APPROVAL SHEET

Title of Dissertation: A Semantic Resolution Framework for Integrating Manufacturing Capability Data

Name of Candidate: Yan Kang
Doctor of Philosophy, 2015

Dissertation and Abstract Approved: ______________________________
Yun Peng
Professor
Department of Computer Science and
Electrical Engineer

Date Approved: ______________

Curriculum Vitae

Name: Yan Kang.

Degree and date to be conferred: Doctor of Philosophy
Computer Science, May 2015.

Secondary education: No. 1 High School, Chongqing, China, 2015.

Collegiate institutions attended:

University of Maryland Baltimore County, Mater in Computer Science and Electrical Engineer, 2011.

Chongqing Technology and Business University, Bachelor in Computer Science, 2007.

Major: Computer Science.

Professional Publications:

Y. Kang, J. Kim, and Y. Peng, "eXtensible Dynamic Form Approach for Supplier Discovery," *Information Reuse and Integration (IRI), 2011 IEEE International Conference.* June 2011.

Damian D. G. Gessler, B. Bulka, E. Sirin, and Y. Kang, "iPlant Semantic Web Platform uses SSWAP (Simple Semantic Web Architecture Protocol) to enable Semantic Pipelines across Distributed Web and High Performance Computing Resources (Link)," *Semantic Web Challenge,* October 2012.

Y. Kang and Y. Peng, "Ontology-Based Dynamic Forms for Manufacturing Capability Information Collection"*, International Conference on Advances in Production Management Systems (APMS).* September 2013.

Professional positions held:

System Architect (Intern) at LAPPLS, LLC
Baltimore, MD, USA, 2014 – Present.

Software Developer (Intern) at Clark & Parsia LLC
Washington, D.C., USA 2012 – 2012.

Abstract

Title of Dissertation: A Semantic Resolution Framework for Integrating Manufacturing Service Capability Data

Yan Kang, Ph.D. Computer Science, 2015

Directed By: Yun Peng, Professor
Department of Computer Science and Electrical Engineering

Building flexible manufacturing supply chains requires availability of interoperable and accurate manufacturing service capability (MSC) information of all supply chain participants. Today, MSC information, which is typically published either on the supplier's web site or registered at an e-marketplace portal, has been shown to fall short of interoperability and accuracy requirements. The issue of interoperability can be addressed by annotating the MSC information using shared ontologies. However, this ontology-based approach faces three main challenges: (1) lack of an effective way to automatically extract a large volume of MSC instance data hidden in the web sites of manufacturers that need to be annotated; (2) difficulties in accurately identifying semantics of these extracted data and resolving semantic heterogeneities among individual sources of these data while integrating them under shared formal ontologies; (3) difficulties in the adoption of ontology-based approaches by the supply chain managers and users because of their unfamiliarity with the syntax and semantics of formal ontology languages such as web ontology language (OWL).

The objective of our research is to address the main challenges of ontology-based approaches by developing an innovative approach that is able to extract MSC instances from a broad range of manufacturing web sites that may present MSC instances in various ways, accurately annotate MSC instances with formal defined semantics on a large scale, and integrate these annotated MSC instances into formal manufacturing domain ontologies to facilitate the formation of supply chains of manufacturers. To achieve this objective, we propose a *semantic resolution framework* (SRF) that consists of three main components: a MSC instance extractor, a MSC Instance annotator and a semantic resolution knowledge base. The instance extractor builds a local semantic model that we call instance description model (IDM) for each target manufacturer web site. The innovative aspect of the IDM is that it captures the intended structure of the target web site and associates each extracted MSC instance with a context that describes possible semantics of that instance. The instance annotator starts the semantic resolution by identifying the most appropriate class from a (or a set of) manufacturing domain ontology (or ontologies) (MDO) to annotate each instance based on the mappings established between the context of that instance and the vocabularies (i.e., classes and properties) defined in the MDO. The primary goal of the semantic resolution knowledge base (SR-KB) is to resolve semantic heterogeneity that may occur in the instance annotation process and thus improve the accuracy of the annotated MSC instances. The experimental results demonstrate that the instance extractor and the instance annotator can effectively discover and annotate MSC instances while the SR-KB is able to improve both relaxed precision and relaxed recall of annotated instances and reducing human involvement along with the evolution of the knowledge base.

Key words: *manufacturing supply-chain, interoperability, ontology, data extraction, semantic annotation, semantic similarity, semantic resolution*

A SEMANTIC RESOLUTION FRAMEWORK FOR INTEGRATING MANUFACTURING SERVICE CAPABILITY DATA

By

Yan Kang

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, Baltimore County, in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2015

Acknowledgements

I am thankful for the generous support provided by my advisor, Yun Peng as well as the members of my doctoral committee: Tim Finin, Yaacov Yesha, Matt Schmill, Charles Nicholas and Boonserm Kulvatunyou. I am grateful to Yichuan Gui, Lushan Han, Yunsu Lee, Jaewook Kim and all past and present members of Ebiquity Lab for their generous support in my research and study. Special thanks to my dear friends: Congcong Liu, Zhong Ren, Yi Xin, Sang Ni, Chong Zhang, Lin Cao, Yueyue Song and Zhicuan Huang for their consistent help and support. Finally, thank you Mom and Dad for your loving support and patience through all of the hard work.

Table of Contents

List of Figures

List of Tables

Chapter 1.

Introduction

This dissertation studies the problem of integrating manufacturing service capability (MSC) instance[1] data from different sources into a formally defined manufacturing domain ontology for interoperability among different manufacturers. We propose semantic resolution framework (SRF) that is able to extract a large amount of MSC instances and relations of these instances from web sites of manufacturers, and accurately annotate these instances with vocabulary (i.e., classes and properties) defined in a manufacturing domain ontology. This framework also provides an error-correction mechanism to correct misannotated instances. The level of automation of the correction process increases along with the growth and evolution of the underlying knowledge-base as more websites are annotated.

1.1 Motivation

With the rapid pace of economic globalization, the competitiveness of manufacturers is increasingly defined by the *combined* capabilities of the suppliers that make up the manufacturer's supply chains [1]. Quick formation and optimization of global supply chains requires that the manufacturing service capability (MSC) information of each partner be accessible, understandable, and processable by all others in the chain in an automated way. In other words, such information needs to be *machine-understandable* and *semantically interoperable*. To achieve *machine-understandable*, MSC information of individual suppliers should be accurately annotated with formal semantics and without ambiguity. *Interoperability* deals with the heterogeneity of the source data from suppliers in their structures and vocabularies, and it requires that the

[1] In the rest of this dissertation, whenever we mention instances, we mean manufacturing service capability instances.

MSC information collected from diverse sources be represented in a way that the meanings of data items and their interrelations can be understood correctly by all stakeholders (and their machines). In other words, it requires interoperability at the *semantic level* [2, 3].

Today, MSC information is typically either registered at an e-marketplace portal or published on a supplier's web site. Commercial e-marketplace portals, such as mfg.com[2], thomasnet.com[3], globalspec.com[4], require the users to enter their MSC information in a uniform format according to the portal's proprietary capability information models. This gives potential for interoperability to users within a portal, but not between portals. For ease of use, most e-marketplaces use relatively simple capability information models. These simple models are typically only able to model a small portion of the whole manufacturing domain and not expressive enough to represent the intended meaning of the user's capability information. Compared with entries at e-marketplace portals, websites of suppliers are resources more suitable for acquiring manufacturing capability information. For one thing, these websites are freely accessible. For another, information published on webpages is typically much richer and detailed than that registered at e-marketplace portals. However, webpages are primarily for visualizing information for human consumption. MSC information published on web pages is typically not understandable by machines. Information extraction and natural language techniques may help extract machine-understandable information from web pages. However, they do not offer a solution to integrate extracted information from diverse sources.

The advances in semantic web technologies, including its canonical data model RDF[5] and logic-based web ontology language OWL[6], have led to extensive research in the development of ontology-based approaches to integrating information from heterogeneous sources for semantic interoperability. These traditional ontology-based information integration approaches typically require individual sources to develop their own local ontologies or reuse well-known existing ontologies (e.g., Dublin Core,

[2] http://www.mfg.com/
[3] http://www.thomasnet.com/
[4] http://www.globalspec.com/
[5] http://www.w3.org/RDF/
[6] http://www.w3.org/TR/owl-ref/

SIOC and GoodRelation[7]) to annotate their legacy data and store these annotated data in the canonical RDF data model. Then, individual RDF datasets are integrated under a common ontology that all local ontologies are mapped onto. Such a requirement imposes severe barriers to the adoption of these ontology-based information integration by the business world in that (1) the syntax and semantics of formal ontology languages such as OWL and RDF are not familiar to the users (i.e., suppliers or manufacturers), causing a steep learning curve; (2) as a newly emerging technique, tools friendly for inexperienced users are not yet sufficiently mature and widely available; and (3) there is no clear tangible business incentive for suppliers to invest resources into making their data publicly accessible and compatible with the semantic web of the future [4, 5, 6].

