Chapter 2

Background

2.1 Introduction
This chapter provides the details related to basics of ontology and mapping. Then it discusses state of the art research on ontology mapping approach in detail.

2.2 Ontology
In recent years development of ontologies has been moving from the realm of Artificial Intelligence laboratories to the desktop of domain experts. Ontologies have become common on the World Wide Web. In philosophy, ontology is a theory about the nature of existence, of what kinds of things exist. Ontology as a discipline studies such theories. Artificial-intelligence and Web researchers have co-opted the term for their own jargon, and for them ontology is a document or file that formally defines the relations among terms. In computer science ontology is defined as formal explicit specification of shared conceptualization. Where formal means machine understandable. Concepts, properties, axioms, and functions are explicitly defined. Shared refers to consensual knowledge. Conceptualization is an abstract model of some phenomenon in the world [Grubber1993]. The most typical kind of ontology for the Web has taxonomy and a set of inference rules [Tim Berners Lee2001]. Ontologies on the Web ranges from large taxonomies categorizing websites (such as Yahoo) to categorization of products for their sale and their features (such as Amazon .com). The WWW Consortium (W3C) has developed the
Resource Description Framework [Brickley and Guha 1999], a language for encoding knowledge on Web pages to make it understandable to electronic agents searching for information. A new language called Web Ontology Language (OWL) for defining ontologies is used. Many disciplines have now developed standardized ontologies that domain experts can use to share and annotate information in their fields. Medicine, as an example, has produced large, standardized, structured vocabularies such as Systematized Nomenclature of Medicine (SNOMED) [Price and Spackman 2000] and the semantic network of the Unified Medical Language System [Humphreys and Lindberg 1993]. Broad general-purpose ontologies are emerging as well. For example, the United Nations Development Program and Dun & Bradstreet combined their efforts to develop the United Nations Standard Products and Services Code (UNSPSC) ontology which provides terminology for products and services (www.unspsc.org). An ontology defines a common vocabulary for researchers who need to share information in a domain. It includes machine-interpretable definitions of basic concepts in the domain and relations among them.

Why would someone want to develop an ontology? Some of the reasons are:

i) Sharing common understanding of the structure of information among people or software agents is one of the more common goals in developing ontologies [Musen 1992; Gruber 1993]. For example, imagine several different Web sites contain medical information or provide medical e-commerce services. If these Web sites share and publish the same underlying ontology of the terms they all use, then computer agents can extract and aggregate information from these different sites. The agents can use this aggregated information to answer user queries or as input data to other applications.

ii) Enabling reuse of domain knowledge is one of the driving forces
behind the recent surge in ontology research. For example, models for many different domains need to represent the notion of time. This representation includes the notions of time intervals, points in time, relative measures of time, and so on. If one group of researchers develops such an ontology in detail, others can simply reuse it for their domains. Additionally, if it needs to build a large ontology, it can integrate several existing ontologies describing portions of the large domain. It can also reuse a general ontology, such as the UNSPSC ontology, and extend it to describe the needed domain of interest.

iii) Making explicit domain assumptions underlying an implementation makes it possible to change these assumptions easily if the knowledge about the domain changes.

Hard-coding assumptions about the world in programming-language, makes these assumptions not only hard to find and understand but also hard to change, particularly for someone without programming expertise. In addition, explicit specifications of domain knowledge are useful for new users who must learn what the terms in the domain mean.

iv) Separating the domain knowledge from the operational knowledge is another common use of ontologies. It can describe a task of configuring a product from its components according to a required specification and implement a program that does this configuration independent of the products and components themselves [McGuinness and Wright 1998]. It can then develop an ontology of PC-components and characteristics and apply the algorithm to configure made-to-order PCs. It can also use the same algorithm to configure elevators if it is “fed” to an elevator component ontology. [Rothenfluh et al. 1996].

v) Analyzing domain knowledge is possible once a declarative specification of the term is available.
Formal analysis of terms is extremely valuable when both are attempting to reuse existing ontologies and extending them [McGuinness et al. 2000]. Taxonomy defines classes of objects and relations among them. As an example, an address may be defined as a type of location, and city-codes may be defined to apply only to locations, and so on. Classes, subclasses and relations among entities are a very powerful tool for Web use. It is possible to express a large number of relations among entities by assigning properties to classes and allowing subclasses to inherit such properties. If city-codes must be of type city and cities generally have Web sites, it is possible to discuss the Web site associated with a city-code even if no database links a city code directly to a Web site.

Inference rules in ontologies supply further power. An ontology may express the rule "If a city code is associated with a state code, and an address uses that city code, then that address has the associated state code." A program could then readily deduce, for instance, that a Manipal University address, being in Manipal, must be in Karnataka State, which is in India and therefore should be formatted to Indian standards. The computer doesn't truly "understand" any of this information, but it can now manipulate the terms much more effectively in ways that are useful and meaningful to the human user.

With ontology pages on the Web, solutions to terminology (and other) problems begin to emerge. The meaning of terms or XML codes used on a Web page can be defined by pointers from the page to an ontology. Of course, the same problems now arise if an ontology that defines addresses as containing a zip code and point to one that uses postal code. This kind of confusion can be resolved if ontologies (or other Web services) provide equivalence relations: one or both of the ontologies may contain the information that the
zip code is equivalent to postal code. This is where ontology mapping comes in.

