been manually determined by analyzing each processed article, recall, precision and F-measure could be calculated for each data set.

	share	r	p
equal	24.1 %	0.73	0.87
is-a	55.4 %	0.87	0.88
has-a / part-of	19.4 %	0.58	0.63
field ref.	1.1 %	0.36	0.57

Table 11.6: Distribution, recall and precision for each individual relation type in the Furniture experiment.

	S.	O.	2L O.	F.	$\sum$ Con.	Rel.
F	9	21	30	1	61	87
D	5	15	19	2	41	67
O	12	9	12	2	35	49
V	6	5	19	0	30	66
$\sum$	32	50	80	5	167	269

Table 11.7: Number of falsely extracted concepts and relations in each experiment.

The recall ranges between 64 and 76 % (with an average of 71 %), while the precision is somewhat better (74 to 81 %, with an average of 78 %). The F-measure ranges between 69 and 78 %, with an average of 74 %.

The results show that about 21 % of all extracted relations are incorrect. Precision and recall change considerably w.r.t. the relation type, which is a consequence of the different recall and precision values achieved in the term extraction phase. Table 11.6 shows the recall and precision of each relation type, i.e., for `equal` relations, `is-a` relations, `has-a` resp. `part-of` relations and field reference relations. It can be observed that `is-a` relations have the highest share on all extracted relations and obtained the best recall and precision. Thus, the relation extraction approach scores best at `is-a` relations, which constitute the core element in definition sentences. Relations of type `equal` have still a good precision, but a somewhat lower recall. As expected, `has-a` and `part-of` relations, as well as field references, have the lowest recall and precision, which is a result of the lower recall and precision achieved in the term extraction.

Finally, the number of falsely extracted terms and relations is shown in Table 11.7. Column 2 – 5 show how many subject terms, first-level and second-level object terms as well as field references were falsely extracted in each experiment. The overall number of falsely extracted concepts is shown in column 6. The number of falsely extracted relations, which is mostly a result of the falsely extracted concepts, is shown in column 7. The number of falsely extracted relations (269) is much higher than the number of falsely extracted terms (167). This reveals that a false term can impair more than one relation, as it can take part in several relations at the same time.

11.1.5 Observations

Though the Wikipedia relation extraction approach is highly effective and retrieved a large number of relevant semantic concept relations, this approach is unable to find all relations encoded in the definition sentences, nor can erroneous extractions be prevented. In this section, we will present some observations and insights gained from the previous experiments that illustrate why a perfect recall or precision cannot be achieved.

If semantic relations cannot be extracted from a sentence, it is mostly because of a too complex sentence structure or an uncommon definition style, which cannot be handled by the FSMs. Examples include:

1. *Cryptic Infections: an infection caused by an as yet unidentified pathogen...*[95]
2. *A hospital-acquired infection, also known as a HAI or in medical literature as a nosocomial infection, is an infection...*[96]
3. *Lower respiratory tract infection, while often used as a synonym for pneumonia, can also be applied to other types of infection including...*[97]

The first example does not contain any semantic pattern, but uses the dictionary form where the object follows directly after the subject, like "car: a motor vehicle to transport passengers". The second example is too complex, i.e., it consists of an apposition that is too long to be successfully handled. Although the approach finds the `is-a` pattern, it cannot successfully parse the first fragment and thus no relations are extracted. The third examples has feats of both previous examples: It has no obvious semantic pattern and is quite complex. The phrase *"can also be applied to other types of"* suggests an `is-a` relation, but is no typical `is-a` pattern that could be discovered by the approach.

Sometimes, articles cannot be extracted because of erroneous POS-tagging. For example, in the sentence *A dog sled is a sled used for...* both occurrences of *sled* are denoted as verb. Although the word *sled* can be a verb, it is a noun in this instance. The relation extraction approach is unable to parse this sentence, as it does not expect a verb occurring before an `is-a` pattern. According to the Apache Open NLP documentation, the part-of-speech tagger has an accuracy of 96.6 %, which means that a false assignment can occasionally occur.[98]

The precision of term extraction is impaired by the following reasons:

1. The correct end of a compound word cannot always be correctly determined. In the example *"A minibus is a passenger carrying motor vehicle"*, the word *carrying* is denoted as a verb. As a result, the relation <*minibus*, `is-a`, *passenger*> is extracted,

[95] `http://en.wikipedia.org/wiki/Cryptic_infection` (October 2013)
[96] `http://en.wikipedia.org/wiki/Hospital-acquired_infection` (October 2013)
[97] `http://en.wikipedia.org/wiki/Lower_respiratory_tract_infection` (October 2013)
[98] `https://opennlp.apache.org/documentation/1.5.3/manual/opennlp.html#tools.postagger.eval`

because a verb should always mark the end of a concept term in a sentence. However, in this case the correct object term is *passenger carrying motor vehicle* or simply *motor vehicle*.

2. Similar to the above case, the beginning of compound words cannot always be correctly determined. In the example *"A draisine is a light auxiliary rail vehicle"*, the object *light auxiliary rail vehicle* would be extracted, although the correct object should be rather *rail vehicle*.

3. Itemizations can sometimes contain nouns that are erroneously regarded as concepts. For example, in *"Jet pack, rocket belt, rocket pack and similar names are used for various types of devices"* the concept *jet pack*, *rocket belt*, *rocket pack* and *similar names* are extracted. Also in phrases like *is a noun for* or *is the act of*, such unreasonable concepts could be extracted.

Some problems presented in this section can be resolved by a more complex FSM or by more comprehensive word lists defining illegal or inappropriate concepts, while other problems like the exact compound extraction or erroneous POS-tagging seem much more difficult to handle.

11.2 SemRep

SemRep was developed to foster STROMA in the mapping enrichment process and is a very effective strategy. To evaluate the quality of SemRep and its beneficial influence on mapping enrichment, we will use the same gold standards as in the STROMA evaluation and will enrich these mappings first without applying STROMA, then with using WordNet as the only background knowledge resource and finally with using the full SemRep. To alleviate this evaluation, we will work on the perfect mappings, because the quality of SemRep is the main focus of this evaluation and not the overall quality of the enrichment process. For this reason, we will handle the type *undecided* as false, i.e., we will not use the *Undecided-as-Equal* configuration. Moreover, we will generally disable the heuristic strategies Multiple Linkage and Word Frequency, because they impair the analysis of the background knowledge effectiveness. The results achieved by using all strategies were already illustrated in the previous STROMA evaluation (Chapter 7).

SemRep has been already evaluated in [9]. Due to the adjustment of SemRep and the different filters for quality improvement, those values are no longer up-to-date. Instead, we will present the evaluation results of the current version of SemRep (as of April 2015).

11.2.1 SemRep Configuration

Before we come to the actual evaluation of SemRep, we will show the behavior of SemRep for different configurations in order to determine the best configuration of the repository. The most crucial parameters of SemRep are the maximum path length (l) and the termination mode (*first path* or *all paths*), while the threshold parameters for the different

	First Path			All Paths		
	$p = 2$	$p = 3$	$p = 4$	$p = 2$	$p = 3$	$p = 4$
W	61.5	61.5	**61.8**	61.5	61.5	**61.8**
D	73.5	73.8	73.8	73.5	**74.0**	73.8
T	68.4	66.4	65.9	68.4	69.2	**72.5**
F	**63.2**	62.5	61.0	**63.2**	62.5	**63.2**
G	47.9	47.9	47.9	**49.1**	47.9	47.9
C	**69.0**	64.8	59.9	**69.0**	64.8	59.9
L	**63.4**	57.7	57.7	**63.4**	59.1	56.3
Avg.	63.8	62.1	61.1	**64.0**	62.7	62.2

Table 11.8: F-measures achieved in the experiment for the two modes *first path* and *all paths* and for maximum path lengths 2, 3 and 4.

strategies and types have only little impact. We will show the F-measure for each experiment for the maximum path lengths of 2, 3 and 4 and for the two modes *first path* and *all paths*. We will only enable the Background Knowledge and Itemization Strategy in STROMA to illustrate the direct influence of the SemRep configuration on the mappings. Table 11.8 shows the results for each experiment and each configuration. For better legibility, the best result in each experiment is marked bold.

In 4 of the 7 experiments (F, G, C, L), the best results were achieved with maximum path lengths of 2. In 3 of those 4 experiments (F, C, L) *first path* and *all paths* led to the same result, while only in experiment G *all paths* led to better results. In experiments W, D, T the best results were either achieved with maximum path lengths of 3 (D) or 4 (W, T). *All paths* proved to achieve somewhat better results again.

Except for experiment T, an increasing path length could improve the results only very slightly. Changing the path length from 2 to 3 or 4 improved the F-measure of experiment W by 0.3 % and of experiment D by 0.5 %. By contrast, the difference between shorter and longer path lengths is considerably larger in the other experiments. For example, changing the path length from 2 to 4 in experiment C decreased the result by almost 10 %.