To address those problems and others, we propose an innovative approach to integrate MSC information. Similar to traditional ontology-based information integration approaches, our approach integrates MSC information of different (heterogeneous) sources based on a common ontology. However, the important difference is that our approach builds local semantic models for web sites of individual suppliers (i.e, manufacturers). These local semantic models capture the structure and other semantic relations of MSC information published on these web sites. Based on these local semantic models, our approach automatically annotates extracted MSC instances rather than ask suppliers to open and annotate their data. We also offer a semantic resolution knowledge base (SR-KB) that iteratively improves the quality of annotated MSC instances by correcting errors that may occur in the annotation process. It evolves itself along with the accumulation of the validated MSC instances during the MSC instance integration process. This evolution of SR-KB helps improving the accuracy of integrated MSC information while at the same time reduces human intervention.

[7] http://www.heppnetz.de/projects/goodrelations/

1.2 Dissertation Statement and Contributions

In this dissertation, we present a semantic resolution framework (SRF) that aims at efficiently *extracting* a large volume of manufacturing service capability instances from the web, accurately *annotating* these instances with semantics and *integrating* these instances under a formally defined manufacturing domain ontology as a manufacturing knowledge base. The development of the semantic resolution framework makes several important contributions to the field of ontology-based information integration.

Our SRF offers an *instance extractor* that is able to deal with diverse ways in which MSC instances are presented in web pages and a local semantic model called *instance description model* (IDM) that organizes MSC instances extracted from web sites in a structural way. Having this local semantic model at hand, we can leverage ontology mapping techniques to establish mappings between components of IDM and those of MDO. Then, the instance integration is performed based on these mappings rather than directly applying MDO to web pages, which is neither efficient nor accurate.

When integrating instances from the distributed and diverse web, it is unavoidable to make mistakes in annotating some of the MSC instances because of semantic heterogeneities [7, 8, 9, 10]. Resolving domain-specific semantic heterogeneities, thereby more accurately annotating instances, require domain expertise. However, domain experts are not necessarily familiar with OWL-based ontologies. Tools that can assist users with little knowledge on ontology in examining the correctness of annotated instances are needed. To this end, we developed a tool called *Instance Validation Platform* (IVP) that enables domain experts to examine all the annotated instances through a user-friendly interface and make corrections if necessary. Putting human in the loop of validating the correctness of annotated instances is tedious work. To address this issue, we developed a *manufacturing concepts mapping repository* and a *Naïve Bayes-based annotation corrector* that are able to gradually reduce human interventions while improving the accuracy of annotated instances.

The manufacturing concept mapping repository (MCMR) helps the instance annotator more accurately map contexts of instances to classes defined in the manufacturing domain ontology by resolving terminological heterogeneity, thereby improving the accuracy of annotated instances. The Naïve Bayes-based annotation corrector (NBAC), on the other hand, aims to correct misannotated instances (i.e., assigned with wrong or inaccurate class labels) mainly by resolving conceptual heterogeneity. It is trained by annotated instances validated by domain experts with the assistance of IVP. The MCMR, NBAC and IVP together form the semantic resolution knowledge base (SR-KB) that is a core component of our semantic resolution framework. The experimental result demonstrates that the SR-KB not only helps the instance annotator improve the accuracy of annotated instances but also gradually reduces human intervention during the instance annotation process.

Practically, this framework can be adopted by commercial e-marketplace portals to improve the structure of manufacturing data registered by manufacturers and thus increase the accuracy of the customer-manufacturer matching results. It can also be applied independently to extract manufacturing service capability data from manufacturing web sites. Extracted manufacturing data can be published to the Linked Open Data (LOD). To the best of our knowledge, no single data set related to manufacturing domain is available in the LOD cloud. Therefore, publishing manufacturing service capability data to LOD cloud would be a great contribution to both manufacturing and Linked Data community.

1.3 Dissertation Outline

The rest of the dissertation is organized as follows. Chapter 2 describes background and related work. Chapter 3 provides an overview of the proposed semantic resolution framework. Chapter 4 defines the similarity measures and mapping functions that will be used in this dissertation. From Chapter 5 to Chapter 7, we discuss our proposed approaches in detail. Specifically, Chapter 5 explains how the instance description model (IDM) is constructed. Chapter 6 elaborates how instances and relations of the IDM are automatically annotated with classes and properties defined in the manufacturing domain ontology. Chapter 7 presents the semantic resolution knowledge base (SR-KB). The experimental results that demonstrate the effectiveness of the SR-KB in improving the overall performance of the instance annotator are also presented in Chapter 7. Finally, Chapter 8 concludes and gives directions for future research. To help the reader, a list of abbreviations is provided at the end of the dissertation.

Chapter 2.

Background and Related Work

In this chapter, we introduce the Semantic Web technologies that we adopt as the basis for our semantic resolution framework and survey related research. The first section provides a high level description of the core components of Semantic Web technologies. The second section surveys research closely related to our work of developing the semantic resolution framework.

2.1 Background

In this section we first introduce the basic concepts of the Semantic Web and then the core components that compose Semantic Web technologies. The Linked Data, a pragmatic approach that aims at achieving the vision of Semantic Web, is briefly described at the end of this section.

2.1.1 Semantic Web

The broad vision of Semantic Web is to extend current web of documents and create additional layer of data above it. This web of data is not merely about data. It is about linked data that have rich semantics such that it can be easily explored and consumed by both human and machines. A more ambitious vision of Semantic Web is that it would facilitate integration of data from heterogeneous sources such that intelligent applications or systems can be developed to perform tasks that can only be done with human intelligence.

A major shift from the traditional relational databases to Semantic Web is the mindset of modeling the world. Compared to Closed World Assumption (CWA) taken by traditional relational databases, Semantic Web adopts the Open World Assumption (OWA) [15, 16] that possesses the following main points:

- Lack of a given fact does not imply it to be false. It is simply not known.
- Everything is permitted until it is prohibited.
- Ontologies can be developed and matured incrementally; later versions are an enhancement and not a replacement for prior ontologies.
- Information at various levels is incomplete and only partially known.

This OWA suits well the heterogeneous, distributed and ever-increasing nature of the web of data. To facilitate the adoption of OWA and the construction of Semantic Web, the development of the pieces comprising the Semantic Web layered architecture has been continuing since the original *Scientific American* article on Semantic Web appeared in 2001 [17]. Figure 2.1 shows the Semantic Web Layer Cake.

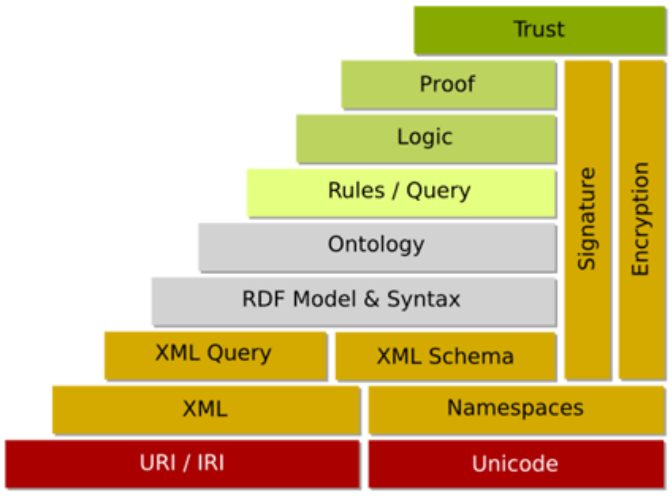


Figure 2.1 Semantic web layer cake

In the next two subsections, we will introduce the core components of Semantic Web, including RDF, RDFS, OWL and Ontology. Linked data, a pragmatic approach to achieve the Semantic Web, will be introduced in section 2.1.4.

2.1.2 RDF, RDFS, OWL

RDF (Resource Description Framework) serves as the foundation of Semantic Web and is essentially a data model that provides an abstract and conceptual framework for defining and using both data and metadata. It represents both data and metadata in the form of triples, which relate data entities in a *subject-predicate-object* pattern. *Subject* is the thing we are talking about; *predicates* and their corresponding *objects* describe features of this thing. One major benefit of this triple representation is the ease to aggregate and integrate heterogeneous data. This is because a triple in RDF is essentially a directed graph with the subject and object as vertices and the predicate as the edge between them. Hence, any number of triples can be easily added up to form another directed graph.