2.3 Ontology Mapping

Ontology mapping is also known as ontology matching, or ontology alignment. Ontology mapping or matching is different from Ontology merging. Ontology mapping tries to make the source ontologies consistent and coherent with one another while keeping them separate. In contrast, ontology merging aims to create a single coherent ontology that includes the information from all the sources. Ontology mapping is used to “establish correspondences among the source ontologies, and to determine the set of overlapping concepts, concepts that are similar in meaning but have different names or structure, and concepts that are unique to each of the sources” [Noy and Musen 2000]. It is also defined by Ehrig [Ehrig and Staab 2004] as follows. “Given two ontologies O1 and O2, mapping one ontology onto another means that for each entity (concept C, relation R, or instance I) in ontology O1, it tries to find a corresponding entity, which has the same intended meaning, in ontology O2”.

In this research, it defines ontology mapping so as to find a set of semantic correspondences between similar elements in different ontologies. Where, semantic correspondence refers to different relationships, e.g. the equivalence (=), the broader (⊇), the narrower (⊆), the disjoint (∣), the joint (∪) etc., and elements could be classes, properties, instances and relations between the instances of an ontology. Figure 2.1 shows an example for ontology mapping.
In order to illustrate the matching problem the two simple ontologies, O1 and O2 have been considered from Figure 2.1. Classes are shown in rectangles with rounded corners, e.g., in O1, Book being a specialization (subclass) of Product, while relations are shown without the latter, such as price being an attribute defined on the integer domain and creator being a property. Albert Camus: Lachute is a shared instance. Correspondences are shown as thick arrows that link an entity from O1 with an entity from O2. They are annotated with the relation that is expressed by the correspondence: for example, Person in O1 is less general than Human in O2.

Assume that an e-commerce company acquires another e-commerce company. Technically, this acquisition requires the integration of their information sources, and hence, of the ontologies of these companies. The documents or instance data of both companies are stored according to ontologies O1 and O2, respectively. In the example above these ontologies contain subsumption statements, property specifications and instance descriptions. The first step in integrating ontologies is matching, which identifies
correspondences, namely the candidate entities to be merged or to have subsumption relationships under an integrated ontology. Once the correspondences between two ontologies have been determined, they may be used, for instance, for generating query expressions that automatically translate instances of these ontologies under an integrated ontology. For example, the attributes with labels title in O1 and in O2 are the candidates to be merged, while the class with label Monograph in O2 should be subsumed by the class Product in O1.[Pavel Shvaiko 2013].

2.4 Literature Review

Several tools for ontology mapping exist in the literature. Most of these tools are semi-automatic and the design of fully automatic tools is usually a complex issue. This section outlines such state of the art methods. The approaches for ontology mapping in the literature can be classified into instance only methods, metadata only methods and hybrid ones, based upon which part of ontologies are considered for mapping. All the methods proposed in this thesis are hybrid methods. The approaches are also classified into element level, structure level and combination matchers [Namyoun Choi et.al 2006].

2.4.1. Element Level Matchers

2.4.1.1 String Based Matcher

In this the idea is more similar [Namyoun Choi et.al 2006] two strings are, more likely it is that they denote the same concept. Distance functions or variations of them are normally used. These functions map a pair of strings to a real number, where a smaller number denotes lesser similarity.

2.4.1.2 Language Based Matcher

Natural Language Processing [Namyoun Choi et.al 2006] techniques
use the morphological properties of words to identify important concepts within a source. NLP (Natural Language Processing) begins by tokenizing an input stream to locate potential words of relevance within a data source. Lemmatization then looks at each candidate word and finds all of its permutations (e.g., dog, dogs). Finally, elimination flags, articles, prepositions, conjunctions and other parts of speech which do not denote concepts for removal.

2.4.1.3 Constraint Based Matcher
In this, evaluations are made by considering data types and cardinality of attributes [Namyoun Choi et al. 2006] of entities to determine similarity between entities.

2.4.1.4 Linguistic Resources Based Matcher
Common language resources such as thesauri maintain information that can be used to ascertain [Namyoun Choi et al. 2006] whether two concepts are equal or similar.

2.4.1.5 Alignment Reuse Based Matcher
Existing mappings between domains can be employed to facilitate the mapping to new domains. For instance, when it is needed to match schema/ontology O₁ and O₂, given the alignments between O and O₁, and between O and O₂ from the external resource, storing previous match operation results. The alignment reuse is motivated by the intuition that many schemas/ontology to be matched are similar to already matched schemas/ontology, especially [Namyoun Choi et al. 2006] if they are describing the same application domain.”

2.4.1.6 Upper Level Formal Ontology
Upper level ontology are a form of external knowledge resource that can be used to ground ontology [Namyoun Choi et al. 2006] in a shared semantic context. Suggested Upper Merged Ontology (SUMO) and Descriptive Ontology for Linguistic and Cognitive Engineering (DOLCE) are examples of upper ontology currently
under development. The key characteristic of these ontology is that they are logic-based systems, and therefore, matching techniques exploiting them can be based on the analysis of interpretations.

2.4.2. Structure level matchers

2.4.2.1 Graph Based Techniques

It takes a data source as a labeled graph. It is based [Namyoun Choi et.al 2006] on the idea that if nodes from two separate ontology are related or similar, then the nodes around them are also likely to be similar. Graph Matching is computationally expensive and typically works by approximation. Variations of the technique look at child nodes, leaf nodes and the types of relation between these nodes.

2.4.2.2 Taxonomy Based Techniques

Taxonomy based matching is a form [Namyoun Choi et.al 2006] of graph matching that only looks at the specialization relationship. It works on the idea that *is-a relations* exists through nodes that are already similar, and that neighbors are then also likely to be similar.

2.4.2.3 Repository of Structures

Uses a repository to store [Namyoun Choi et.al 2006] ontology and their fragments along with some similarity coefficient. “When new structures are to be matched, they are first checked for similarity to the structures which are already available in the repository. The goal is to identify structures which are sufficiently similar to be worth matching in more detail, or reusing already existing alignments.”