The obtained results allow two significant conclusions: first, that *all paths* invariably leads to the same or better results, and second, that $l = 2$ seems to be the best maximal path length. There are gold standards where longer path lengths lead to better results, but the gain in those experiments is only small compared to the loss of F-measure occurring in the other gold standards. By default, SemRep is thus run in *all paths* mode with maximum path lengths of 2 (default configuration). This configuration led to the best average F-measure in the evaluation (64.0 %). Another insight is that in some experiments different configurations only lead to minor changes in the mapping quality (e.g., W, D, G) while in other experiments the results changed quite noticeably (e.g., C, L).

	First Path			All Paths		
	$p = 2$	$p = 3$	$p = 4$	$p = 2$	$p = 3$	$p = 4$
W	11.2	11.5	11.8	**10.9**	12.3	12.4
D	**7.1**	15.4	31.5	**7.1**	19.6	113.1
T	12.7	57.9	112.3	**12.4**	354.6	1,749
F	**42.3**	96.6	271.8	46.1	216.0	2,380
G	**47.7**	107.4	746.1	48.3	259.3	6,353
C	**37.3**	89.0	159.6	37.4	158.1	990.3
L	**43.7**	288.2	1,107	45.0	597.4	6,637
Avg.	**28.9**	95.1	348.6	29.6	231.0	2,605

Table 11.9: Average execution times (in ms) for one correspondence in each experiment and configuration.

Time Complexity

The quality of the repository should not be entirely made up by the F-measures achieved in the seven experiments. Execution time is another important issue that must be regarded to ascertain the overall usefulness and effectiveness of SemRep. Table 11.9 shows the average execution times for a correspondence in each experiment in milliseconds. The best results are marked bold again, i.e., the configuration in which the processing time was minimal. The time was measured internally using the system time in milliseconds, which means that they are only approximate values. Analyzing the time behavior, there is a clear answer about the best configuration w.r.t. execution times. Maximum path lengths of 2 always lead to the best execution times, as only simple paths have to be calculated in this configuration.

Theoretically, the configuration $p = 2$ and *first path* is the fastest configuration. Sometimes, as in experiments W and T *all paths* is faster. However, this is just an expected variation of the processing time of the server and the general inaccuracy of execution time measurement, as *all paths* requires the computation of at least as many operations as *first path*. However, there is no noticeable difference between *all paths*, with an average of 28.9 ms per correspondence and *first path*, with an average of 29.5 ms per correspondence. Both configurations allow a processing time of about 34 correspondences per second, which means an overall execution time of about 30 seconds to carry out a mapping of 1,000 correspondences. This is quite an acceptable result. Compared to the previously carried out STROMA experiments, the average execution times are somewhat higher in this evaluation (they have been about 22 ms per correspondence, see Table 7.13). Thus, using SemRep in the entire STROMA architecture is a little faster than using SemRep as the only selected strategy.

As soon as complex paths are calculated ($p = 3$), the execution time increases dramatically. While execution times remain moderate in *first path* mode (95 ms per correspondence), they are already quite high in *all paths* mode (231 ms per correspondence). For $p = 4$, execution times increase further on. For first path, there is only a moderate increase to 349 ms per correspondence. This is quite natural, because most paths found between

two concepts are of length 3 or less and only in some specific cases the calculation of a path longer than 3 becomes necessary. In *all paths* configuration, SemRep has to calculate all possible paths of length 4, though, which leads to the longest execution times (2,605 ms per correspondence). This is an unacceptable amount of time for standard mappings, as processing 1,000 correspondences would require about 45 minutes.

Again, there are tremendous differences between the separate experiments. Experiments W and D only require very specific parts of SemRep (the German part or the rather isolated biomedical part) and thus scale well for longer path lengths. The other experiments usually access the whole repository and require different execution times. For path lengths of $p = 2$ and *all paths* mode the execution times range between 12.4 and 48.3 ms, which is a factor of about 4. For $p = 4$, the range is between 990 and 6,637 ms, which is even a factor of 6.7. Apparently, the kind of concepts appearing in the gold standards and their complexity is a crucial factor for the resulting processing time. Generally, itemizations require more execution time than simple concepts, because they contain different words that have to be looked up in SemRep. Less general concepts require less execution time than more general concepts, because they have a lower node degree, which means that less paths need to be calculated.

A remarkable outcome of the evaluation is that the default configuration of SemRep ($p = 2$, all paths) does not only lead to the best mapping results, it also leads to the fastest execution time (besides the *first path* mode). Even larger mappings can thus be processed within seconds or at most minutes. Possibly, SemRep might be even used to support matching tasks, which usually require more calculations. If, for instance, two 250-concepts ontologies were matched, the resulting 31,250 comparisons could be carried out in approx. 15 minutes. Still, SemRep is solely designed for the mapping enrichment process and may require much time to support the (complete) matching of medium-sized or larger ontologies. In this case, the initial reduction of the search space by means of blocking would be a very convenient step.

11.2.2 Quality Improvement

Eventually, we want to show how SemRep supports STROMA in mapping enrichment and how the mapping quality is improved. We present the results STROMA achieves if no background knowledge is used (first configuration), if only WordNet is used as background knowledge (second configuration) and if the full repository is used (third configuration). The results are illustrated in Table 11.10.

It can be observed that the mapping quality of STROMA without using any background knowledge ranges widely between a poor F-measure of 18.9 % (T) and rather good F-measures above 60 % (F, W, D). The average F-measure is exactly 50 %. As the perfect mappings were used as input (i.e., it holds $r = p = f$), this means that only every second correspondence is correctly typed.

If WordNet is used, the F-measure improves considerably to a range between 52.6 and 75.7 %, with an average of 66.4 %. Thus, WordNet can already increase the mapping quality to a large degree. There was no improvement in experiment W, as it is a German-

Gold Standard	**Without Background Knowledge**	**With WordNet**	**With Full SemRep**
Web Directories (W)	63.2	63.2	63.8
Diseases (D)	65.4	71.2	75.8
Text Mining Tax. (T)	18.9	69.2	77.2
Furniture (F)	60.2	69.1	68.3
Groceries (G)	29.5	52.6	52.6
Clothing (C)	56.0	75.7	83.1
Literature (L)	56.6	63.8	66.2
Average	50.0	66.4	69.6

Table 11.10: F-measures for using STROMA without background knowledge, with WordNet as only background knowledge resource and with the fully initialized SemRep.

language scenario, and only slight improvements in the experiments D and L (5.8 to 7.2 %). There were moderate improvements in the experiments F and C (8.9 to 19.7 %) and even remarkable improvements in experiments G and T (23.1 to 50.3 %). The average improvement is 16.4 %. Note that in the context of the evaluation, SemRep was used with WordNet being the only resource that was loaded. Thus, the mappings could also benefit from some of the SemRep specifics, such as the different forms of preprocessing and path scoring. If the WordNet API had been used, results may have been somewhat lower, as these techniques are not provided by the WordNet API.

Finally, the last column shows the results for using STROMA together with the entire SemRep, i.e., with all resources loaded. Again, there is an increase of F-measure in all experiments, except for experiment F, where there is a decrease of 0.8 % and G, where the result remained the same. The other experiments led to increased results between 0.6 % in W and 8 % in T. The average quality was increased by 3.2 % compared to the configuration in which only WordNet has been used. This shows that the additional knowledge combined in SemRep is quite effective compared to a WordNet-only approach, although the already good quality achieved with WordNet (66.4 %) seemingly does not allow much further progress by combining additional resources in SemRep.

In comparison to using STROMA alone (first configuration), the mapping quality could be boosted by 19.6 %, which also demonstrates the overall importance of background knowledge in the field of semantic mapping enrichment. As we have shown in the previous STROMA evaluation, some of this apparent quality improvement gained by background knowledge is blurred by some other strategies, e.g., the heuristic strategies, so that the quality difference is lower if STROMA is used with all strategies. In some particular scenarios, heuristic strategies could even outperform background knowledge. Nonetheless, SemRep is an efficient, universally applicable background knowledge repository and its overall benefits should not be doubted because of the mere existence of other implemented strategies that may achieve similar good results.

11.3 Quality Improvement

11.3.1 Filtering

Without any filtering, the Wikipedia approach extracts 4,386,119 concepts and 12,519,916 relations (see Table 11.11). The first step of quality improvement is the category filtering, which is performed during the actual extraction phase. Applying this filter technique, the number of concepts is reduced by 66 % to about 1.48 million concepts and the number of relations is reduced by about 63 % to about 4.69 million. Morphological analysis reduces the number of concepts and relations further on so that only 2.84 million relations resp. 1.05 million concepts remain, which is less than a forth of the original data set.

Data Set	Relations	Concepts	Remaining Relations (%)	Remaining Concepts (%)
Original	12,519,916	4,386,119	100	100
Category Filter	4,693,514	1,477,462	37.5	33.7
Morph. Analysis	2,843,428	1,051,170	22.8	24.0
Wiktionary Filter	1,489,577	548,685	11.9	12.5

Table 11.11: Number of remaining concepts and relations in the Wikipedia data sets after different quality improvement techniques.

After analyzing 800 random articles that were blocked by the category filtering and morphological analysis, only 1 article was considered to be rather a concept than an entity. This means a precision of 99.875 % and shows that the filter technique is very reliable and that concepts are not inadvertently removed from the repository. However, analyzing the remaining 1.05 million concepts revealed that there were still about 49 % entity concepts in the repository that have not been blocked so far. These often included villages, administrative districts, events, companies and product names.