Although RDF can describe basic facts about things (e.g., one thing is a type of another thing), it has very limited expressive power and thus provides limited support for logical reasoning. For example, it has no range or domain constraints and no ability to organize things in taxonomical structure. RDFS (RDF Schema) enriches RDF data model with more expressive power by adding the *subclass* and *subproperty* declarations and adding *domain* and *range* to properties. OWL (Web Ontology Language) is a combination of RDF data model and a form of description logic [18]. The primary advantage of RDF is that it can easily describe things in an "entity-relation" like schema using "subject-predicate-object" triples and thus it facilitates the information integration of heterogeneous sources. The description logic and its well-defined formal semantics allow one to perform logical reasoning over the ontology and its instances. OWL thus provides a formal framework with sufficient constructs for modeling data of complex relations, describing that data through controlled vocabularies, and interoperating that data through logical reasoning.

2.1.3 Ontology

An ontology can be seen as a framework for representing knowledge of a certain domain. It defines concepts and relationships between these concepts in the domain of interest, and encodes them in a machine-understandable form to make it available to information systems. Many versions of definitions of what an ontology is exist, the most dominating one defines an ontology as: "*a formal, explicit specification of a shared conceptualization*" [19]. The "formal, explicit specification of a conceptualization" means that a conceptualization that is typically in the mind of people should be modeled accurately and clearly in a way that it can be understood by machines or information systems. The word "shared" indicates that in order for an ontology to support a certain degree of interoperability, ontology stakeholders should reach certain agreements on concepts and/or relationships used in the ontology, i.e., they are sharing a common conceptualization of the domain.

Within the Semantic Web community, ontologies are typically written in OWL (Web Ontology Language), or its various species (e.g., OWL DL/Full/Light, OWL RL and OWL QL). The graphic nature of RDF combined with the expressive power and reasoning ability provided by OWL offer ontologies significant advantages over traditional schema such as XML schema, relational database schema and taxonomies. Here, we list its most appealing advantages. There are likely many others [20, 21, 22].

- **Flexible Representation of Data:** The graph-based data model of OWL (inherited from RDF) is able to represent data of various types, including unstructured (e.g., free text), semi-structured (i.e, XML or web pages) and structured (i.e., traditional relational database). This characteristic is crucial in transforming and integrating data from different (heterogeneous) sources.
- **Facilitate Information Integration**: instance data from difference sources can be easily linked together by using properties of OWL such as *owl*:*sameAs*. Classes and properties from different (heterogeneous) ontologies can be directly mapped by using OWL built-in properties (e.g., *owl*:*subClassOf*, *owl*:*subPropertyOf* and *owl*:*equivalentClass*), logical axioms or properties from other upper ontologies that aim at integrating ontologies.
- **Logic-based Reasoning**: An ontology expressed in a knowledge representation language based on the ground of formal logic such as OWL enables an information system to derive implicit knowledge from explicit knowledge and facts by means of logical reasoning. Moreover, the reasoning can be performed across heterogeneous ontologies, if certain mappings have been established between these ontologies. For example, if a Service class from one ontology is mapped to a Capability class from the other by using *owl*:*equivalentClass* property of OWL, we can infer that instances of the Service class are also instances of the Capability class.

These and other advantages of OWL-based ontology make it very attractive for collecting and integrating information from different sources. However, its terminology and underlying inference mechanism, especially those from description logic, may be unfamiliar to many non-technical participants; the conceptualization (classes and inter-class relations) expressed in an OWL ontology are likely to be different from the one that users have in their mind.

2.1.4 Linked Data

Linked Data is coined by Tim Berners-Lee [23] as a pragmatic approach to accomplish the vision of Semantic Web where data is semantically annotated and connected based on shared ontologies. To help reach such vision, Linked Data approach offers a set of best practices and technological principles for publishing and connecting machine-processable data on the web open to use by governments, organizations, businesses and individuals.

One of the challenges facing Linked Data is how to transform unstructured data from various sources into semantic-rich linked data. Linked Open Data[8] is a project that aims at addressing this challenge by identifying existing data sets available under open licenses, converting these data to RDF and linking these data according to the Linked Data principles, and finally publishing them on the web. Founded in 2007, LOD has been gaining great popularity these years. Its nucleus dataset, DBpedia[9], alone consists of approximately 3 billion RDF triples. Currently, LOD covers 295 data sets and more than 31 billion RDF triples[10] from different domains such as Media, Geographic, Government, Economy, Energy, Life sciences, Commerce, etc. The three biggest producers and consumers of LOD are Geographic, Life Science and Government. Figure 2.2 depicts the current view of Linked Open Data cloud. It is believed that more and more industries will perceive LOD as a cost-efficient way to integrate and consume data.

[8] http://linkeddata.org/
[9] http://dbpedia.org/About
[10] http://lod-cloud.net/state/

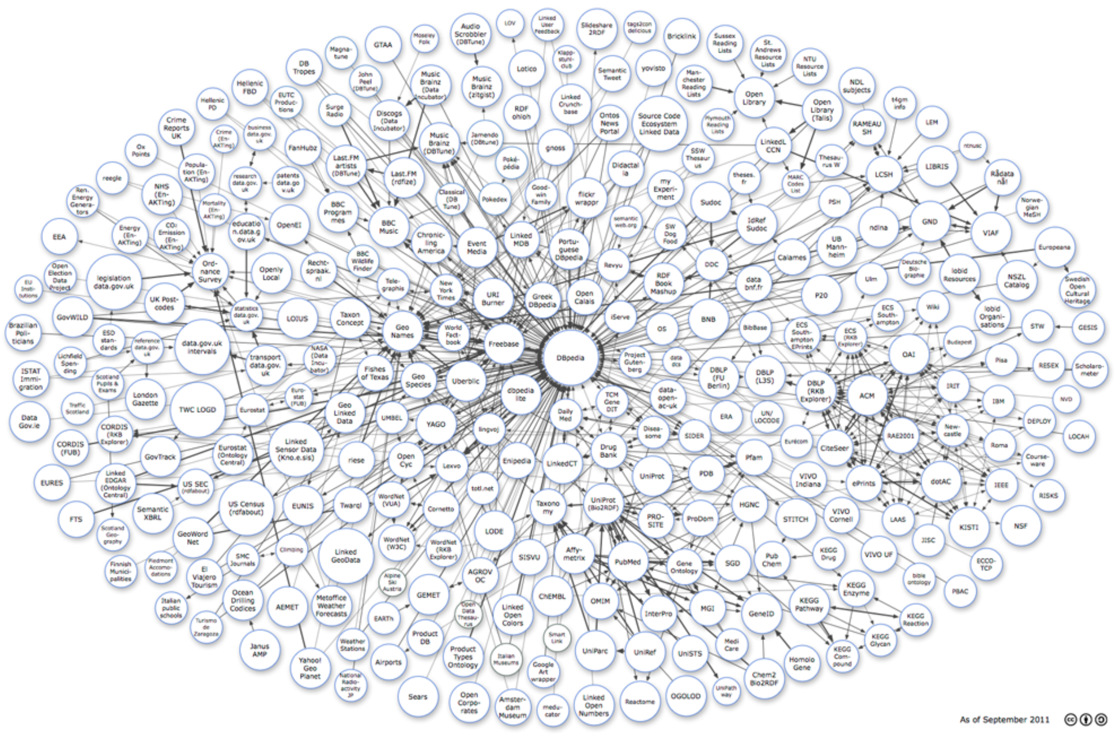


Figure 2.2 Linked Open Data cloud

LOD has achieved significant success in noncommercial sectors, however, its uptake in commercial sectors is slow. Particularly, to the best of our knowledge, no single data set related to manufacturing domain is available in the LOD cloud. Therefore, publishing linked data sets of manufacturing domain to LOD cloud would be a great contribution to both manufacturing and Linked Data community. The manufacturing knowledge base, the output of our semantic resolution framework, could serve as the basis for that purpose.

2.2 Related Work

This section briefly surveys existing research related to ontology-based information integration. We first introduce the three ontology-based Information Integration paradigms, their advantages and disadvantages as well as the challenges they are facing. Then, we present state of the art Information Extraction (IE) methods and systems in two subsections, for traditional IE and ontology-based IE, respectively. We include traditional IE for the sake of completeness. The ontology-based IE, on the other hand, deals with extracting ontology instances and is thus closely related to Ontology Population, the topic of this dissertation. Finally, the related works on the ontology mapping are reviewed in the last subsection.

2.2.1 Ontology-Based Information Integration

Ontology-based Information Integration can be categorized from different perspectives using different terms. By summarizing several discussions [24, 25, 26] on the topic of ontology-based Information Integration, we group the ontology-based Information Integration approaches into three categories: *bottom-up*, *top-down and hybrid*. Figure 2.3 depicts the high level view of the three categories.