2.4.2.4 Model Based Techniques

Matching is handled based [Namyoun Choi et.al 2006] on the model-theoretic semantics of the input. Techniques include Propositional Satisfiability (SAT) and Description Logics (DL).

2.4.3. Combination Matchers

These are aggregates of element and structure level matchers.
2.4.4 Instance based methods

These methods take instance information from ontologies to be mapped for matching. Some of the instance mapping methods are discussed below.

2.4.4.1 Glue

It uses machine learning techniques based on instances to find equivalent concepts between two ontologies. It neglects the structure information contained in ontology. Neither does it mention about how to map between attributes [AnHai Doan et.al. 2003].

2.4.4.2 Quick Ontology Mapping (QOM)

This uses a heuristic method, referred as dynamic programming method [Ehrig and Staab. 2004]. QOM uses ontological structures to reduce the quantity of candidate mappings in the search step selection process. In the similarity computation step, this method avoids the complete pairwise evaluation of ontology trees and restricts the costly feature comparisons. Similarity aggregation and interpretation steps are performed once per candidate mapping and therefore, do not influence efficiency. Finally iterations are performed to find mappings based on lexical knowledge and then knowledge structures. This method is faster than other methods, but optimizing the mapping operation for efficiency decreases overall mapping quality.

2.4.4.3 Information Flow-Mapping (IF-MAP)

It uses logic based on strong type checking approach. Where taxonomy structure and the types of relations are taken into account. However in the definition of classes, if the types of different attributes are the same there may be multiple possible mapping results for attributes, which are consistent with the type or taxonomy constraints. Moreover, these algorithms are not suitable for problems with large heterogeneity [Yanis Kalfoglou et.al. 2002].
2.4.4.4 Formal Concept Analysis-Merge (FCA-Merge)

It accepts two input ontologies and a set of documents of concepts in ontologies and produces a merged ontology. The domain expert with background knowledge needs to interact with the system. It uses linguistic analysis and a titanic algorithm for computation for pruned concept lattice [Stumme G. Maedche A. 2001].

2.4.5 Metadata based methods

In this, the concept names, attributes, their types, annotations, their structural relationship, etc are considered for mapping computation. Some important methods under this category are discussed below.

2.4.5.1 Lexicon Based Ontology Mapping (LOM)

This finds morphism between vocabularies in order to reduce human factor in ontology mapping using four methods, whole term, synset, word constituent, type matching. But there is no guarantee accuracy or correctness in mappings and has limitations in dealing with abstract symbols or codes in chemistry, maths and medicine [J.Li.2004].

2.4.5.2 Prompt

Two input ontology are merged to obtain one single ontology. It uses heuristics based analyzer [N.F.Noy et.al.2000]. User can suggest, accept, modify or adjust the merging process. It does not use any instance based knowledge.

2.4.5.3 Ontology Composition (ONION)

Input to this system is terms in two ontologies and output is sets of articulation rules between them. A human expert interacts with the system and is capable of deleting, choosing, modifying and suggesting matches. [Mitra and Weider Hold 2002]. It does not use any instance based knowledge.
2.4.5.4 Context Match (*CTXMATCH*)

It accepts concepts in hierarchy and gives semantic relation between them. Mapping strategy used is logical deduction. No user interaction. Instance based knowledge is not used.

2.4.6 Recent Mapping Approaches

2.4.6.1 A System for Matching and Merging Biomedical Ontologies (SAMBO)

SAMBO [P.Lambrix and H.Tan 2006] is a system for matching and merging biomedical ontologies. It handles ontologies in OWL and outputs 1:1 alignments between concepts and relations. The system uses various similarity based matchers including

2.4.6.1.1 Terminological Matcher

N-gram(a contiguous sequence of n items from a given sequence of text or speech), edit distance, comparison of the lists of words from which the terms are composed. The results of these matchers are combined via a weighted sum with pre-defined weights.

2.4.6.1.2 Structural Matcher

Through an iterative algorithm that checks if two concepts occur in similar positions with respect to is-a or part-of hierarchies relative to already matched concepts, with the intuition that the concepts under consideration are likely to be similar as well.

2.4.6.1.3 Background Knowledge Based Matcher

It uses two sub matchers. (i) a relationship between the matched entities in UMLS (Unified Medical Language System) and (ii) a corpus of knowledge collected from the published literature exploited through a naive Bayes classifier.

The results produced by these matchers are combined based on user-defined weights. Then, filtering based on thresholds is applied to come up with an alignment suggestion, which is further displayed to the user for feedback (approval, rejection or modification). Once matching has been accomplished, the system can merge the matched
ontologies, compute the consequences, check the newly created ontology for consistency, etc.

2.4.6.2 Finding, Aligning, Learning Capturing Ontology (FALCON)

Falcon is an infrastructure for Semantic Web applications, which aims at providing fundamental technology for finding, aligning and learning ontologies, and ultimately for capturing knowledge from the Web via an ontology-driven approaches. Falcon [W.Hu et.al.2008] is an automatic divide-and-conquer approach to ontology matching. It handles ontologies in Resource Description Framework Schema (RDFS) and OWL. It has been designed with the goal of dealing with large ontologies (of thousands of entities). The approach operates in three phases: (i) partitioning ontologies, (ii) matching blocks, and (iii) discovering alignments. The first phase starts with a structure-based partitioning to separate entities (classes and properties) of each ontology into a set of small clusters. Partitioning is based on structural proximities between classes and properties, e.g. how closely are the classes in the hierarchies of rdfs: subClassOf relations and on an extension of the Rock agglomerative clustering algorithm. Then it constructs blocks out of these clusters. In the second phase the blocks from distinct ontologies are matched based on anchors (pairs of entities matched in advance), i.e., the more anchors are found between two blocks, the more similar the blocks are. In turn, the anchors are discovered by matching entities with the help of the sub string comparison technique. The block pairs with high similarities are selected based on a cutoff threshold. Notice that each block is just a small fragment of an ontology. Finally, in the third phase the results of the so-called virtual documents (V-Doc) (a linguistic matcher) and graph matching algorithm for ontologies (GMO) (an iterative structural matcher) techniques are combined via sequential composition to discover alignments between the matched
block pairs. Ultimately, the output alignment is extracted through a greedy selection.