The Wiktionary Filter considerably reduces the number of concepts and relations once again. After the category filter and morphological analysis, it reduces the number of relations by 47.6 % to about 1.49 million and the number of concepts by 47.8 % to now 548,685. The precision of this technique is about 97 %, meaning that about 3 % of concepts are erroneously blocked. Besides this, several terms from the biomedical domain are blocked, which are partly concepts and partly entities. Analyzing the remaining concepts, about 32 % of entities remained while about 68 % were concepts (about 394,000 in absolute numbers). This value is similar to the previous approximation of Wikipedia having about 300,000 concept articles, plus some biomedical concepts, plus some synonymous terms, and minus some concepts that could not be extracted by the extraction process (see Section 8.3.3). Still, it seems impossible to provide a definite answer about the distribution of entities and concepts in the remaining data set, because many terms cannot be unequivocally assigned to concepts or entities. Nonetheless, the three filter techniques are highly effective, because they limit the number of original concepts and relations by almost 90 %, with only few concepts being erroneously removed.

	WN Relations	WN Concepts	Wikipedia Relations	Wikipedia Concepts
Original	2,065,967	120,435	1,489,577	548,685
UCF	1,881,346	119,895	1,488,784	548,610

Table 11.12: Number of remaining concepts and relations in the Wikipedia and WordNet data set after applying Universal Concept Filtering.

Finally, we will discuss the Universal Concept Filtering (UCF) technique, which eliminates all relations in which a universal concept participates. This filter has both effects on the Wikipedia data set and the WordNet data set, while the previous techniques are rather used for the Wikipedia set alone. Therefore, we will show the effect of this filter both for Wikipedia and WordNet (while it has no notable effect on the biomedical UMLS data set or the German OpenThesaurus data set). The results are shown in Table 11.12. Evidently, UCF has an important impact on the WordNet data set, where 184,621 irrelevant relations are filtered (8.9 % of all relations). By contrast, the number of filtered Wikipedia relations is only 793 (about 0.05 % of all relations). This large gap does not come as a surprise, because Wikipedia definition sentences usually do not contain universal concepts. For example, a definition for *table* would scarcely be of the kind *A table is an object...* or *A table is a thing* while WordNet provides many relations between common concepts and such universal concepts.

The number of reduced concepts is small in both resources, because the number of universal concepts is very low (about 80 terms were defined). As the number of reduced concepts in WordNet is 540 and thus exceeds the number of defined universal concepts, a few additional concepts have been removed, though. These had been concepts X that exclusively participated in relations (X, Y) where Y is a universal concept.

11.3.2 Overall Statistics

Eventually, we present the number of relations and concepts imported from each resource. All filters were applied, i.e., Category Filter and Wiktionary Filter for Wikipedia and the Morphological Analysis and Universal Concept Filter for all resources. The results are shown in Table 11.13. As of today, SemRep contains 7,040,329 relations and 3,540,077 concepts. There is an expected overlap of concepts between the different resources. If the concepts occurring in each resource would be added up, the number of concepts would be (theoretically) 3,939,834.

The average degree of each node (concept) is 1.99, while 65.1 % of all nodes have a degree of 1 (among them many concepts from the Wikipedia Redirects). The node having the highest degree is *person*, which is a concept existing both in German and English and having numerous links to specific persons (like an optimist, a prankster, etc.) and especially occupations (baker, teacher, physician, etc.). The node degree of *person* is 20,796 and evidently the power laws hold in SemRep just as in many other real-world graphs.

Resource	Relations	Concepts	Rel./Concept
WordNet	1,881,346	119,895	15.7
Wikipedia	1,488,784	548,610	2.71
Wikipedia Redirects	1,472,117	2,117,001	0.70
Wikipedia Field Ref.	72,500	66,965	1.08
OpenThesaurus	614,559	58,473	10.5
UMLS	1,265,703	938,527	1.35
ConceptNet	245,320	90,364	2.71

Table 11.13: Number of relations, concepts and relations per concept for each resource.

The average node degree of about 2 may appear very low. Theoretically, calculating all paths of length 4 would thus require $2^4 = 16$ operations, which was apparently not the case in all the experiments. However, everyday concepts found in the mappings have usually a high degree, even with the now considerably filtered data sets. Thus, average degrees for common words like *furniture*, *chair* or *carpet* have still around 100 or even more outgoing relations. Thus, the different implementation adjustments to make SemRep faster were (and still are) absolutely necessary.

In SemRep, 46 % of all relations are of type `equal`, 39 % are `is-a` or `inverse is-a` and 15 % are of type `has-a` or `part-of`. By contrast, the extracted WordNet data set consists of 14 % `equal`, 66 % `is-a` or `inverse is-a` and 20 % `has-a` or `part-of` relations, which means that SemRep has a larger share of `equal` and lower share of `is-a` relations. This is also a result of the import of the many Wikipedia redirect relations, which are all treated as `equal` relations.

Part IV

Conclusions and Outlook

12 Conclusions

In this doctoral thesis, we have introduced a novel approach for semantic mapping enrichment. The primary focus of this research is the relation type determination of the correspondences within a given input mapping. We could show on several examples that advanced data integration techniques like ontology evolution or ontology merging can benefit from the knowledge about the correspondence types. In fact, such enriched mappings are much more expressive than the mappings produced by state-of-the-art tools, which either do not provide a relation type or tacitly assume equality-relations, although correspondences of a different type may be contained.

Starting with a comparison of match tools, we could show that there are only few approaches being able to calculate semantic correspondences; the evaluation showed that such tools do not achieve satisfying results. Most of those tools have not even been evaluated w.r.t. semantic relation types, mostly because of the substantial lack of suitable benchmarks in this research field. Some approaches were compared to other match tools that could not determine semantic types, which naturally resulted in a complete neglect of the semantic relation type verification. It appears that most semantic match tools determine relation types as a side effect, with the focus still being on the problem of finding correct links between schemas or ontologies, and not on relation type determination. In this instance, the presented two-step approach stands out against previous work, as it is fully dedicated to the relation type determination, while the general correctness of a correspondence is rather subsidiary.

The enrichment tool STROMA is able to determine six different relation types (`equal`, `is-a`, `inverse is-a`, `has-a`, `part-of` and `related`). So far, it seems to be the only tool that distinguishes between the types `is-a` (subsumption) and `part-of` (aggregation), which are usually not differentiated in related approaches. STROMA is a very flexible tool that can technically process any imaginable mapping, as the required input is a simple list of

concept correspondences. To determine the relation types, different linguistic strategies have been developed: morphological strategies (compound), structural strategies that determine the relation type indirectly, background knowledge strategies and heuristic strategies. Thus, many sub-disciplines of the field of linguistics have been touched, e.g., word formation, word frequency, word semantics and sentence parsing. Each strategy comes with different strengths and weaknesses, so that only the combination and interrelation of all strategies leads to the very good results STROMA is able to achieve on perfect mappings.

STROMA achieves an average F-measure of approx. 80 %. Though this means that in 20 % of all cases the correct type could not be determined, this is quite a remarkable result, given the high complexity of languages. Since the determination of the concrete relation type is usually more difficult than the determination of a match between concepts, we consider STROMA to be a fully successful and effective mapping enrichment tool. An important issue is the relatively low strict F-measure in real input mappings though, which may be erroneous and incomplete. STROMA depends highly on the input mapping, resp. on the match tools used in the first step, and is hence unable to make up for imprecise or missing correspondences. We will discuss different possibilities for improvement in the following chapter.

In addition to the problem of relation type determination between concepts, side issues had to be addressed in this thesis which occasionally occur in the schema and ontology mapping domain. For instance, itemizations like *Computers, PCs and Screens* had to be handled separately and a correspondence between two identical leaf concepts like *Shoes* does not necessarily imply an actual equality-correspondence. In this instance, it became obvious that the full concept path has to be taken into account and not the concept alone. STROMA is also able to cope with such specific situations and is ultimately a mapping enrichment tool that addresses all necessary issues and circumstances for a profound, universal enrichment of schema and ontology mappings.

Background Knowledge proved to be a successful strategy and constituted a key role in this thesis. As WordNet alone did not yield satisfactory results, the combination of several resources was investigated and implemented. To obtain additional lexicographic knowledge, English Wikipedia articles have been processed, resulting in a large number of semantic concept relations. The extracted relations were combined with other lexicographic resources in the semantic repository SemRep, which now serves as an independent background knowledge repository for STROMA or any other schema or ontology mapping tool.