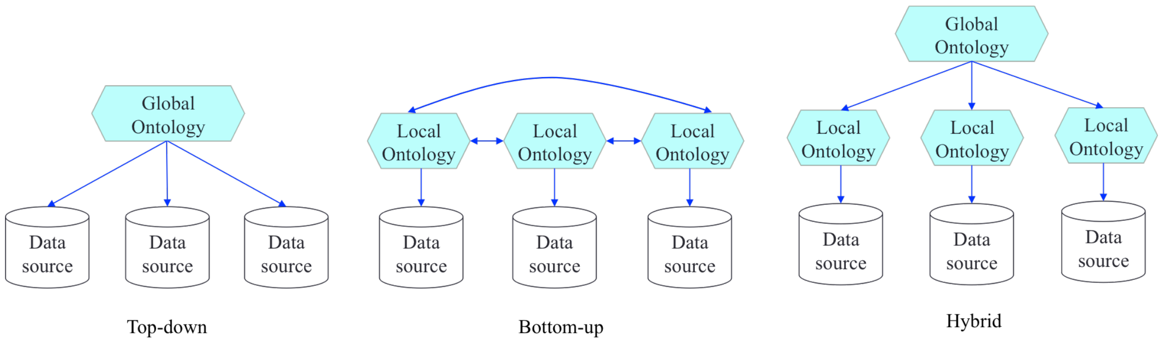


Figure 2.3 High level view of three ontology-based information integration categories

Top-down Approach. The top-down approach requires all individual data sources refer to one global ontology to describe their data. This approach can be applied to information integration system where all data sources to be integrated share nearly the same view on a domain and represent their views in the same way. In such a system, semantic interoperability is easy to achieve since all participants share the same view. However, for domains that enjoy vast diversity, developing one global ontology that is agreed upon by all individual data sources is extremely difficult. For this reason, the top-down approach is considered impractical in communities where it is hard to achieve common ground.

Bottom-up Approach. In bottom-up approach, individual data source is annotated by its own ontology. Then, these annotated data of individual (heterogeneous) sources can be combined or communicate with each other through pair-wise mappings established among these data sources. The advantages of the bottom-up approach include that the description of a data source by using the local ontology can achieve a high degree of accuracy since the local ontology is tailored to describe its own data, the changes made on individual ontology would not affect other ontologies and no commitment on a global ontology is needed. However, the lack of a global ontology makes it extremely difficult for individual sources to cooperate with each other. For example if certain semantic interoperability needs to be achieved among multiple data sources, pair-wise schema-level mappings need to be established among these sources. The pair-wise mappings would grow dramatically as the number of data sources increases. Moreover, as pointed out by Wache et al. [25, 27] and others, in practice the pair-wise mappings are very difficult to define and maintain, because of many semantic heterogeneity problems that need to be dealt with. Although, the new endeavor of Linked Open Data (LOD) initiative, lowers the technical barriers of adopting Semantic Web techniques to aggregate data, little effort was seen in developing and populating LOD dataset for the manufacturing domain or its subdomains.

Hybrid Approach. The hybrid approach strikes a balance between the top-down approach and the bottom-up approach. For one thing, as opposed to top-down approach, individual sources use their local ontologies to describe data. These local ontologies are typically expressed in the same ontology language (e.g., OWL) as the global ontology is. For another, as opposed to bottom-up approach, the hybrid approach achieves the semantic interoperability by establishing mappings from local ontologies to a global ontology. Thus, instead of establishing and maintaining $O(C_2^n)$ set of mappings, as bottom-up approach does, the hybrid approach only need to maintain O(n) sets of mappings. The global ontology defines generic concepts and relationships to describe the domain of interest and provides a unifying conceptual view on that domain. This enables the formulation of high-level queries over the entire domain through one interface without knowledge about the heterogeneous back-end data sources.

Despite all these benefits, there is one critical challenge that prevents the hybrid approach from being adopted by domains such as manufacturing. That is, in order to achieve semantic interoperability, the proprietary data models (e.g., rational databases and XML schemas) of different sources should be expressed in the same ontology language and mapped to the global ontology. Mapping each of them to the global ontology is not only costly but also not scalable in the web since there are simply too many websites that are represented in different modeling languages and have different degree of formalization for their different conceptualizations. From a more technical perspective, the hybrid approach is difficult to be adopted by business entities because (1) the syntax and semantics of the formal ontology languages such as OWL and the state of art Semantic Web techniques are not familiar to the users and thus they would have a steep learning curve to model their legacy data using these techniques and (2) no sufficiently mature and widely available tools for inexperienced users to embrace these newly emerging techniques.

2.2.2 Information Extraction

In this section, we briefly review some researches on Information Extraction (IE). We include this section because the ontology-based information extraction is a subfield of IE and it adopts many techniques from the field of IE. For more thorough review of IE, we refer the reader to [28, 29, 30].

According to Russell and Norvig [31], Information Extraction is the process of (semi-) automatically extracting certain types of structured information (e.g., a particular class of objects or events and occurrences of relationships among them) from unstructured documents (e.g., free text) or semi-structure documents (e.g., xml document and web pages). Here, we present some IE methods/systems by grouping them into categories based on the two types of input sources, namely the free text and the web-pages. Surveys of IE methods from other perspectives can be found in [30, 29].

Extract data from free text. When dealing with free text, IE typically employs natural language processing techniques such as part-of-speech taggers, lexical semantic interpretation, name-entity recognizers and morphological analyzer [32, 33].

Based on these techniques, linguistic rules or extraction patterns can be built by human experts or learned by machine learning (ML) algorithms. Manually created linguistic rules typically harness human intuition and domain knowledge, and thus no training data are needed. In addition, manually created linguistic rules can achieve high degree of accuracy since they are tailored to the specific problem. Building Finder [34] is a domain specific system aimed to retrieve and integrate information about streets and buildings mainly from texts of heterogeneous sources and present them in satellite images. Its record-linkage system applies manually created mapping rules to various documents, including free-text documents, of this specific domain to extract relevant data. Amilcare [35] is an adaptive IE system that uses ML algorithms to learn to adapt to new domains by using a set of annotated texts. More specifically, Amilcare first creates training data by annotating text using ANNIE [36], GATE's shallow IE system. Then, it applies supervised ML algorithm [37] to induce rules from the training data. These induced rules are applied to the text of the domain to extract more instance data.

Manually created rules require both language process skills and domain knowledge from human experts, and are perceived as expensive to create. ML-based rules, on the other hand, can be (semi-) automatically learned from some training data. Agichtein [38] proposed an approach that first creates extraction patterns by training an information extraction system called *snowball* with only a small set of seed instances. Then these extraction patterns combined with name-entity tags are used to identify new instances. Shinyama [39] applies an unrestricted relation discovery technique to discover all relations from a large corpus of new articles. In the process of unrestricted relation discovery, all articles are first clustered based on their topics. Then each article is name-entity tagged and each sentence of these articles are transformed into a GLARF structure (Grammatical and Logical Argument Representation Framework) [40]. A collection of extraction patterns, called basic patterns, is created from these GLARFized articles. Finally, these basic patterns are applied to extract relations. S-CREAM (Semiautomatic CREAtion of Metada) [41] is a system that automatically annotates texts given a set of training data. This system provides users with two tools, Onto-O-Mat and Amilcare, to create domain-specific training data. With the assistance of the two tools, S-CREAM is trainable for different domains.

Extraction rules or patterns targeting at free text are typically learned from supervised machining learning algorithm since free text is lack of structure that is understandable by machines. For each specific task, a new set of training data is required. Thus, these supervised approaches are not robust to changes of domain. Information extraction systems dedicated to extract data from semi-structured web pages, however, can achieve higher degree of automation and robustness, since they can exploit existing structure of web pages, especially web pages with templates.

Extract data from web pages. Earlier web information extraction systems are designed to facilitate programmers in writing extraction rules. Later systems utilize supervised learning methods to automatically generate such rules. As discussed earlier, supervised methods need human-labeled training examples to train the extractor and thus are expensive in terms of cost for human expertise and manual labeling process. Recently, many efforts have been devoted to creating systems with unlabeled training examples, typically web pages represented by HTML DOM (Document Object Model). These systems are called semi-supervised or unsupervised information extraction systems since human labeling may not be required. IEPAD [42] is one of the first IE systems that use semi-supervised methods to generate extraction rules. IEPAD works under the observation that multiple homogeneous data records are normally rendered regularly using some templates for good visualization. Thus, patterns can be discovered by exploiting such regularity. Since human interventions are still required to ease uncertainties in found patterns, IEPAD is considered a semi-supervised system. RoadRunner [43] is an unsupervised IE system that does not use any labeled training examples and has no human interventions. It first clusters web pages with similar structure and then compares web pages from the same cluster to identify possible templates or patterns. Based on these similarities and differences, extraction patterns then can be induced. Finally, relevant data can be extracted by applying these patterns. Other than web IE approaches that target at web pages, approaches dedicated to extract data from specific HTML data structures (e.g., table, list) have also been proposed [44, 45]. [46] reports one of the most recent efforts to extract linked data from tables.