2.4.6.3 Automatic Semantic Matching of Ontologies with Verification (ASMOV)

ASMOV [Y.R.JeanMary et.al.2009] is an automatic approach for ontology matching that targets information integration for bioinformatics. Overall, the approach can be summarized in two steps: (i) similarity calculation, and (ii) semantic verification. It takes as input two OWL ontologies and an optional input alignment and returns as output an n:m alignment between ontology entities (classes and properties). In the first step, it uses lexical (string equality, a variation of Levenshtein distance), structural (weighted sum of the domain and range similarities) and extensional matchers to iteratively compute similarity measures between two ontologies, which are then aggregated into a single one as a weighted average. It also uses several sources of general and domain specific background knowledge, such as WordNet and UMLS, to provide more evidence for similarity computation. Then, it derives an alignment and checks it for inconsistency. Consistency checking is pattern based, i.e., instead of using a complete solver, the system recognizes sets of correspondences that are proved to lead to an inconsistency. The semantic verification process examines five types of patterns, e.g., disjoint-subsumption contradiction, subsumption incompleteness. This matching process is repeated with the obtained alignment as input until no new correspondences are found.

2.4.6.4 Risk Minimization Based Ontology Mapping (RiMOM)

RiMOM [Juan Zi Li et.al. 2009] is a dynamic multi-strategy ontology matching framework. It extends a previous version of the system that focused on combining multiple matching strategies, through risk minimization of Bayesian decision. The new version quantitatively estimates the similarity characteristics for each
matching task. These characteristics are used for dynamically selecting and combining the multiple matching methods. Two basic matching methods are employed: (i) linguistic similarity (edit distance over entity labels, vector distance among comments and instances of entities) and (ii) structural similarity (a variation of Similarity Flooding implemented as three similarity propagation strategies: concept-to-concept, property-to-property and concept-to-property). In turn, the strategy selection uses label and structure similarity factors, obtained as a preprocessing of the ontologies to be matched, in order to determine what information should be employed in the matching process. Specifically, the strategy selection dynamically regulates the concrete feature selection for linguistic matching, the combination of weights for similarity combination, and the choice of the concrete similarity propagation strategy. After similarity propagation, the matching process concludes with alignment refinement and extraction of the final result.

2.4.6.5 Lily

The core principle of the matching strategy in Lily (name of a girl) is utilizing the useful information effectively and correctly. Lily [P.Wang and B.Xu 2009] aims to provide high quality 1:1 alignments between concept/property pairs. The main specific techniques used by Lily are as follows.

2.4.6.5.1 Semantic Subgraph

An entity in a given ontology has its specific meaning. In the ontology mapping view, capturing such meaning is very important to obtain good alignment results. Therefore, before similarity computation, Lily first describes the meaning for each entity accurately. The solution is inspired by the method proposed by [Faloutsos et al.2004] for discovering connection sub graphs. It is based on electricity analogues to extract a small subgraph that best captures the connections between two nodes of the graph. Another
paper [Ramakrishnan et al.2005] also exploits such idea to find the informative connection subgraphs in RDF graph. The problem of extracting semantic sub graphs has a few differences from Faloutsos’s connection sub graphs. They modified and improved the methods provided by the above two work, and proposed a method for building an \textit{n-size} semantic sub graph for a concept or a property in ontology. The sub graphs can give the precise descriptions of the meanings of the entities, and they call such sub graphs semantic sub graphs. The significance of semantic subgraphs is that it can build more credible matching clues based on them. Therefore it can reduce the negative affection of the matching uncertain.

\textbf{2.4.6.5.2. Generic Ontology Matching Method}

The similarity computation is based on the semantic sub graphs, i.e. all the information used in the similarity computation is come from the semantic sub graphs. Lily combines the text matching and structure matching techniques. Semantic Description Document (SDD) matcher measures the literal similarity between ontologies. A semantic description document of a concept contains the information about class hierarchies, related properties and instances. A semantic description document of a property contains the information about hierarchies, domains, ranges, restrictions and related instances. Finally, all separate similarities are combined with the experiential weights. For the regular ontologies, the SDD matcher can find satisfactory alignments in most cases. To solve the matching problem without rich literal information, a similarity propagation matcher with strong propagation condition (SSP matcher) is presented, and the matching algorithm utilizes the results of literal matching to produce more alignments. Compared with other similarity propagation methods such as similarity flood and SimRank, the advantages of their similarity propagation include defining stronger propagation condition, semantic sub graphs-based
and with efficient and feasible propagation strategies. Using similarity propagation, Lily can find more alignments that cannot be found in the text matching process. However, the similarity propagation is not always perfect. When more alignments are discovered, more incorrect alignments would also be introduced by the similarity propagation. So Lily also uses a strategy to determine when to use the similarity propagation.

2.4.6.5.3. Large Scale Ontology Matching
Large scale ontology matching tasks propose the rough time complexity and space complexity for ontology mapping systems. To solve this problem, they proposed a novel method, which uses the negative anchors and positive anchors to predict the pairs can be passed in the later matching computing. The method is different from other several large scale ontology matching methods, which are all based on ontology segment or modularization.