Many important obstacles had to be overcome while SemRep was designed, implemented and tested. First of all, there was a high need for fast query execution, which turned out to be a sophisticated requirement, as the repository consists of some million relations and the mapping enrichment is expected to be carried out within a short time. An important insight was that current database systems like MySQL or even the graph system Neo4j are too slow for high-performance path search and that the individually implemented, hash-based SemRep outperforms these systems by a wide margin. Second, the relation type calculation in complex (indirect) paths had to be investigated. The result is a complete matrix that shows the resulting relation type in an indirect path, which is

based on semantic deduction and has not yet been introduced so far. Significant observations were the prevalence of aggregation (`has-a` and `part-of`) against subsumption and the unsolvability of some specific inverse relation types occurring within a path. Further refinements of SemRep, e.g., the preprocessing of concepts, allow even correct results if a given input word is not contained in the repository, which makes SemRep a very convenient resource that goes far beyond previous approaches for background knowledge exploitation.

One drawback of the Wikipedia relation extraction approach was the extraction of irrelevant and false relations. The falsely or unnecessarily extracted relations can impair the quality of SemRep and increase the general execution time for queries. As a last big issue of this work, the quality augmentation of SemRep has been investigated. Applying different filters, more than 80 % of unnecessary concepts and relations could be removed from SemRep. These investigations also showed the limits of quality augmentation in the vast pool of combined, partly automatically created relations. Until today, some entities cannot be discovered in SemRep and only about 70 % of all words can be considered real concepts. Though there are still possibilities to further reduce the remaining entities, the required effort would exceed the additional benefits gained from it. This is a classic example of the so-called pareto principle (80-20 rule). For example, experimental results did not change after the Wiktionary filter was applied, which indicates that the repository already had a good quality before this filter was used. The distinction between concepts and entities was another important part in these studies; besides actual concepts and entities, numerous terms were discovered that share properties of both, which exacerbated the general evaluation of the repository quality.

The discovery of false relations in SemRep could not be accomplished so far and remains an unsolved issue. We could show that the search engine approach for relation verification, as also introduced in some related work, is practically impossible in the given situation, and that it is neither applicable from a time complexity point of view, nor from a quality point of view. Still, many irrelevant relations have already been removed by the entity filters, so that the number of remaining false relations can be considered to be acceptable.

13 Outlook

From a linguistic point of view, this doctoral work covers many important sub-disciplines and methodologies that are applicable for the relation type determination. Still, there remain different opportunities for adaptation and improvement. First of all, the high dependency of STROMA to the initial match tool can be alleviated if post-matching is performed. To achieve this, STROMA needs to parse the input ontologies and possibly calculate new correspondences if correspondences seem incorrect or are potentially missing. The implemented strategies can highly support such a match task, however, the processing time for mappings would consequently increase. Besides this, STROMA would lose much of its current independence, as it would need the original ontologies for mapping enrichment. Therefore, we consider it more advantageous to use a highly efficient match tool in the first phase, possibly with relaxed configurations. Additionally, mapping verification or mapping repair could be carried out during this first phase, or directly after it, in order to pass a high-quality mapping to STROMA.

Another possibility to increase the relation type determination is to analyze instance data, just as carried out by many common match tools. Given m instances in the source concept and n instances in the target concept, STROMA could compare the $m \times n$ concepts with the already implemented strategies. The relation type that holds between most instance pairs could indicate the overall relation type of the correspondence. Though this approach could improve the general results, it is time-expensive and requires actual instance data, something that is not always available and would again reduce much of STROMA's independence.

There are still many background knowledge resources that could be integrated in SemRep. Existing resources like the NCI Thesaurus or GeoNames can be integrated to cover more scientific domains. Besides, there exist many comprehensive resources which contain similar concept definitions as provided by Wikipedia. A slightly adapted version

of the Wikipedia relation extraction approach could also yield a vast number of semantic relations from resources like Wiktionary or the Free Dictionary. It might be even possible to parse sentences in publicly available corpora to obtain further semantic relations, though there is always the risk of extracting too many irrelevant or erroneous relations. For example, the import of ConceptNet relations did not lead to any better results in the experiments and showed that more knowledge does not automatically imply better results, but that quality plays a crucial part in this case.

Until today, there is no general answer to the problem of detecting false relations in SemRep. In this case, manual verification of selected (dubious) relations might be conceivable. Crowd sourcing proved to be quite successful in related approaches and might be very useful in this case as well. As mentioned before, SemRep can be also used to foster mapping re-use, and the availability of correct (manually verified) mappings can be exploited to discover some false or contradicting relations in SemRep. Additionally, the homonym issue has not been solved so far, and appears to be similarly complex as the removal of false relations.

The discovery of `part-of` and `has-a` relations still poses an important issue in the context of this research. Unlike `equal`, `is-a` and `inverse is-a` relations, they are not the primary focus of most background knowledge sources and there are no fully reliable linguistic techniques to discover such relations. As illustrated in this work, `part-of` and `has-a` relations are often more fuzzy and less universally valid, which further exacerbates their discovery. Possibly, compounds need to be more thoroughly analyzed in order to figure out whether they constitute a `has-a` or `part-of` relation. The acquisition of new `part-of` relations from natural texts, similar to the relation extraction from Wikipedia, may be another opportunity to overcome this current shortcoming.

Eventually, STROMA is currently rather designed for English-language mappings. Processing German-language mappings is still quite a challenge, partly because of the high complexity of German compared to English, and partly because of the absence of comprehensive, freely available lexicographic resources. Adapting STROMA to cope with other languages than English may be an interesting undertaking. The parsing of Wikipedia articles can also be performed in other languages and only requires an adaptation of the current approach towards the specifics of the respective language. The compound approach works for Germanic languages, but can also be adjusted to work in Romance languages like French, Italian or Spanish, where compounds are differently formed. Such an enhancement would make STROMA more useful for scenarios of different languages.

There are still more possibilities for improvement and adaptation, as each implemented strategy, and SemRep on top of it, is complex enough for further research. The confidence calculation of relation types or paths in SemRep could be further refined, as well as the currently applied strategy weights. STROMA or SemRep could be combined with other match tools, possibly allowing even better results. However, in the final analysis it must be acknowledged that the nature of language, with its high complexity and intangibility, suggests that no match or mapping tool could ever produce complete and entirely correct mappings, and it seems unlikely to come even close to such a desirable result. Keeping this in mind, the given quality of STROMA reaching average F-measures of some 80 % does not leave much scope for quality improvement after all.

Part V

Appendix

A

STROMA Default Configuration

Strategy Weights	
Compound Strategy	1.0
Itemization	1.0
Background Knowledge Strategy	0.9
Multiple Linkage Strategy	0.8
Structure Strategy	0.7
Word Frequency Strategy	0.6
Parameters for Selected Strategies	
Minimal modifier length (Compound Strategy)	3
Minimal head length (Compound Strategy)	3
Dictionary Look-up (Compound Strategy)	no
Minimal word frequency quotient	8.0
Minimal word frequency quotient (compounds)	1.0
Minimal number of links (Multiple Linkage Strategy)	3
Type Verification	
Equal verification	false
Is-a verification	true
Part-of verification	true
Undecided-as-equal handling	true
Matching and Selection	
Strong acceptance threshold (θ)	0.4
Weak acceptance threshold (θ_0)	0.2
Minimal type confidence for acceptance	1.0
Minimal prefix overlap (verification)	5 letters

Table A.1: Default settings used by STROMA.

B

SemRep Default Configuration

Default Resources to be Loaded	
Wikipedia Original Data Set	no
Wikipedia Filtered Data Set	yes
Wikipedia Redirects	yes
Wikipedia Field References	yes
WordNet	yes
UMLS	yes
OpenThesaurus	yes
ConceptNet	no
Repository Configuration	
Concept pre-processing (GMR, SMR, SHR)	yes
Post-GMR	yes
Node degree comparison	yes
Minimal node degree quotient	1.5
Path Search	
Maximal path length	2
Termination mode	All paths
Minimal path acceptance confidence	0.4
Resource Weights	
Wikipedia (Original/Filtered/Field Ref.)	0.8
Wikipedia Redirects	0.94
WordNet	0.97
UMLS	0.98
OpenThesaurus	0.95
ConceptNet	*(disabled)*
Type Weights	
equal	0.97
is-a / inverse is-a	0.95
has-a / part-of	0.86
Special Path Weights	
Inverse path decrement (IP)	0.2
Loss of generality decrement (LoG)	0.12
WordNet path increment (WP)	0.1

Table B.1: Default settings used by SemRep.

Bibliography

[1] Rakesh Agrawal, Tomasz Imieliński, and Arun Swami. Mining Association Rules Between Sets of Items in Large Databases. In *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data*, pages 207–216, 1993.

[2] Zharko Aleksovski, Michel Klein, Warner ten Kate, and Frank van Harmelen. Matching Unstructured Vocabularies using a Background Ontology. In *Proceedings of the 15th International Conference on Knowledge Engineering and Knowledge Management (EKAW)*, pages 182–197, 2006.

[3] Alsayed Algergawy, Sabine Massmann, and Erhard Rahm. A Clustering-based Approach For Large-scale Ontology Matching. In *Proceedings of the 15th East-European Conference on Advances in Databases and Information Systems (ADBIS)*, 2011.

[4] Patrick Arnold. Semantic Enrichment of Ontology Mappings: Detecting Relation Types and Complex Correspondences. In *25. GI-Workshop Grundlagen von Datenbanken*, 2013.