A common shortcoming of most traditional information extraction approaches is that extracted data are not organized according to a shared formal ontology. Thus, data extracted from multiple sources cannot be queried as a whole, since they lack semantic interoperability. Ontology-based Information Extraction (OBIE) approaches are proposed to extract information from text documents (both free text and web pages) guided by ontologies and represent results using ontologies. Thus, the extracted data from heterogeneous sources are organized under a common ground such that they can be consumed by users using high-level queries without the knowledge about the back-end sources. This characteristic, and many others, makes OBIE useful in realizing the vision of the Semantic Web.

2.2.3 Ontology-based Information Extraction

Ontology-based Information Extraction (OBIE) is a subfield of information extraction [47]. It typically utilizes an ontology or a set of ontologies to guide the information extraction process. Some OBIE systems [48, 49] output a set of instances of classes and properties defined in existing ontologies. Others [50, 34] may output a set of new classes or/and properties as an extension to the existing ontologies. Figure 2.4 depicts a typical view of OBIE architecture.

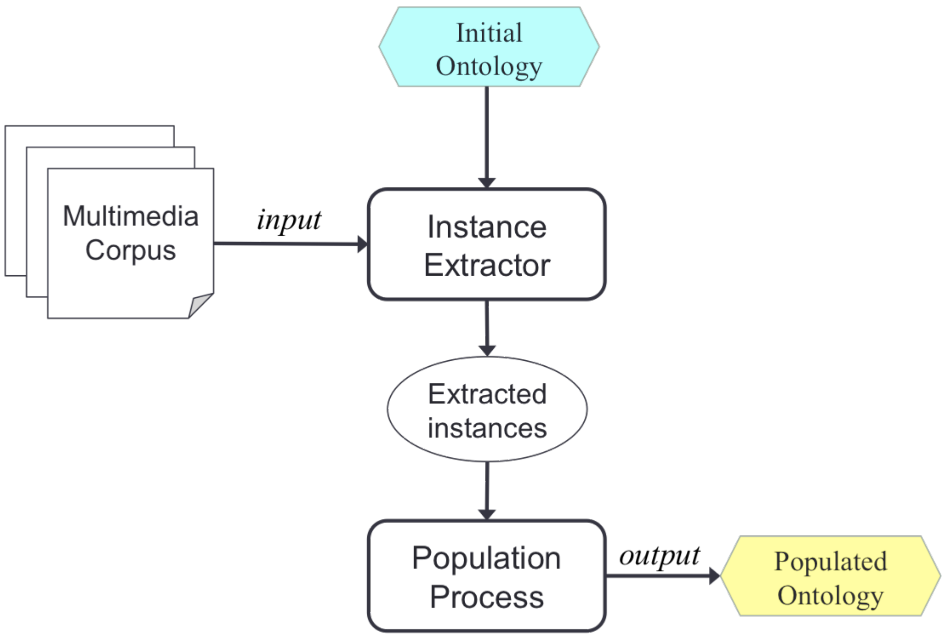


Figure 2.4 Ontology-based Information Extraction architecture

As described in [25, 47], OBIE has many advantages over traditional IE. Following highlights some of these advantages [51, 52, 53, 54]:

1) **Resolving semantic heterogeneity among proprietary data models**. The ontology (or a collection of ontologies) adopted by an OBIE system is typically an upper domain ontology that defines generic concepts and relations of the domain of interest. The semantic heterogeneities among proprietary data models can be resolved to certain degree by establishing schema-level mappings from these proprietary data models to this upper domain ontology. These mappings are typically considered as a part of the OBIE system [55].

2) **Creating structured data for the Semantic Web**. The success of Semantic Web relies on the existence of a large volume of structured data that can be processed and interpreted by computer programs. OBIE provides an automatic (or semi-automatic) mechanism to generate these structured data and store them as ontology instances. This process is known as *semantic annotation* [52, 56, 57].

3) Improving the quality of ontologies and enriching ontologies. If a domain ontology can successfully guide an OBIE system to extract data related to that domain, this ontology can be seen as a good representation of this domain. If a domain ontology fails to fulfill such task, the shortcomings of this ontology can be identified by analyzing the types of semantic content it has failed to extract. Further, the improvement and enrichment on this ontology can be achieved based on the analysis [53, 54].

The difference between OBIE and traditional IE mostly lies in the fact that the information extraction process in OBIE is guided by ontologies and the results are represented using ontologies. Except this fact, OBIE is almost identical to traditional IE. In other words, no new information extraction methods are invented in OBIE but the existing methods are applied to identify instances of an ontology. We have reviewed some systems on the traditional IE earlier. We now present some systems/tools that utilize OBIE to extract information:

SOBA [58] is an ontology-based information extraction and integration system that integrates information on football match events from web pages and focuses on HTML tables. The manually created mapping rules are leveraged to transform HTML tables to an XML representation that is then used to update the SWintO ontology [59], which integrates a number of domain and task ontologies. MUSING [48] constitutes a query answering system by adopting OBIE approach. It semi-automatically extracts information from text documents using linguistic rules and gazetteer lists guided by a manually defined ontology. The incorrect extractions made by this process can be corrected by domain experts. BOEMIE [60] extracts, fuses and interprets information from heterogeneous sources including texts, images and videos published in web pages. BOEMIE adopted a synergistic approach that on the one hand it automatically extract and semantically mark-up information from multimedia content guided by one or more ontologies in order to populate and enrich these ontologies and, on the other hand, it deploys enriched ontologies to enhance the robustness of the extraction system. PANKOW [52] automatically categories instance data from text with respect to a given ontology by exploiting extraction patterns and the redundancy on the web. It constructs the patterns by first mapping each instance datum extracted from the text to one of the classes defined in their tourism ontology. Each pair of the instance and class is then checked against the web via Google queries. The number of hits is used as the measure of the degree of correctness of the mapping. Mappings with large hit number will be transformed as extraction patterns.

2.2.4 Ontology Mapping

Ontology mapping is aimed at resolving semantic heterogeneities among ontologies of heterogeneous data sources. For this reason, establishing mappings between ontologies is a crucial task in any Ontology-based Information Integration (OBIE) system. In this section, we first discuss various forms of semantic heterogeneities and then briefly survey basic techniques used to establish mappings between ontologies. Most of the materials covered in this section are from the book [14] and papers [61, 62].

2.2.4.1 Types of semantic heterogeneity

Due to the diverse ways of developing ontologies, the semantic heterogeneities that may occur between different ontologies come with different types. [62] categorizes them mainly into three types: syntactic heterogeneity, terminological heterogeneity and conceptual heterogeneity.

Syntactic heterogeneity occurs when two ontologies expressed in different ontology language (OWL and XML) or modeled using different representation formalism (description logic and F-logic).

Terminological heterogeneity refers to mismatches occurring in the process of naming entities (e.g., instances, classes and properties) in different ontologies. Typical examples of these mismatches are

- Using different words to refer to the same entity
- Using the same word to refer to different entities
- Using word of different language to name entities
- Syntactic variations of the same word

Conceptual heterogeneity refers to the differences in modeling the domain of interest (i.e., what entities should be represent in an ontology and what is the definition of an entity). Many reasons may cause conceptual heterogeneity. Three important ones are listed below:

- **Coverage**: Two ontologies model different regions of the domain of interest.
- **Granularity**: Two ontologies model the same region of the domain of interest but at different level of detail.
- **Perspective**: Two ontologies model the same region of the domain of interest at the same level of detail but with different viewpoints.

The common approach of overcoming heterogeneity is to establish correspondence or relations between entities defined in different ontologies. These correspondences are established typically through the measure of the similarity between entities of different ontologies.

2.2.4.2 Basic techniques for measuring semantic similarity

To resolve the semantic heterogeneities between ontologies, a large number of complex similarity measures [14, 63] have been developed to assess the similarity between entities of different ontologies. These complex similarity measures typically assess the similarity between two entities by aggregating similarities between various features (e.g., labels, types and relations) of these two entities [64]. In this section, we only examine basic similarity measures that assess the similarity of two entities based on a single particular feature of these entities. Composite similarity measures based on multiple features used in this dissertation will be presented in Chapter 4. [14] groups the basic similarity measures into four broad categories: Name-based similarity, Structure-based similarity, Extension-based similarity and semantic-based technique.