2.4.6.5.4. Semantic Ontology Matching
Their semantic matching method is based on the idea that Web is a large knowledge base, and from which it can gain the semantic relations between ontologies through Web search engine. Based on lexico-syntactic patterns, this method first obtains a candidate mapping set using search engine. Then the candidate set is refined and corrected with some rules. Finally, ontology mappings are chosen from the candidate mapping set automatically.

2.4.6.5.5. Ontology Mapping Debugging
Lily uses a technique called ontology mapping debugging to improve the alignment results. During debugging, some types of mapping errors, such as redundant and inconsistent mappings, can be detected. Some warnings, including imprecise mappings or abnormal mappings, are also locked by analyzing the features of mapping
result. More importantly, some errors and warnings can be repaired automatically or can be presented to users with revising suggestions.

2.4.6.6 AgreementMaker

AgreementMaker [I.F.Cruz et al. 2009] comprises several matching algorithms or *matchers* that can be used for aligning the source and target ontologies. The matchers are not restricted to any particular domain. The architecture of AgreementMaker relies on a stack of matchers that belong to three different layers (see Figure 2.2). Specific configurations of the stack have been used for the benchmarks, anatomy, and conference tracks, as discussed in what follows. However, it describes first the different components in the stack: the matchers, the combination and evaluation modules, and the final alignment module. The three different Layers shown in Figure 2.2 are described next.

![Figure 2.2 Agreementmaker (Pavel Shvaiko 2010)]

Matchers (First Layer) can be concept-based (if they consider only one concept) or structural (if they consider a subgraph of the ontology). The concept-based matchers support the comparison of
strings. They include: the Base Similarity Matcher (BSM), the Parametric String-based Matcher (PSM) and the Vector-based Multi-Word Matcher (VMM). BSM is a basic string matcher that computes the similarity between concepts by comparing all the strings associated with them. PSM is a more in-depth string matcher, which for the competition is set to use a substring measure and an edit distance measure. VMM compiles a virtual document for every concept of an ontology, transforms the resulting strings into Term Frequency-Inverse Document Frequency (TF-IDF) vectors and then computes their similarity using the cosine similarity measure. These matchers are extended in the AgreementMaker configuration used in the most recent version by plugging in a set of lexicons, which are used to expand the set of strings with synonyms. The extended matchers are therefore called BSMlex, PSMlex, and VMMlex. The Advanced Similarity Matcher (ASM) is a string-based matcher that computes mappings between source and target concepts (including their properties) by comparing their local names, and providing better similarity evaluation in particular when compound terms are used. ASM outperforms generic string-based similarity matchers because it is based on a deeper linguistic analysis.

Structural matchers include the Descendants’ Similarity Inheritance (DSI) matcher. This matcher is based on the idea that if two nodes are similar, then their descendants should be similar. The Group Finder Matcher (GFM) is another structural matcher that filters out the mappings provided by another matcher (the input matcher).

It identifies groups of concepts and properties in the ontologies and assumes that two concepts (or properties) that belong to two groups that are not mapped by the input matcher will likely have different meanings and should not be mapped. The Iterative Instance Structural Matcher (IISM) takes into account instances. Classes that have mapped individuals can then be aligned. In addition, values of
The properties are also considered. The structural part of IISM is quite complex and takes into account superclasses, subclasses, properties, subproperties, cardinalities, and the range and domain of properties. The combination and evaluation modules (Second Layer) are used together, as follows. The Linear Weighted Combination (LWC) combines its inputs (e.g., from several string matchers), using a local confidence quality measure provided by the evaluation module, in-order to automatically assign weights to each result computed by the input matchers. After this step, it has a single combined set of alignments that includes the best alignments from each of the input matchers. The final alignment module (Third Layer) is given as input a mapping cardinality (e.g., 1:1) and a threshold and outputs the best set of alignments given those two inputs.

2.4.6.7 Eff2Match

Eff2Match [Pavel Shvaiko et.al. 2010] takes as input the Uniform Resource Identifier (URI) of a pair of ontologies to be aligned and matches entities (concepts or properties) in the source ontology to those in the target ontology. The alignment process consists of four stages: i) Anchor Generation, ii) Candidates Generation, iii) Anchor Expansion and iv) Iterative Score Boosting as shown in Figure 2.3. In the Anchor Generation stage, matching entities are identified using an exact string matching technique. In the Candidates Generation stage, it enumerates candidates for entities in the source ontology that has not been matched in the previous stage using a Vector Space Model (VSM) approach. In the anchor expansion stage, more equivalent pairs of entities are identified by comparing the source entities with their candidate entities using terminological methods. In the final stage of the matching process, an iterative boosting (Iter-Boost) process is used to identify more pairs of equivalent concepts using the expanded anchor set $A_{exp}$. 
Figure 2.3 Eff2matcher Algorithm Flow.[Pavel Shvaiko et.al. 2010]

2.4.6.8 Nankai Baidu Joint Lab Matcher (NBJLM)

NBJLM procedure is shown in Figure 2.4. It uses a new matching strategy that considers literal similarity measure and ontology structure similarity, simultaneously.[Pavel Shvaiko et.al. 2010] They use the following formula:

\[ \text{Sim}(ID_1, ID_2) = \theta \times \text{Sim literal}(ID_1, ID_2) + (1 - \theta) \times \text{Sim struct}(ID_1, ID_2) \]

where \( \text{Sim literal}(ID_1, ID_2) \) is the literal concept similarity measure, \( \text{Sim struct}(ID_1, ID_2) \) is the structural similarity measure; and \( \theta \) (\( 0 < \theta < 1 \)) is the parameter to control how much literal and ontology structure contributes to the ontologies matching respectively. Firstly, the measure of literal similarity is a preliminary matching. It takes account of polysemy (A polysemy is a word or phrase with different, but related senses. Ex: man: it has different meanings such as i). The human species (i.e., man vs. animal) ii). Males of the human species (i.e., man vs. woman). iii). Adult males of the human
species (i.e., man vs. boy)) and synonym of a word, by transforming the word into a semantic collection using WordNet. Then it can get the preliminary matching results that is semantic mapping rather than spelling mapping of words. Secondly, based on the literal matching results, the measure of ontology structure similarity is calculated through the relation between hypernym (A linguistic term for a word whose meaning includes the meanings of other words. For instance, \textit{flower} is a hypernym of \textit{daisy} and \textit{rose}) and hyponym (In linguistics, a specific term used to designate a member of a class. Ex: \textit{daisy} and \textit{rose} are hyponyms of \textit{flower}) of a word, considering distance of edges, and depth and density of node in the hierarchy of ontology. With the final combination of the two values, and with adjustment of the parameter, it could obtain more reasonable matching results.