[5] Patrick Arnold and Erhard Rahm. Semantic Enrichment of Ontology Mappings: A Linguistic-based Approach. In *Proceedings of the 17th East-European Conference on Advances in Databases and Information Systems (ADBIS)*, 2013.

[6] Patrick Arnold and Erhard Rahm. Enriching Ontology Mappings with Semantic Relations. *Data and Knowledge Engineering*, 93:1–18, 2014.

[7] Patrick Arnold and Erhard Rahm. Extracting Semantic Concept Relations from Wikipedia. In *Proceedings of the 4th International Conference on Web Intelligence, Mining and Semantics (WIMS)*, 2014.

[8] Patrick Arnold and Erhard Rahm. Automatic Extraction of Semantic Relations from Wikipedia. *International Journal on Artificial Intelligence Tools (IJAIT)*, 24(2), 2015.

[9] Patrick Arnold and Erhard Rahm. SemRep: A Repository for Semantic Mapping. In *16. Fachtagung "Datenbanksysteme für Business, Technologie und Web" (BTW)*, 2015.

[10] Mark Aronoff and Janie Rees-Miller. *The Handbook of Linguistics*. Blackwell Publishers Ltd, 2001.

[11] Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. DBpedia: A Nucleus for a Web of Open Data. In *Proceedings of the 6th International Semantic Web and 2nd Asian Conference on Asian Semantic Web Conference*, pages 722–735, 2007.

[12] Sören Auer and Jens Lehmann. What Have Innsbruck and Leipzig in Common? Extracting Semantics from Wiki Content. In *Proceedings of the 4th European Semantic Web Conference (ESWC)*, pages 503–517, 2007.

[13] Laurie Bauer. *English Word-Formation*. Cambridge University Press, 1983.

[14] Laurie Bauer. Exocentric compounds. *Morphology*, 18(1):51–74, 2008.

[15] Rohan Baxter, Peter Christen, and Tim Churches. A Comparison of Fast Blocking Methods for Record Linkage. In *Proceedings of the ACM Workshop Data Cleaning, Record Linkage and Object Consolidation*, pages 253–291, 2003.

[16] Zohra Bellahsene, Angela Bonifati, Fabien Duchateau, and Yannis Velegrakis. On evaluating schema matching and mapping. In *Schema Matching and Mapping*, pages 253–291. Springer, 2011.

[17] Zohra Bellahsene, Angela Bonifati, and Erhard Rahm. *Schema Matching and Mapping*. Springer, 2011.

[18] Matthew Berland and Eugene Charniak. Finding Parts in Very Large corpora. In *Proceedings of the 37th annual meeting of the Association for Computational Linguistics on Computational Linguistics*, pages 57–64, 1999.

[19] Philip A. Bernstein and Sergey Melnik. Model management 2.0: Manipulating richer mappings. In *Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data*, pages 1–12, 2007.

[20] Olivier Bodenreider. The Unified Medical Language System (UMLS): integrating biomedical terminology. *Nucleic Acids Research*, 32:267–270, 2004.

[21] Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. Freebase: A Collaboratively Created Graph Database for Structuring Human Knowledge. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, pages 1247–1250, 2008.

[22] Danushka Bollegala, Yutaka Matsuo, and Mitsuru Ishizuka. Measuring Semantic Similarity Between Words Using Web Search Engines. In *Proceedings of the 16th International Conference on World Wide Web*, pages 757–766, 2007.

[23] Marco De Boni and Suresh Manandhar. Automated Discovery of Telic Relations for WordNet. In *Proceedings of the First International Conference on General WordNet*, 2002.

[24] Angela Bonifati, Elaine Qing Chang, Aks V. S. Lakshmanan, Terence Ho, and Rachel Pottinger. HePToX: Marrying XML and Heterogeneity in Your P2P Databases. In *Proceedings of the 31st International Conference on Very Large Data Bases (VLDB)*, pages 1267–1270. VLDB Endowment, 2005.

[25] Angela Bonifati, Giansalvatore Mecca, Alessandro Pappalardo, Salvatore Raunich, and Gianvito Summa. Schema Mapping Verification: The Spicy Way. In *Proceedings of the 11th International Conference on Extending Database Technology: Advances in Database Technology*, pages 85–96, 2008.

[26] David Guy Brizan and Abdullah Uz Tansel. A Survey of Entity Resolution and

Record Linkage Methodologies. *Communications of the International Information Management Association (IIMA)*, 6(3):41–50, 2006.

[27] Luca Cardelli. A Semantics of Multiple Inheritance. In *Proceedings of the International Symposium on Semantics of Data Types*, 1984.

[28] Andrew Carstairs-McCarthy. *An Introduction to English Morphology*. Edinburgh University Press Ltd, 2002.

[29] S. Castano, A. Ferrara, and S. Montanelli. H-MATCH: an Algorithm for Dynamically Matching Ontologies in Peer-based Systems. In *Proceedings of the 1st International Workshop on Semantic Web and Databases (SWDB)*, pages 231–250, 2003.

[30] Silvana Castano, Alfio Ferrara, Davide Lorusso, and Stefano Montanelli. The HMatch 2.0 Suite for Ontology Matchmaking. In *Proceedings of the 4th Italian Workshop on Semantic Web Applications and Perspectives (SWAP)*, 2007.

[31] Junpeng Chen and Juan Liu. Combining ConceptNet and WordNet for Word Sense Disambiguation. In *Proceedings of the Fifth International Joint Conference on Natural Language Processing*, 2011.

[32] Timothy Chklovski and Patrick Pantel. VerbOcean: Mining the Web for Fine-Grained Semantic Verb Relations. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 33–40. Association for Computational Linguistics, 2004.

[33] Victor Christen. Verfahren für die Reparatur von Ontologie-Mappings in den Lebenswissenschaften. Master's thesis, Universität Leipzig, 2014.

[34] Watson W. K. Chua and Jung-jae Kim. Discovering Cross-Ontology Subsumption Relationships by Using Ontological Annotations on Biomedical Literature. In *International Conference of Biomedical Ontologies (ICBO)*, 2012.

[35] William W. Cohen, Pradeep Ravikumar, and Stephen E. Fienberg. A Comparison of String Distance Metrics for Name-Matching Tasks. In *Proceedings of the IJCAI-03 Workshop on Information Integration on the Web (IIWeb-03)*, pages 73–78, 2003.

[36] Isabel F. Cruz, Flavio Palandri Antonelli, and Cosmin Stroe. AgreementMaker: Efficient Matching for Large Real-World Schemas and Ontologies. *Proceedings of the VLDB Endowment*, 2:1586–1589, 2009.

[37] Jérôme David, Fabrice Guillet, and Henri Briand. Matching Directories and OWL Ontologies with AROMA. In *Proceedings of the 15th ACM International Conference on Information and Knowledge Management*, 2006.

[38] Jérôme David, Fabrice Guillet, and Henri Briand. Association Rule Ontology Matching Approach. *International Journal on Semantic Web & Information Systems*, 3(2):27–49, 2007.

[39] Gerard de Melo and Gerhard Weikum. Towards a Universal Wordnet by Learning from Combined Evidence. In *Proceedings of the 18th ACM Conference on Information and Knowledge Management*, pages 513–522, 2009.

[40] Robin Dhamankar, Yoonkyong Lee, Anhai Doan, Alon Halevy, and Pedro Domin-

gos. iMAP: Discovering Complex Semantic Matches between Database Schemas. In *Proceedings of the 2004 ACM SIGMOD International Conference on Management of Data*, 2004.

[41] Hong-Hai Do and Erhard Rahm. COMA - A System for Flexible Combination of Schema Matching Approaches. In *Proceedings of the 28th International Conference on Very Large Databases (VLDB)*, 2002.

[42] AnHai Doan and Alon Y. Halevy. Semantic-integration Research in the Database Community: A Brief Survey. *AI Magazine*, 26(1):83–94, 2005.

[43] AnHai Doan, Jayant Madhavan, Pedro Domingos, and Alon Halevy. Ontology Matching: A Machine Learning Approach. In *Handbook on Ontologies*, 2004.

[44] Judith Eckle-Kohler, Iryna Gurevych, Silvana Hartmann, Michael Matuschek, and Christian M. Meyer. UBY-LMF - A Uniform Model for Standardizing Heterogeneous Lexical-Semantic Resources in ISO-LMF. In *Proceedings of the 8th International Conference on Language Resources and Evaluation (LREC)*, pages 275–282, 2012.

[45] Ahmed K. Elmagarmid, Panagiotis G. Ipeirotis, and Vassilios S. Verykios. Duplicate Record Detection: A Survey. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 19(1), 2007.

[46] Jérôme Euzenat, Alfio Ferrara, Willem Robert van Hage, Laura Hollink, Christian Meilicke, Andriy Nikolov, Dominique Ritze, François Scharffe, Pavel Shvaiko, Heiner Stuckenschmidt, Ondrej Sváb-Zamazal, and Cássia Trojahn dos Santos. Final Results of the Ontology Alignment Evaluation Initiative 2011. In *Proceedings of the ISWC 2011 Workshop on Ontology Matching (OM)*, 2011.