Name-based (or label-based) similarity. Part of the semantics of an entity lies in the words used to name or label that entity. Name-based similarity computation attempts to measure semantic similarity between labels of two entities. The most widely used techniques for similarity between two labels are string-based measures. A large number of string-based similarity measures have been proposed, including Hamming distance, n-gram similarity, Jaro measure and edit distance [65]. Since a label is often a word or concatenation of several words, some of the name-based similarity measures require the label first be tokenized or normalized to a set of words and then the similarity score between two sets of word calculated. Since stop words (e.g., and, of) appear often and consequently have low information content in information theory [66], they can be removed from the similarity computation. To further enhance the accuracy of the Name-based measures, common knowledge corpora, such as WordNet[11], Wikipedia[12] and domain-specific corpora can be utilized to identify semantically similar words. For example, WordNet groups terms based on a notion of *synsets* or sets of synonyms. In other words, terms in the same synset are synonyms. Thus, two terms can be considered similar if they belong to the same synset. Name-based similarity is a common way used to resolve the terminological heterogeneity between entities.

Structure-based similarity. Structure-based similarity measures the similarity of two entities by comparing the structures of these entities. The most commonly used structures are property structure and taxonomic structure. Property structure is often used as comparison source for matching instances. The idea of comparing the property structures of two instances is that when two instances have similar properties with similar values, the two instances are likely referring to the same object. The taxonomic structure reflects the subsumption relationship of classes defined in an ontology. It is often used as comparison source for mapping classes. When two classes have similar taxonomic structure, the two classes are likely referring to the same concept. The Wu-Palmer similarity [67] and Upward Cotopic similarity [68] are often used to compute the similarity between two classes based on their taxonomies. Wu-Palmer measures the similarity between two classes of the same taxonomy. It takes into account the locations of classes in the taxonomy as well as the distance between classes in terms of edges. The Upward Cotopic similarity treats the taxonomic structure of a class as a set of its superclasses and compares two classes by applying the Jaccard similarity to the two sets of superclasses of these two classes. In this sense, Upward Cotopic similarity is useful to match classes that have different names but share similar taxonomic structures.

Extension-based similarity. Extension-base similarity measures normally compute the similarity between two classes (or properties) when their instances are

[11] http://www.wordnet.princeton.edu
[12] http://www.wikipedia.ord

available. The simplest way of comparing two classes in terms of their instances is to test the intersection of their instance sets. If the two classes share a large portion of instances, then they are likely referring to the same concept. A more formally defined extension-based similarity measure is the Jaccard similarity [69], which computes the similarity based on probabilistic interpretation of the set of instances. Extension-based similarities are useful to compute the overlap between two classes when instances of both classes are available. However, in situations where instances of classes are not available, other techniques have to be considered.

Model theoretic techniques. The model theoretic techniques cannot compute the similarity between entities or establish correspondences by itself. It is typically used to prune incorrect correspondences and discover new ones by using deductive techniques after certain correspondences between ontologies have been established. For example, after classes from two ontology O_1 and O_2 have been mapped to a reference ontology O, the relationships between classes in O_1 and O_2 can be inferred by using the reasoning ability of the ontology O. Deductive techniques such as propositional satisfiability (SAT) techniques can be also applied to check the completeness and consistency of the established correspondences [70].

2.3 Summary

Ontology-based information integration is a process that adopts a combination of information extraction, information retrieval, natural language processing, machine learning and semantic web techniques to (semi-)automatically extract features of interests (e.g., MSC instances) from raw unstructured text and/or (semi-)structured documents (e.g., XHTML web pages). Guided by ontologies, extracted features then are annotated and integrated based upon correspondences built among heterogeneous data sources by leveraging various similarity measures based on terminological, conceptual as extensional arguments. The basic techniques surveyed in §2.2.4.2 are the building blocks on which various mapping solutions are built. In Chapter 4, we will present similarity measures and mapping functions that we develop for discovering, annotating and integrating MSC instances and relations. Before going into detail on how these processes are performed, we will first overview the semantic resolution framework in the next chapter

Chapter 3.

Semantic Resolution Framework Overview

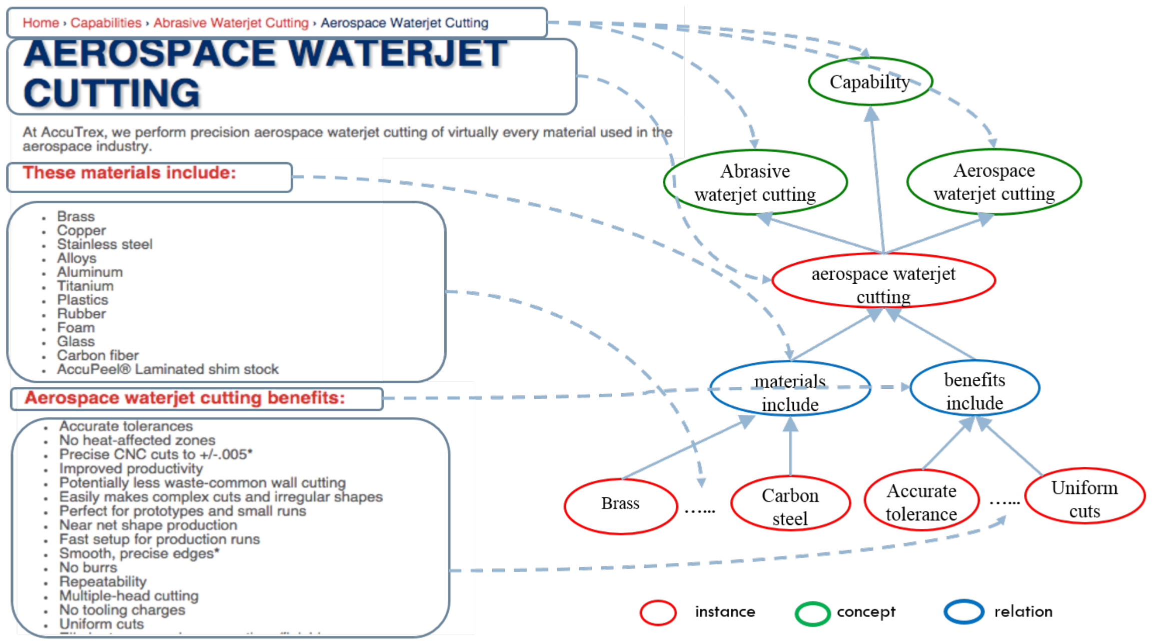
This chapter provides an overview of the semantic resolution framework (SRF). It starts with explaining the functionalities of core components that constitute the SRF and then, depicts the architecture of the SRF and describes how these components interact with each other.

3.1 Framework Components

This section describes in detail the functionalities of the three core components that constitute the semantic resolution framework. The three components together aim at addressing the challenges of extracting a large volume of manufacturing service capability instances from web sites of manufacturers, accurately annotating and integrating these instances into a manufacturing domain ontology (MDO) with minimal human intervention.

3.1.1 Instance Extractor

The instance extractor takes as input a website of a manufacturer and outputs an instance description model (IDM) of instances extracted from that web site. To facilitate the annotation of instances at later stage, IDM describes each instance n_i with a context containing a set of concepts of n_i with broader meaning of n_i and a set of relations that connects n_i with other instances or numerical data values. Intuitively, each concept indicates a category that n_i may belong to and each relation indicates a characteristic that n_i may have. Thus, concepts and relations of instance n_i describe the semantics of this instance and will be seen to be very useful in identifying a class label defined in MDO for n_i at the annotation stage later. Figure 3.1 shows an example that illustrates the entities (i.e., instances and relations) extracted from a web page and how these entities are described by IDM in graphic form.


Home › Capabilities › Abrasive Waterjet Cutting › Aerospace Waterjet Cutting

AEROSPACE WATERJET CUTTING

At AccuTrex, we perform precision aerospace waterjet cutting of virtually every material used in the aerospace industry.

These materials include:

- Brass
- Copper
- Stainless steel
- Alloys
- Aluminum
- Titanium
- Plastics
- Rubber
- Foam
- Glass
- Carbon fiber
- AccuPeel® Laminated shim stock

Aerospace waterjet cutting benefits:

- Accurate tolerances
- No heat-affected zones
- Precise CNC cuts to +/-.005*
- Improved productivity
- Potentially less waste-common wall cutting
- Easily makes complex cuts and irregular shapes
- Perfect for prototypes and small runs
- Near net shape production
- Fast setup for production runs
- Smooth, precise edges*
- No burrs
- Repeatability
- Multiple-head cutting
- No tooling charges
- Uniform cuts

Figure 3.1 Example of IDM describing instances extracted from a web page

The left picture of Figure 3.1 shows one web page of the accutrex.com[13] web site while the right picture shows the IDM that describes the entities extracted from this web page in graphic form. In this particular example, entities in red circles are instances, in green circles are concepts of instance **aerospace waterjet cutting**, in blue circles are relations applied to **aerospace waterjet cutting** and other entities in red circles with edges directed to blue circles are values of the relations. The concepts and relations of **aerospace waterjet cutting** forms the context of **aerospace waterjet cutting** and will be leveraged to identify class label for **aerospace waterjet cutting** in the instance annotation stage.