![Diagram](image)

**Figure 2.4 Procedure of NBJLM [Pavel Shvaiko et.al.2010]**

### 2.4.6.9 Ontology Mapping by Particle Swarm Optimization (Map PSO)

The development of the Map PSO algorithm has been motivated by the following observations:

i). Ontologies are becoming numerous in number and large in size.

ii). Ontologies evolve gradually.
iii). Ontologies differ in key characteristics that can be exploited in order to compute alignments.

Solving the ontology alignment problem using a PSO-based approach, as done by the MapPSO system, tackles these observations as follows:

i). PSO works inherently parallel, such that large ontologies can be aligned on a parallel computation infrastructure.

ii). PSO works incrementally, which allows the algorithm to start with an initial or partial configuration (i.e. for instance an alignment of previous ontology versions) and refine it as the ontologies evolve.

iii). PSO works as a meta-heuristic, i.e. independently of the objective function to be optimized. In the case of ontology alignment this means that the objective function can be adjusted according the particular alignment scenario at hand.

MapPSO [Pavel Shvaiko et.al. 2010] treats the ontology alignment problem as an optimization problem and solves it by applying a discrete particle swarm optimization (DPSO) algorithm. Particle Swarm Optimization (PSO) is a biologically-inspired, population-based optimization technique that has been successfully applied to various problems in science and engineering. In the context of semantic technologies, optimization problems also occur but have rarely been considered as such. This work addresses the problem of ontology alignment, which is the identification of overlaps in heterogeneous knowledge bases backing semantic applications. To this end, the ontology alignment problem is revisited as an optimization problem. A discrete particle swarm optimization algorithm is designed in order to solve this optimization problem and compute an alignment of two ontologies. A number of characteristics of traditional PSO algorithms are partially relaxed in this article, such as fixed dimensionality of particles. A complex fitness function
based on similarity measures of ontological entities, as well as a tailored particle update procedure are presented. This approach brings several benefits for solving the ontology alignment problem, such as inherent parallelization, anytime behaviour, and flexibility according to the characteristics of particular ontologies. To this end, each particle in the swarm represents a valid candidate alignment, which is updated in an iterative fashion. In each iteration, knowing about the particle representing the best alignment in the swarm, other particles adjust their alignments, influenced by this best particle. A random component when adjusting an alignment makes sure that the swarm does not converge to a local optimum. In MapPSO the quality of an alignment is determined by the average of the qualities of its correspondences, as well as by the number of correspondences in the alignment. Each correspondence is evaluated by a number of base matchers, whose evaluation values are aggregated by a specified aggregator. Base matchers and aggregator can be selected via the params.xml configuration file. This mechanism makes MapPSO highly adjustable, since different alignment scenarios will most likely require different base matchers in order to determine similarity between entities. By following the instructions in the MapPSO documentation one can easily develop base matchers and aggregators tailored to a particular alignment scenario at hand.

2.4.6.10 TaxoMap

*TaxoMap* [Pavel Shvaiko et.al. 2010] is designed to retrieve useful alignments for information integration between different sources. The alignment process is then oriented from ontologies that describe external resources (named *source* ontology) to the ontology (named *target* ontology) of a web portal. The target ontology is supposed to be well-structured whereas *source* ontology can be a flat list of concepts. *TaxoMap* makes the assumption that most semantic
resources are based essentially on classification structures. This assumption is confirmed by large scale ontologies which contain rich lexical information and hierarchical specification without describing specific properties or instances. Then, to find mappings it uses the following available elements: labels of concepts and hierarchical structures. The new implementation of *TaxoMap* introduces a step of refinement of mappings (the alignment results) which extends the alignment process and completes it. *TaxoMap* has been designed to align owl ontologies $O = (C; H)$. $C$ is a set of concepts characterized by a set of labels and $H$ is a subsumption hierarchy which contains a set of ‘is-a’ relationships between nodes corresponding to concepts. The alignment process is an oriented process which tries to connect the concepts of a source ontology $O_S$ to the concepts of a target ontology $O_T$. The correspondences found are equivalence relations (isEq), subsumption relations (isA) and their inverse (isMoreGnl) or proximity relations (isClose). To identify these correspondences, *TaxoMap* implements techniques which exploit the labels of the concepts and the subsumption links that connect the concepts in the hierarchy. The morpho-syntactic analysis tool, TreeTagger, is used to classify the words of the labels of the concepts and to divide them into two classes, full words and complementary words, according to their category and their position in the labels. At first the repartition between full and complementary words is used by a similarity measure that compares the tri-grams of the labels of the concepts and gives more weight to the common full words. Then it is also used by the alignment techniques. Ex: one technique named Label Inclusion generates an isA mapping between cs and ctmax if i) the concept ctmax is the concept of (Target ontology) $O_T$ having the highest similarity value with the concept cs of (Source Ontology) $O_S$, ii) one of the labels of ctmax is included in one of the labels of cs, iii) all the words of the included label of ctmax are classified as full
words by TreeTagger. Given a concept $c_s$ of the ontology source OS, their similarity measure identifies the concept $c_{tmax}$ of the target ontology OT which has the highest similarity with $c_s$. The alignment techniques are then used to decide if the concept $c_s$ can be effectively aligned with this concept $c_{tmax}$ and which relation should be established between the two concepts, or whether, another concept of OT must be chosen. A proposed mapping belongs to a single method, a concept of OS can be aligned at most with one concept of OT. In contrast, the concepts of OT may be involved in several proposed alignments. The main methods used to extract mappings between a concept $c_s$ in OS and a concept $c_t$ in OT are:

Label equivalence: An equivalence relationship, isEq, is generated if the similarity between one label of $c_{tmax}$ and one label of $c_s$ is greater than a threshold (Equiv.threshold).