[47] Jérôme Euzenat, Christian Meilicke, Heiner Stuckenschmidt, Pavel Shvaiko, Cássia Trojahn, and Stefano Spaccapietra. Ontology Alignment Evaluation Initiative: Six Years of Experience. In *Journal on Data Semantics XV*, pages 158–192. 2011.

[48] Christiane Fellbaum. *WordNet: An Electronic Lexical Database*. Bradford Books, 1998.

[49] Charles J. Fillmore, Christopher R. Johnson, and Miriam R. L. Petruck. Background to FrameNet. *International Journal of Lexicography*, 16(3):235–250, 2003.

[50] Tiziano Flati and Roberto Navigli. SPred: Large-scale Harvesting of Semantic Predicates. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, pages 1222–1232, 2013.

[51] Evgeniy Gabrilovich and Shaul Markovitch. Computing Semantic Relatedness Using Wikipedia-based Explicit Semantic Analysis. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence*, pages 1606–1611, 2007.

[52] Avigdor Gal. Managing Uncertainty in Schema Matching with Top-K Schema Mappings. *Journal on Data Semantics VI*, 6:90–114, 2006.

[53] Avigdor Gal. Why is Schema Matching Tough and What Can We Do About It? *SIGMOD Rec.*, 35(4):2–5, December 2006.

[54] Michael Gasser. The Origins of Arbitrariness in Language. In *Proceedings of the 26th Annual Conference of the Cognitive Science Society*, 2004.

[55] Roxana Girju, Adriana Badulescu, and Dan Moldovan. Learning Semantic Constraints for the Automatic Discovery of Part-Whole Relations. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language*, 2003.

[56] Fausto Giunchiglia, Aliaksandr Autayeu, and Juan Pane. S-match: An Open Source Framework for Matching Lightweight Ontologies. *Semantic Web*, 3(3):307–317, 2012.

[57] Fausto Giunchiglia, Pavel Shvaiko, and Mikalai Yatskevich. S-match: an Algorithm and an Implementation of Semantic Matching. In *Proceedings of the First European Semantic Web Symposium (ESWS)*, pages 61–75, 2004.

[58] Fausto Giunchiglia, Mikalai Yatskevich, and Pavel Shvaiko. Semantic Matching: Algorithms and Implementation. In *Journal on Data Semantics IX*, pages 1–38. Springer-Verlag, 2007.

[59] Risto Gligorov, Warner ten Kate, Zharko Aleksovski, and Frank van Harmelen. Using Google Distance to Weight Approximate Ontology Matches. In *Proceedings of the 16th International Conference on World Wide Web*, pages 767–776, 2007.

[60] Jorge Gracia, Jordi Bernad, and Eduardo Mena. Ontology Matching with CIDER: evaluation report for OAEI 2011. In *Proceedings of the ISWC 2011 Workshop on Ontology Matching (OM)*, 2011.

[61] Jorge Gracia, Vanessa Lopez, Mathieu d'Aquin, Marta Sabou, Enrico Motta, and Eduardo Mena. Solving Semantic Ambiguity to Improve Semantic Web based Ontology Matching. In *Proceedings of the 2nd International Workshop on Ontology Matching (OM-2007) collocated with the 6th International Semantic Web Conference (ISWC-2007) and the 2nd Asian Semantic Web Conference (ASWC-2007)*, 2007.

[62] Anika Groß, Julio Cesar Dos Reis, Michael Hartung, Cédric Pruski, and Erhard Rahm. Semi-Automatic Adaptation of Mappings between Life Science Ontologies. In *Proceedings of the 9th International Conference on Data Integration in the Life Sciences (DILS)*, 2013.

[63] Anika Groß, Michael Hartung, Andreas Thor, and Erhard Rahm. How do computed ontology mappings evolve? - A case study for life science ontologies. In *Proceedings of the Joint Workshop on Knowledge Evolution and Ontology Dynamics*, 2012.

[64] Thomas R. Gruber. A Translation Approach to Portable Ontology Specifications. *Knowledge Acquisition*, 5(2):199–220, 1993.

[65] Willem Robert Van Hage. A Method to Combine Linguistic Ontology-Mapping Techniques. In *Proceedings of the 4th International Semantic Web Conference (ISWC)*, pages 732–744, 2005.

[66] Patrick A. V. Hall and Geoff R. Dowling. Approximate String Matching. *ACM Computing Surveys*, 12(4):381–402, 1980.

[67] Fayçal Hamdi, Brigitte Safar, Nobal B. Niraula, and Chantal Reynaud. TaxoMap in the OAEI 2009 alignment contest. In *Proceedings of the 4th International Workshop on Ontology Matching (OM-2009) collocated with the 8th International Semantic Web*

Conference (ISWC-2009), 2009.

[68] Fayçal Hamdi, Brigitte Safar, Nobal B. Niraula, and Chantal Reynaud. TaxoMap alignment and refinement modules: Results for OAEI 2010. In *Proceedings of the 5th International Workshop on Ontology Matching (OM-2010) collocated with the 9th International Semantic Web Conference (ISWC-2010)*, 2010.

[69] Michael Hartung, Anika Groß, and Erhard Rahm. COnto-Diff: Generation of Complex Evolution Mappings for Life Science Ontologies. *Journal of Biomedical Informatics*, 46:15–32, 2013.

[70] Bin He and Kevin Chen-Chuan Chang. Making Holistic Schema Matching Robust: An Ensemble Approach. In *Proceedings of the 11th ACM SIGKDD international conference on knowledge discovery and data mining*, pages 429–438, 2005.

[71] Marti A. Hearst. Automatic Acquisition of Hyponyms from Large Text Corpora. In *Proceedings of the 14th Conference on Computational Linguistics (COLING) - Volume 2*, pages 539–545, 1992.

[72] Aurelie Herbelot and Ann Copestake. Acquiring Ontological Relationships from Wikipedia Using RMRS. In *Proceedings of the ISWC 2006 Workshop on Web Content Mining with Human Language Technologies*, 2006.

[73] Ming-Hung Hsu, Ming-Feng Tsai, and Hsin-Hsi Chen. Query Expansion with ConceptNet and WordNet: An Intrinsic Comparison. In *Proceedings of the Third Asia Conference on Information Retrieval Technology*, pages 1–13, 2006.

[74] Ming-Hung Hsu, Ming-Feng Tsai, and Hsin-Hsi Chen. Combining WordNet and ConceptNet for Automatic Query Expansion: A Learning Approach. In *Proceedings of the 4th Asia Information Retrieval Symposium*, pages 213–224, 2008.

[75] Wei Hu and Yuzhong Qu. Block Matching for Ontologies. In *Proceedings of the 5th International Semantic Web Conference (ISWC)*, pages 300–313, 2006.

[76] Wei Hu and Yuzhong Qu. Discovering Simple Mappings Between Relational Database Schemas and Ontologies. In *Proceedings of the 6th International Semantic Web and 2nd Asian Conference on Asian Semantic Web Conference*, pages 225–238, 2007.

[77] Wei Hu and Yuzhong Qu. Falcon-AO: A practical ontology matching system. *Journal of Web Semantics*, 6:237–239, 2008.

[78] Yves R. Jean-Mary and Mansur R. Kabuka. ASMOV: Ontology Alignment with Semantic Validation. In *Joint SWDB-ODBIS Workshop*, 2007.

[79] Yves R. Jean-Mary, E. Patrick Shironoshita, and Mansur R. Kabuka. ASMOV: Results for OAEI 2009. In *Proceedings of the 4th International Workshop on Ontology Matching (OM-2009) collocated with the 8th International Semantic Web Conference (ISWC-2009)*, 2009.

[80] Yves R. Jean-Mary, E. Patrick Shironoshita, and Mansur R. Kabuka. ASMOV: Results for OAEI 2010. In *Proceedings of the 5th International Workshop on Ontology Matching (OM-2010) collocated with the 9th International Semantic Web Conference (ISWC-2010)*, 2010.

[81] Ernesto Jiménez-Ruiz and Bernardo Cuenca Grau. LogMap: Logic-Based and Scalable Ontology Matching. In *Proceedings of the 10th International Semantic Web Confernece (ISWC)*, pages 273–288, 2011.

[82] Ernesto Jiménez-Ruiz, Christian Meilicke, Bernardo Cuenca Grau, and Ian Horrocks. Evaluating Mapping Repair Systems with Large Biomedical Ontologies. In *The 26th International Workshop on Description Logics*, 2013.

[83] Claudia Kunze and Lothar Lemnitzer. *Computerlexicographie*. Narr Francke Attempto Verlag GmbH, 2007.

[84] Jung Ae Kwak and Hwan-Seung Yong. Ontology Matching Based on Hypernym, Hyponym, Holonym, and Meronym Sets in WordNet. *International Journal of Web & Semantic Technology (IJWesT)*, 1(2):1–14, 2010.

[85] Patrick Lambrix and He Tan. SAMBO - a system for aligning and merging biomedical ontologies. *Journal of Web Semantics*, 4(1):196–206, 2006.

[86] Douglas Lenat. CYC: A Large-Scale Investment in Knowledge Infrastructure. *Communications of the ACM*, 38(11):33–38, 1995.