The fact that HTML provides no constructs to explicitly define schema of instances makes identifying contexts (i.e., concepts and relations) of instances a challenging task. Information extraction (IE) techniques may help identify those entities. However, they may not be robust enough to deal with the diverse ways of

[13] www.accutrex.com

presenting instance data in web pages. Moreover, IE techniques typically require manually created linguistic rules or entity recognizers. Thus, it is costly in terms of human effort. Our approach identifies instances and their contexts by leveraging site-independent features such as XHTML markup tags, hierarchical structure of XML source tree that represents web pages and the redundancy in web pages – the existence of multiple variations of the same phrase or term (e.g., the two phrases "industry served" and "we serve industries include" are actually referring to the same relation).

After identifying all instances and their contexts, we then semantically annotate these instances based on their contexts.

3.1.2 Instance Annotator

Given an instance description model (IDM) constructed from a manufacturer's web site, the instance annotator identifies a class label for each instance of the IDM based on the mapping established from the context of that instance to vocabularies (i.e., classes and properties) defined in the MDO. Then, these annotated instances are integrated into the MDO.

The challenge here is that ontology mapping techniques applied in traditional ontology-based information integration approaches cannot be applied directly to this instance annotation process. This is because that the objective of ontology mapping and instance annotation differs. More specifically, the ontology mapping is concerned with establishing correspondences between schema-level entities, typically classes and properties, of formally defined ontologies (e.g., OWL-based ontologies). Thus, in ontology mapping, ontology-specific semantics (e.g., class hierarchies and property restrictions [71]) defined for these entities are typically leveraged to establish correspondences. Further, based on established correspondences, instances of these ontologies can be integrated. While the instance annotation process focuses on identifying class labels from a formal ontology (i.e., MDO) for instances described by the local semantic model IDM that informally encodes the semantics of each instance via the context of that instance. To address this difference in formalism, we designed and developed a new scheme of annotating instances of the IDM.

More specifically, we first map each relation r of IDM to a property p of MDO that is most similar to r based on the similarity aggregated along three dimensions – domain, range and lexical label – between r and p. Then, for each instance n_i of IDM, we map concepts in the context of n_i to classes of MDO based on label similarity. Finally, we choose a class label from MDO for instance n_i based on the mappings established between entities in the context of n_i and those of MDO. Figure 3.2 shows an example that illustrates the mappings established between entities in IDM (on the left) and those defined in MDO (on the right).

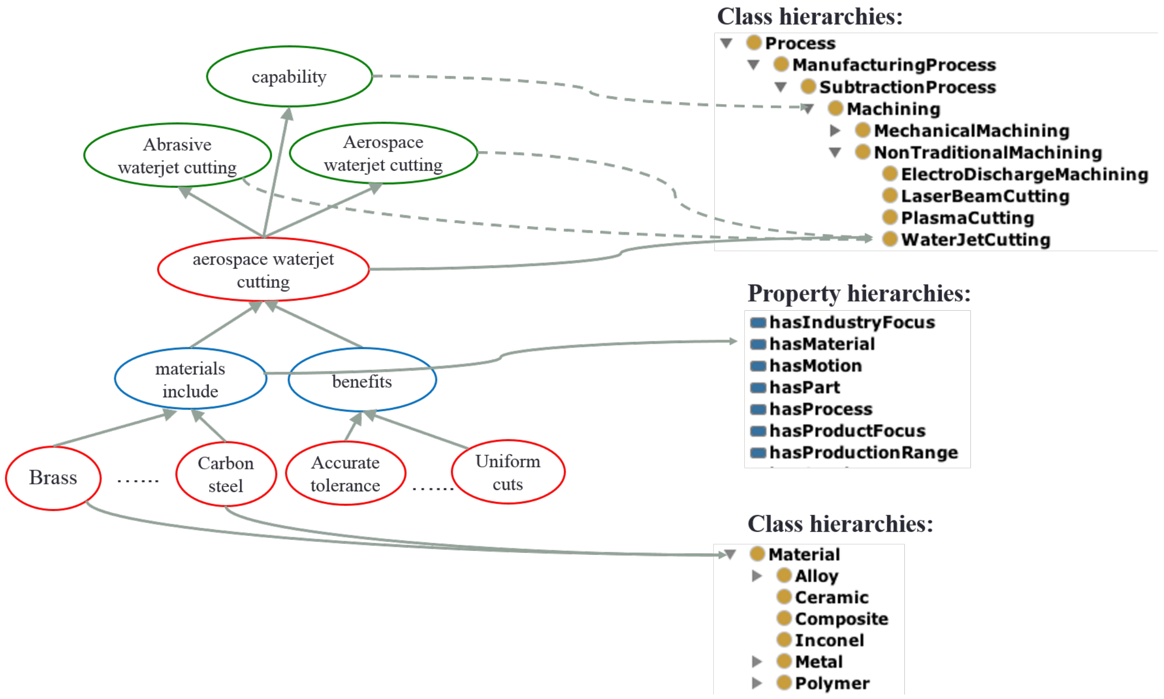


Figure 3.2 Example of mappings established between IDM and MDO

The solid lines in Figure 3.2 indicate that entities on the IDM side of the line are annotated by entities on the MDO side of the line. For example, **aerospace waterjet cutting** is annotated by class **WaterJetCutting** and **materials include** is annotated by property **hasMaterial**. The dash lines present the assistant mappings (established between concepts of instances in IDM and classes in MDO) that help identify class labels of instances. Note that the IDM shown in Figure 3.2 just a partial view of the whole accutrex.com web site and thus it does not show all the instances, concepts and relations.

Because of the semantic heterogeneity, the instance annotator may not be able to accurately annotate some instances. For example, as shown in Figure 3.2, the **Brass** and **Carbon steel** were annotated with the class **Material**, which is not accurate in describing the meaning of the two instances. With the help of the semantic resolution knowledge base, the accuracy of the instance annotator can be improved.

3.1.3 Semantic Resolution Knowledge-Base

The semantic resolution knowledge base (SR-KB) tries to address the technical challenges of effectively improving the accuracy of instance annotator while at the same time reducing or minimizing human intervention.

SR-KB provides a manufacturing concept mapping repository (MCMR) that each entry in this repository is a mapping that maps a concept u extracted from manufacturing web sites to a class v defined in MDO such that u and v have the same or similar meaning but share low label similarity. Thus, MCMR helps map more concepts of instances of IDM to classes of MDO and thereby may improve the accuracy of annotating instances by resolving terminological heterogeneity that exists between manufacturing web sites and the MDO.

SR-KB also offers a Naïve Bayes-based annotation corrector (NBAC) to automatically correct misannotated (i.e., assigned with wrong or inaccurate class label) instances. In the initial stage, the NBAC may not be mature enough to be able to correct misannotated instances. Thus, domain experts in the loop perform most of the work of validating instances and correcting misannotated ones. As the instance annotation process goes by, the NBAC is consistently trained by features extracted from instances that have been validated by domain experts, and its accuracy continuously improved. Consequently, the role of domain experts can be gradually replaced by the NBAC.

Last but not least, the SR-KB includes an instance validation platform (IVP) that assists domain experts to validate all annotated instances through a simple user interface. There are primarily two options open to domain experts to make correction:

(1) mappings from context of an instance to classes of MDO and (2) the class label of an instance. An innovative part of the IVP is that it offers a list of recommended class labels for each instance as alternatives to the original class label (automatically identified by instance annotator) of that instance. Thus, domain experts can correct a misannotated instance by simply choosing a class label from the list of recommended class labels rather than navigating the whole MDO that may contains hundreds or even thousands of classes.

3.2 Framework Architecture

As stated earlier, the semantic resolution framework (SRF) consists of three core components: including an instance extractor, an instance annotator and a semantic resolution knowledge base (SR-KB) and SR-KB in turn contains three subcomponents: a manufacturing concepts mapping repository (MCMR), a Naïve Bayes-based annotation corrector (NBAC) and an instance validation platform (IVP). The architecture of the five components and how they interact with each other are depicted in Figure 3.3.