Label inclusion (and its inverse): If one of the labels of $c_{tmax}$ is included in one of the labels of $c_s$, and if all words of included label are full words, it proposes a subclass relationships $< c_s$ isA $c_{tmax} >$. Inversely, if one of the labels of $c_s$ is included in one of the labels of $c_{tmax}$, it proposes the relationships $< c_s$ isMoreGnl $c_{tmax} >$.

High lexical similarity: If the similarity measure of $c_{tmax}$ is greater than a threshold (HighSim.threshold) and if one of its labels shares at least two full words in common with one of the labels of $c_s$, without being including in the labels of $c_s$, the heuristic generates the relationship $< c_s$ isClose $c_{tmax} >$.

Reasoning on similarity values: Let $c_{tmax}$ and $c_{t2}$ be the two concepts in OT with the highest similarity measure with $c_s$, the relative similarity is the ratio of $c_{t2}$ similarity on similarity $c_{tmax}$. If the relative similarity is lower than a threshold (isA.threshold), one of the two following techniques can be used: the relationship $< c_s$ isClose $c_{tmax} >$ is generated if the similarity of $c_{tmax}$ is greater than a threshold (isClose.thresholdMax). An isA relationship is
generated between cs and the father of ctmax if the similarity of ctmax is greater than a second threshold (isA.thresholdMax).

Best similarity: If none of the above techniques is applicable, the relationship < cs isClose ctmax > is generated if the similarity of ctmax is greater than a threshold (Better.thresholdMax).

Property similarity: Two classes cs and ct are likely to be aligned if they share the same properties.

2.4.6.11 Reasoner for Ontology mapping (OMReasoner)

OMReasoner [Pavel Shvaiko et.al. 2011] includes summarizing algorithm to combine the multiple match results. The combination can be summarized over the n weighted similarity methods. The formula for summarizing is as shown below.

\[ Sim(e_1, e_2) = \sum_{k=1}^{n} w_k Sim_k(e_1, e_2) \quad -(2.1) \]

where \( w_k \) is the weight for a specific method, and \( Sim_k(e_1, e_2) \) is the similarity evaluation by the method.

OMReasoner uses semantic matching methods like WordNet matcher and description logic (DL) reasoning. WordNet1 is an electronic lexical database for English, where various senses (possible meanings of a word or expression) of words are put together into sets of synonyms. Relations between ontology entities can be computed in terms of bindings between WordNet senses. This individual matcher uses an external dictionary: WordNet to achieve semantic correspondences. Another important matcher uses edit distance, which is a measure of the similarity between two words. Based on this value, it calculates the morphology analogous degree by using some math formula. All the results of each individual matcher will be normalized before combination. OMReasoner employs DL reasoner provided by Jena. (Jena is Java API for ontology processing) OMReasoner includes external rules to reason about the
ontology matching. The process of ontology matching in OMReasoner is shown in Figure 2.5.

Figure 2.5 Ontology Matching in OMReasoner [Pavel Shvaiko et.al.2011]

The OMReasoner achieved ontology alignment as the following three steps:

i). Parsing: It can achieve the classes and properties of ontologies by using ontology API: Jena.

48
ii). Combination of multiple individual matchers: the literal correspondences (e.g. equivalence) can be produced by using multiple match algorithms or matchers. Example: string similarity measure (prefix, suffix, edit distance) by string-based, constrained-based techniques. Also, some semantic correspondences can be achieved by using some external dictionary: WordNet. Then the multiple match results can be combined by weighted summarizing method where the Framework of multi-matchers combination is supported, which facilitates inclusion of new individual matchers.

iii). Reasoning: the further semantic correspondences can be deduced by using Description Logic (DL) reasoner, which uses literal correspondences produced in step 2 as input. Finally, it evaluates the results against the reference alignments, and computes two measures: precision and recall.

In OMReasoner, the framework for multi-matchers is flexible, and any new individual matcher can be included. Now, the instances of multi-matchers include EditDistance, Similarity and WordNet as shown in Figure 2.6.

2.4.6.12 Yet Another Matcher Plus Plus (YAM++)

In YAM++ approach, [Pavel Shvaiko et.al. 2011] a generic workflow for a given ontology matching scenario is as follows.

![Figure 2.7 YAM++ Workflow. [Pavel Shvaiko et.al.2011]](image-url)
Figure 2.7 depicts the main components of the YAM++ system. YAM++ discovers mappings between two input ontologies by two matchers: element level matcher and structural level matcher. The combination of the mappings resulting from element level and structural level are then revised by the semantical matcher in order to remove inconsistent mappings.

At the element level, input ontologies are processed in order to extract annotation information for every entity. Based on these information, similarity score between entities are computed by different terminological metrics. Where, similarity metrics can work independently or can be combined by combination methods in order to produce mappings at the element level. Currently, YAM++ supports machine learning based combination methods such as Decision Tree, SVM, NaiveBayes, etc. In that case, the training data are provided by the user or are taken from knowledge base (KB) resources. Otherwise, by default, YAM++ performs their proposed matching method based on information retrieval technique.