[87] Yi Li, Qian Zhong, Juanzi Li, and Jie Tang. Result of ontology alignment with RiMOM at OAEI'06. In *Proceedings of the ISWC 2006 Workshop on Ontology Matching (OM)*, 2006.

[88] Rochelle Lieber and Pavol Štekauer. *The Oxford Handbook of Compounding*. Oxford University Press, 2011.

[89] Dekang Lin. Automatic Retrieval and Clustering of Similar Words. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics*, 1998.

[90] H. Liu and P. Singh. ConceptNet - a practical commonsense reasoning tool-kit. *BT Technology Journal*, 22(4):211–226, 2004.

[91] Yoëlle S. Maarek, Daniel M. Berry, and Gail E. Kaiser. An Information Retrieval Approach for Automatically Constructing Software Libraries. *IEEE Transactions on Software Engineering*, 17(8):800–813, 1991.

[92] Jayant Madhavan, Philip A. Bernstein, AnHai Doan, and Alon Halevy. Corpus-Based Schema Matching. In *Proceedings of the 21st International Conference on Data Engineering*, pages 57–68, 2005.

[93] Jayant Madhavan, Philip A. Bernstein, and Erhard Rahm. Generic Schema Matching with Cupid. In *Proceedings of the 27th International Conference on Very Large Data Bases (VLDB)*, 2001.

[94] Bruno Marnette, Giansalvatore Mecca, Paolo Papotti, Salvatore Raunich, and Donatello Santoro. ++Spicy: an OpenSource Tool for Second-Generation Schema Mapping and Data Exchange. *Proceedings of the VLDB Endowment*, 4(12):1438–1441, 2011.

[95] Sabine Massmann and Erhard Rahm. Evaluating Instance-based Matching of Web Directories. In *11th International Workshop on the Web and Databases (WebDB)*, 2008.

[96] Robert McCann, Warren Shen, and AnHai Doan. Matching schemas in online communities: A web 2.0 approach. In *Proceedings of the 2008 IEEE 24th International Conference on Data Engineering*, pages 110–119, 2008.

[97] Christian Meilicke. *Alignment Incoherence in Ontology Matching*. PhD thesis, University of Mannheim, Chair of Artificial Intelligence, 2011.

[98] Christian Meilicke and Heiner Struckenschmidt. Applying Logical Constraints to Ontology Matching. In *Proceedings of the 30th Annual German Conference on Advances in Artifical Intelligence (AI)*, 2007.

[99] George A. Miller, Richard Beckwith, Christiane Fellbaum, Derek Gross, and Katherine Miller. WordNet: An on-line lexical database. *International Journal of Lexicography*, 3(4):235–312, 1990.

[100] Alexandra Moraru, Klemen Kenda, Blaž Fortuna, Luka Bradeško, Maja Škrjanc, Dunja Mladenić, and Carolina Fortuna. Supporting Rule Generation and Validation on Environmental Data in EnStreaM. In *The Semantic Web: ESWC 2012 Satellite Events*, pages 441–446, 2012.

[101] Felix Naumann and Ulf Leser. *Informationsintegration - Architekturen und Methoden zur Integration verteilter und heterogener Datenquellen*. dpunkt.verlag, 2007.

[102] Roberto Navigli and Simone Paolo Ponzetto. BabelNet: Building a Very Large Multilingual Semantic Network. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 216–225, 2010.

[103] DuyHoa Ngo and Zohra Bellahsene. YAM++: A Multi-strategy Based Approach for Ontology Matching Task. In *Knowledge Engineering and Knowledge Management*, 2012.

[104] Quoc Viet Hung Nguyen, Tam Nguyen Thanh, Zoltán Miklós, and Karl Aberer. On Leveraging Crowdsourcing Techniques for Schema Matching Networks. In *18th International Conference on Database Systems for Advanced Applications (DASFAA)*, 2013.

[105] Andriy Nikolov, Victoria Ure, Enrico Motta, and Anne de Roeck. Overcoming Schema Heterogeneity between Linked Semantic Repositories to Improve Coreference Resolution. In *Proceedings of the Fourth Asian Conference (ASWC)*, 2009.

[106] Natalya F. Noy, Nicholas Griffith, and Mark A. Musen. Collecting Community-Based Mappings in an Ontology Repository. In *Proceedings of the 7th International Semantic Web Conference (ISWC)*, 2008.

[107] Catia Pesquita, Daniel Faria, Emanuel Santos, and Francisco M. Couto. To repair or not to repair: reconciling correctness and coherence in ontology reference alignments. In *Proceedings of the 8th International Workshop on Ontology Matching (OM-2013) co-located with the 12th International Semantic Web Conference (ISWC 2013)*, pages 13–24, 2013.

[108] Eric Peukert, Julian Eberius, and Erhard Rahm. A Self-Configuring Schema Matching System. In *Proceedings of the 28th International Conference on Data Engineering (ICDE)*, 2012.

[109] Eric Peukert, Sabine Massmann, and Kathleen König. Comparing Similarity Com-

bination Methods for Schema Matching. In *GI-Workshop - Informationsintegration in Service-Architekturen*, 2010.

[110] Eric Peukert and Katja Pfeifer. Mapping Text Mining Taxonomies. In *The 6th International Conference on Knowledge and Information Retrieval*, 2013.

[111] Emanuele Pianta, Luisa Bentivogli, and Christian Girardi. MultiWordNet: developing an aligned multilingual database. In *Proceedings of the First International Conference on Global WordNet*, 2002.

[112] Ingo Plag, Maria Braun, Sabine Lappe, and Mareile Schramm. *Introduction to English Linguistics, 2nd revised edition*. Mouton de Gruyter, 2007.

[113] Aleksander Pohl. Classifying the Wikipedia articles into the OpenCyc taxonomy. In *Proceedings of the Web of Linked Entities Workshop in conjuction with the 11th International Semantic Web Conference (ISWC)*, 2012.

[114] Simone Paolo Ponzetto and Roberto Navigli. Knowledge-rich Word Sense Disambiguation Rivaling Supervised Systems. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, 2010.

[115] Simone Paolo Ponzetto and Michael Strube. Deriving a Large Scale Taxonomy from Wikipedia. In *Proceedings of the 22nd National Conference on Artificial Intelligence*, pages 1440–1445, 2007.

[116] Simone Paolo Ponzetto and Michael Strube. WikiTaxonomy: A large scale knowledge resource. In *Proceedings of the 18th European Conference on Artificial Intelligence*, 2008.

[117] Lucian Popa, Mauricio A. Hernández, Yannis Velegrakis, Renée J. Miller, Felix Naumann, and Howard Ho. Mapping XML and Relational Schemas with Clio. In *Proceedings of the 18th International Conference on Data Engineering*, pages 498–499, 2002.

[118] Rachel Pottinger. Mapping-Based Merging of Schemas. In *Schema Matching and Mapping*, chapter 8, pages 223–249. Springer, 2010.

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

[120] Erhard Rahm and Hong-Hai Ho. Data Cleaning: Problems and Current Approaches. *IEEE Bulletin of the Technical Committee on Data Engineering*, 23(4):3–13, 2000.

[121] Salvatore Raunich and Erhard Rahm. ATOM: Automatic Target-driven Ontology Merging. In *Proceedings of the International Conference on Data Engineering*, 2011.

[122] Chantal Reynaud and Brigitte Safar. Exploiting WordNet as Background Knowledge. In *Proceedings of the 2nd International Workshop on Ontology Matching (OM-2007) collocated with the 6th International Semantic Web Conference (ISWC-2007) and the 2nd Asian Semantic Web Conference (ASWC-2007)*, 2007.

[123] Dominique Ritze, Christian Meilicke, Ondrej Sváb-Zamazal, and Heiner Stuckenschmidt. A Pattern-based Ontology Matching Approach for Detecting Complex Correspondences. In *Proceedings of the 4th International Workshop on Ontology Match-*

ing (OM-2009) collocated with the 8th International Semantic Web Conference (ISWC-2009), 2009.

[124] Maria Ruiz-Casado, Enrique Alfonseca, and Pablo Castells. Automatic extraction of semantic relationships for WordNet by means of pattern learning from Wikipedia. In *Proceedings of the 10th international conference on Natural Language Processing and Information Systems*, pages 67–79, 2005.

[125] Maria Ruiz-Casado, Enrique Alfonseca, and Pablo Castells. Automatising the Learning of Lexical Patterns: an Application to the Enrichment of WordNet by Extracting Semantic Relationships from Wikipedia. *Journal of Data and Knowledge Engineering*, 61(3):484–499, 2007.

[126] Marta Sabou, Mathieu d'Aquin, and Enrico Motta. Using the Semantic Web as Background Knowledge for Ontology Mapping. In *Ontology Matching*, 2006.

[127] Emanuel Santos, Daniel Faria, Catia Pesquita, and Francisco Couto. Ontology alignment repair through modularization and confidence-based heuristics. *arXiv preprint, arXiv:1307.5322*, 2013.

[128] Christina Sarasua, Elena Simperl, and Natalya F. Noy. CrowdMap: Crowdsourcing Ontology Alignment with Microtasks. In *Proceedings of the 11th International Semantic Web Conference (ISWC)*, pages 525 – 541, 2012.

[129] Sergio Scalise and Antonietta Bisetto. The classification of compounds. In *The Oxford Handbook of Compounding*, pages 34–53. Oxford University Press, 2009.

[130] Frederik C. Schadd and Nico Roos. Improving ontology matchers utilizing linguistic ontologies: an information retrieval approach. In *Proceedings of the 23rd Belgian-Dutch Conference on Artificial Intelligence*, 2011.

[131] Luciano Serafini, Paolo Bouquet, Bernardo Magnini, and Stefano Zanobini. An algorithm for matching contextualized schemas via SAT. Technical report, DIT University of Trento, 2003.

[132] Pavel Shvaiko and Jérôme Euzenat. A Survey of Schema-based Matching Approaches. *Journal on Data Semantics*, 4:146–171, 2005.

[133] Pavel Shvaiko and Jérôme Euzenat. Ontology Matching: State of the Art and Future Challenges. *IEEE Transactions on Knowledge and Data Engineering*, 25:158 – 176, 2013.

[134] Barry Smith. The Logic of Biological Classification and the Foundations of Biomedical Ontology. In *Handbook on Ontologies: Invited Papers from the 10th International Conference in Logic Methodology and Philosophy of Science*, 2005.

[135] Rion Snow, Daniel Jurafsky, and Andrew Y. Ng. Learning syntactic patterns for automatic hypernym discovery. In *Advances in Neural Information Processing Systems (NIPS)*, 2004.

[136] Alessandro Solimando, Ernesto Jiménez-Ruiz, and Giovanna Guerrini. Detecting and Correcting Conservativity Principle Violations in Ontology-to-Ontology Mappings. In *Proceedings of the 13th International Semantic Web Conference (ISWC)*, 2014.

[137] Robert Speer and Catherine Havasi. Representing General Relational Knowledge in ConceptNet 5. In *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC)*, 2012.

[138] Vassilis Spiliopoulos, George A. Vouros, and Vangelis Karkaletsis. On the Discovery of Subsumption Relations for the Alignment of Ontologies. *Web Semantics*, 8(1):69–88, 2010.

[139] Michael Strube and Simone Paolo Ponzetto. Wikirelate! Computing semantic relatedness using Wikipedia. In *Proceedings of the 21st National Conference on Artificial Intelligence*, 2006.

[140] Weifeng Su, Jiying Wang, and Frederick Lochovsky. Holistic Schema Matching for Web Query Interfaces. In *10th International Conference on Extending Database Technology*, 2006.

[141] Xiaomeng Su and Jon Atle Gulla. Semantic Enrichment for Ontology Mapping. In *9th International Conference on Applications of Natural Language to Information Systems*, pages 217–228, 2004.

[142] Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. Yago: A Core of Semantic Knowledge. In *Proceedings of the 16th International Conference on World Wide Web (WWW)*, pages 697–706, 2007.

[143] Asuka Sumida and Kentaro Torisawa. Hacking Wikipedia for Hyponymy Relation Acquisition. In *Proceedings of the International Joint Conference on Natural Language Processing*, pages 883–888, 2008.

[144] Jie Tang, Juanzi Li, Bangyong Liang, Xiaotong Huang, Yi Li, and Kehong Wang. Using Bayesian Decision for Ontology Mapping. *Web Semantics: Science, Services and Agents on the World Wide Web*, 4(4):243–262, 2006.

[145] Jie Tang, Bang-Yong Liang, Juanzi Li, and Kehong Wang. Risk Minimization Based Ontology Mapping. In *Advanced Workshop on Content Computing (AWCC)*, pages 469 – 480, 2004.

[146] Andreas Thor, Toralf Kirsten, and Erhard Rahm. Instance-based matching of hierarchical ontologies. In *12. Fachtagung "Datenbanksysteme für Business, Technologie und Web" (BTW)*, 2007.

[147] Jörg Tiedemann and Lars Nygaard. The OPUS Corpus - Parallel and Free. In *Proceedings of the Fourth International Conference on Language Resources and Evaluation*, 2004.

[148] Raquel Trillo, Jorge Gracia, Mauricio Espinoza, and Eduardo Mena. Discovering the Semantics of User Keywords. *Journal on Universal Computer Science. Special Issue: Ontologies and their Applications*, 13(12):1908–1935, 2007.

[149] Nwe Ni Tun. Semantic Enrichment in Ontologies for Matching. In *Proceedings of the Second Australasian Workshop on Advances in Ontologies*, pages 91–100, 2006.

[150] Peng Wang and Baowen Xu. Debugging ontology mappings: a static approach. *Computing and Informatics*, 27(1):21–36, 2008.

[151] Michael Witbrock, Kathy Panton, Stephen L. Reed, Dave Schneider, Björn Aldag, Mike Reimers, and Stefano Bertolo. Automated OWL Annotation Assisted by a Large Knowledge Base. In *Workshop Notes of the 2004 Workshop on Knowledge Markup and Semantic Annotation at the 3rd International Semantic Web Conference (ISWC)*, pages 71–80, 2004.

[152] Wentao Wu, Hongsong Li, Haixun Wang, and Kenny Zhu. Probase: A probabilistic taxonomy for text understanding. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, 2012.

[153] Li Xu and David W. Embley. Using Domain Ontologies to Discover Direct and Indirect Matches for Schema Elements. In *Proceedings of the Semantic Integration Workshop at the International Semantic Web Conference (ISWC)*, 2003.

[154] Peigang Xu, Yadong Wang, Liang Cheng, and Tianyi Zang. Alignment Results of SOBOM for OAEI 2010. In *Proceedings of the 5th International Workshop on Ontology Matching (OM-2010) collocated with the 9th International Semantic Web Conference (ISWC-2010)*, 2010.

[155] Mikhail Zaslavskiy, Francis Bach, and Jean-Philippe Vert. Many-to-many Graph Matching: A Continuous Relaxation Approach. In *Proceedings of the 2010 European Conference on Machine Learning and Knowledge Discovery in Databases: Part III*, pages 515–530, 2010.

[156] Songmao Zhan and Olivier Bodenreider. Alignment of multiple ontologies of anatomy: Deriving indirect mappings from direct mappings to a reference. In *Proceedings of Annual Symposium of American Medical Informatics Association (AMIA)*, pages 864–868, 2005.

[157] Xiao Zhang, Qian Zhong, Juanzi Li, and Jie Tang. RiMOM Results for OAEI 2009. In *Proceedings of the ISWC 2009 Workshop on Ontology Matching (OM)*, 2009.

[158] Cäcilia Zirn, Vivi Nastase, and Michael Strube. Distinguishing Between Instances and Classes in the Wikipedia Taxonomy. In *Proceedings of the 5th European Semantic Web Conference (ESWC)*, pages 376–387, 2008.

Wissenschaftlicher Werdegang

Persönliche Angaben: Patrick Arnold
geboren am 26. Juli 1987 in Leipzig

Schulbildung:

08/1994 – 07/1998	157. Grundschule, Leipzig
09/1998 – 07/2005	Uhlandschule, Gymnasium der Stadt Leipzig
08/2005 – 06/2006	Robert-Schumann-Gymnasium Leipzig

Ausbildung und Beruf:

10/2006 – 10/2009	Informatikstudium an der Universität Leipzig Abschluss: B.Sc. in Informatik
10/2009 – 10/2011	Informatikstudium an der Universität Leipzig Abschluss: M.Sc. in Informatik
04/2010 – 01/2012	Studentische/Wissenschaftliche Hilfskraft am WDI-Lab (Abteilung Datenbanken, Universität Leipzig) ⋆ Mitarbeit im Bereich des Schema Matchings
02/2012 – heute	Wissenschaftlicher Mitarbeiter am Institut für Angewandte Informatik Leipzig e.V. (InfAI) bzw. an der Universität Leipzig, Abteilung Datenbanken ⋆ Forschung und Mitarbeit im Bereich Produktpiraterie (Februar 2012 – Dezember 2012) ⋆ Mitarbeit im EU-Projekt Linked Design (Februar 2012 – Februar 2015) ⋆ Forschung im Gebiet Schema- und Ontology Mapping (ab Februar 2012)

Bibliografische Angaben

Arnold, Patrick: Semantic Enrichment of Ontology Mappings. Dissertation, Universität Leipzig, 2015.

Selbständigkeitserklärung

Hiermit erkläre ich, die vorliegende Dissertation selbständig und ohne unzulässige fremde Hilfe angefertigt zu haben. Ich habe keine anderen als die angeführten Quellen und Hilfsmittel benutzt und sämtliche Textstellen, die wörtlich oder sinngemäß aus veröffentlichten oder unveröffentlichten Schriften entnommen wurden, und alle Angaben, die auf mündlichen Auskünften beruhen, als solche kenntlich gemacht. Ebenfalls sind alle von anderen Personen bereitgestellten Materialien oder erbrachten Dienstleistungen als solche gekennzeichnet.

Leipzig, den 17. Dezember 2015 Patrick Arnold