At structural level, input ontologies are parsed and transformed into graph data structure. Then, YAM++ takes the mappings resulting from element level as initial mappings to run a similarity propagation process. The propagation algorithm is inspired from the well-known Similarity Flooding algorithm. In semantical checking module, they make use of global constraint optimization method.

2.4.7 Evaluation of recent matching systems

It provides a comparative experimental review of the matching systems [Pavel Shvaiko et.al. 2013] described previously in section 2.4.2. This depicts progress made in the field. It considers here the three oldest test cases of Ontology Alignment Evaluation Initiative (OAEI) in order to have a substantial set of data for comparison as well as diversity in tasks from automatically generated test cases to
expressive ontologies. These are benchmark, web-directories and anatomy. The goal of the benchmark test case is to provide a stable and detailed picture of each matching algorithm. The algorithms are run on systematically generated test cases for this purpose. The domain of this test case is bibliographic references. It aims at comparing an OWL-DL ontology containing more than 80 entities with its variations. Most of the variations are obtained by discarding features of the original ontology. Other variations select either unrelated ontologies or other available ontologies on the same topic.

![Figure 2.8 Evaluation of Recent Matching Systems on Benchmark](image)

[Pavel Shvaiko et.al.2013]

A comparative summary of the best results of OAEI on the benchmarks is shown in Figure 2.8. Edna is a simple edit distance algorithm on labels, which is used as a baseline. In 2004, they maximized the results of the two best systems Fujitsu and PromptDiff. The two best systems of the last several years are ASMOV and Lily. A notable progress has been made between 2004 and 2005 by Falcon; and the results of 2005 are repeated in 2009 by both ASMOV and Lily.
The directory test case aims at providing a challenging task in the domain of large directories constructed from Google, Yahoo and Looksmart web directories. These directories have vague terminology and modeling principles, thus, the matching tasks incorporate the typical uncontrolled open web modeling and terminological errors. The data set is presented as taxonomies where the nodes of the web directories are modeled as classes and the classification relation connecting the nodes is modeled as rdfs:subClassOf. There are more than 4,500 node matching tasks, where each node matching task is composed from the paths to the root of the nodes in the web directories.

Figure 2.9. Results on Web Directories Test Case [Pavel Shvaiko et.al.2013]

A comparison of the results from the years 2006–2010 for the top-3 systems of each year based on the highest F-measure is shown in Figure 2.7.

Key observation is that from 2006 to 2007 it can measure a continuous improvement of the results, while in 2008 and 2009 the
participating systems have either maintained or decreased their F-measure values. The quality of the best F-measure result of 2009 (0.63) achieved by ASMOV is higher than the best F-measure of 2008 (0.49) demonstrated by (Dempster Shafer Similarity) DSSim. It is higher than that of 2006 by Falcon (0.43). It equals to that of Prior+ and is still lower than the best F-measure of 2007 (0.71) by OWL Lite Alignment (OLA)-2.

The focus of the anatomy test case is to confront existing matching technology with expressive ontologies in the biomedical domain. Two of its specificities are the specialized vocabulary of the domain and the usage of OWL modeling capabilities. The ontologies are part of the Open Biomedical Ontologies (OBO) designed from the (National Cancer Institute) NCI Thesaurus (3304 classes) describing the human anatomy, published by the National Cancer Institute and the Adult Mouse Anatomical Dictionary (2744 classes). This test case has been used since 2007, while in 2005 and 2006 it is run on a different test data, which are not considered and focus on the more recent results instead. A comparison of the results from the years 2007–2010 for the top-3 systems of each year based on the highest F-measure is shown in Figure 2.8. It can be made two key observations can be made. The first one is that a baseline label matcher based on string equality already provides quite good results with F-measure of 0.76. The second one is that in all the years the best F-measure remained stable around of 0.86. However, some progress has been made in terms of efficiency, i.e., the run time reduced from days and hours to minutes and seconds. Example: the best runtime result of 15s in 2009 belongs to Anchor-Flood (its F-measure is 0.75). While in 2007 and 2008 the competition is clearly dominated by the Anatomical Ontology Alignment System (AOAS) and SAMBO systems that are heavily exploiting background knowledge (UMLS);
in turn, in 2009 the best result showed by Sobom is obtained without using any background knowledge.

Finally, in 2010 the best result is shown by AgreementMaker. The results of OMReasoner and YAM++ on benchmark are shown in Table 2.1.

<table>
<thead>
<tr>
<th>Method</th>
<th>precision</th>
<th>Recall</th>
<th>f-measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>OMReasoner</td>
<td>0.949</td>
<td>1</td>
<td>0.990</td>
</tr>
<tr>
<td>YAM++</td>
<td>0.98</td>
<td>0.72</td>
<td>0.83</td>
</tr>
</tbody>
</table>

2.5 Outcome of Literature Survey

There are generic matchers such as Falcon,RiMoM, etc as well as domain specific ASMOV,SAMBO etc. Most of the systems find 1:1 alignments. These systems show good accuracy for benchmarks but not for real time data such as web directories. Many systems are not
equipped graphical user interfaces and semantic methods are still rarely employed in matching systems. Hence still more precise and versatile matchers are required. This can be achieved by improving proximity measures and looking at the matching problem in new dimension.

2.6 Summary

This chapter gives insight into the problem domain such as ontology and ontology mapping. First, it discusses the definition of ontology and ontology mapping with an example. Then it discusses the need and application for ontology mapping. It also gives the outline of state of the art research in ontology mapping. The discussion in this chapter has motivated the proposal of new approaches to ontology mapping, discussed in subsequent chapters.