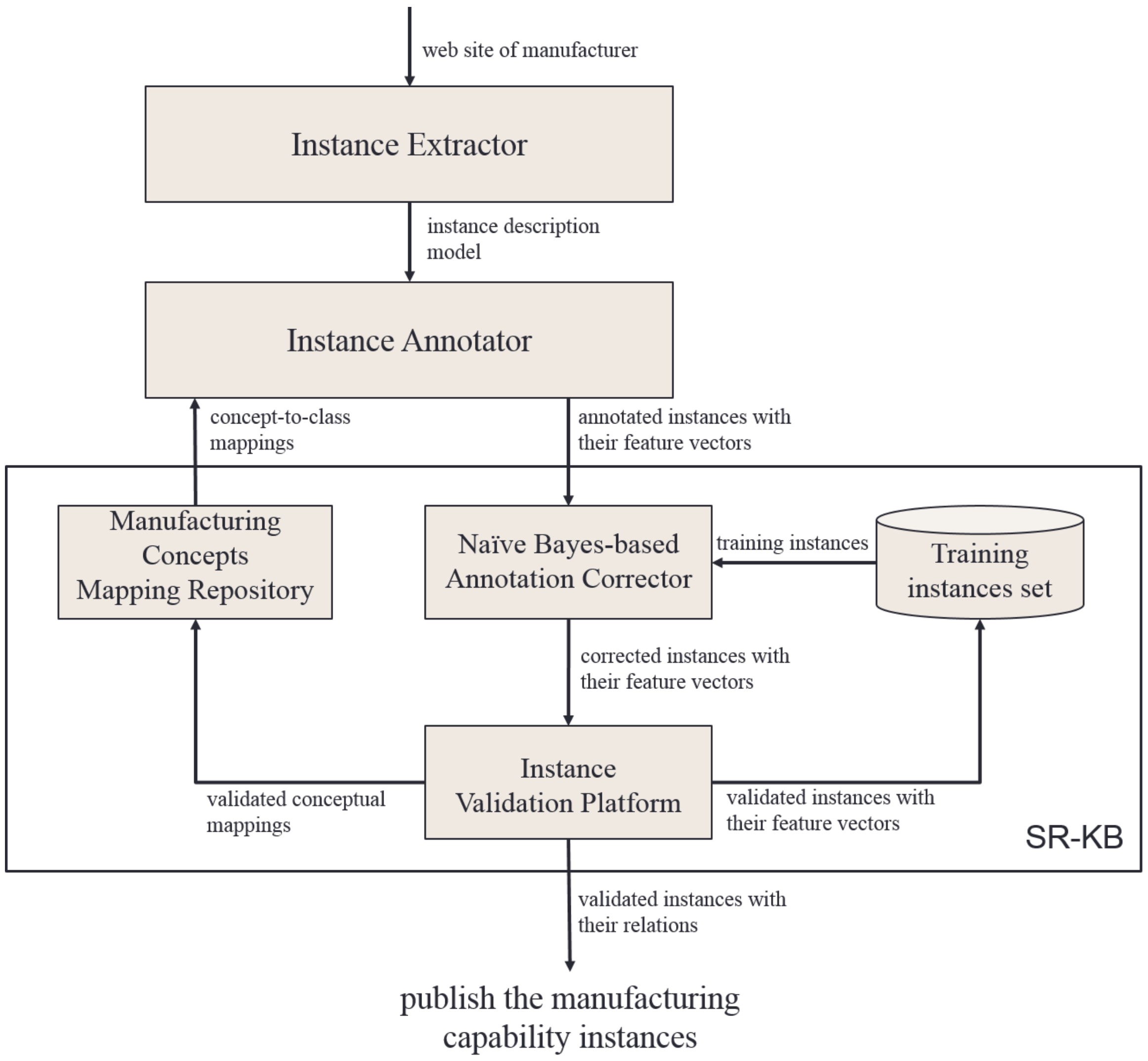

web site of manufacturer
Instance Extractor
instance description model
Instance Annotator
concept-to-class mappings
annotated instances with their feature vectors
Manufacturing Concepts Mapping Repository
Naïve Bayes-based Annotation Corrector
training instances
Training instances set
corrected instances with their feature vectors
Instance Validation Platform
validated conceptual mappings
validated instances with their feature vectors
SR-KB
validated instances with their relations
publish the manufacturing capability instances

Figure 3.3 Semantic resolution framework architecture

As shown in Figure 3.3, the instance extractor builds an instance description model (IDM) upon each manufacturing web site. The instance annotator takes IDM as input and automatically annotates instances of IDM with classes defined in manufacturing domain ontology (MDO) based on contexts of these instances and forms features of these instances. MCMR in SR-KB helps instance annotator more accurately annotate instances by resolving terminological heterogeneity while NBAC is to correct misannotated instances (i.e., assigned with wrong or inaccurate class label) by resolving conceptual heterogeneity. The NBAC is trained by features of annotated instances validated by domain experts with the assistance of

IVP. The output of IVP can be either stored in local database or published in Linked Open Data cloud.

3.3 Summary

In this chapter, we introduced the semantic resolution framework and its core components. Also, we briefly described the functionality of each component and the challenges that each component tries to solve. From Chapter 5 to Chapter 7, we will elaborate the methodologies and algorithms leveraged to implement these components. Before directly jumping into these chapters, in the Chapter 4 we will first define similarity measures and mapping functions that are commonly used by these methodologies and algorithms.

Chapter 4.

Similarity Measures

Similarity measures and mapping functions play a crucial role across the whole life cycle of the manufacturing service capability instance annotation process. Mappings between entities of different sources are typically established through the measure of similarity between those entities. In §2.2.4, we overviewed basic techniques for measuring similarity. In this chapter, we will present the measures we adopted to assess similarity between labels of entities and the functions to establish mappings between entities based on these similarity measures. Specifically, we first introduce three measures of label similarity between two entities. Then, we define two direction-dependent mapping functions and their respective mapping scores. Finally, a symmetric mapping score measure is defined.

4.1 Label Similarity Measures

Label similarity measures we use in this work are for calculating the lexical similarity between labels (or names) of two entities. Particularly, we adopt a string-based similarity measure to calculate the similarity between labels on the character level and a semantic-based similarity measure to calculate the similarity between labels on the word level. The latter one leverages WordNet [72] to match synonyms between words in two labels. Thus it may more accurately reflect the similarity between labels of entities. On the other hand, however, the accuracy of the semantic-based similarity measure is affected by the result of the tokenization preprocess on labels to be compared. Consequently, very similar labels but with different spitting points may obtain low label similarity. To address this issue, we come up with a hybrid similarity measure that takes both (string-based and semantic-based) measures into consideration. The three label similarity measures are described as follow.

String-based similarity measure. The string-based measure uses n-gram [74] to compute the label similarity between two entities. The n-gram similarity is known for its ability to deal with word variations (e.g., child vs children, process vs processing). It computes similarity between two labels (i.e., strings) by counting the number of occurrences of different n-grams (i.e., the substrings of length *n* in the label). More n-gram two labels have in common, the more similar the two labels will be. To compute the label similarity between two entities based on n-gram, we adopted the formula described in [14] which is re-stated bellow.

Definition 4.1: Let *ngram*(*e*, *n*) be the set of substrings of *e* of length *n*. The *ngram* similarity is defined as:

$$Sim^{L}_{ngram}(e_1, e_2) = \frac{2 \times |ngram(e_1, n) \cap ngram(e_2, n)|}{|ngram(e_1, n)| + |ngram(e_2, n)|}$$

Semantic-based similarity measure. The *ngram* does not consider the words comprising the label as well as the semantics of these words. To better measure the semantic similarity between labels, we explore an innovative approach developed by Kim [3]. This approach computes label similarity between two entities by using an ordered maximum-weighted bipartite matching algorithm that leverages WordNet to match synonyms. We define this ordered maximum-weighted bipartite matching algorithm as follow.

Definition 4.2: Given similarity measure σ that measures the similarity between two words, the *ordered maximum-weighted bipartite matching algorithm* (OMBM) that computes similarity between two sets of words that constitute the labels of two entities e_1 and e_2 respectively is defined as:

$$Sim^{L}_{OMBM}(e_1, e_2) = \frac{\max\limits_{m \in mappings(e_1, e_2)} \left(\sum_{(w_1, w_2) \in m} \sigma(w_1, w_2) \right)}{\max(|e_1|, |e_2|)}$$

where *mappings*(e_1,e_2) is a set of mappings, each of which is a one-to-one mapping that maps words in label of e_1 to words in label of e_2.

Kim applied an bottom-up dynamic programming algorithm to identify a mapping $m \in$ *mappings*(e_1,e_2) such that $\sum_{(w_1,w_2)\in m} \sigma(w_1,w_2) \geq \sum_{(w_1,w_2)\in m'} \sigma(w_1,w_2)$ for any mapping $m' \in$ *mappings*(e_1,e_2) and $m \neq m'$. σ computes the similarity between two words based on WordNet. The WordNet groups terms based on a notion of *synsets* or sets of synonyms. In other words, two terms can be considered similar if they belong to the same synset. This approach takes $O(|e_1||e_2|)$ to compute the label similarity between the two labels of entities e_1 and e_2, where $|e_1|$ and $|e_2|$ denote the numbers of words in the labels of entity e_1 and e_2, respectively. To reduce computational cost, a greedy heuristic strategy was proposed for implementing this algorithm. Although the greedy strategy is not globally optimal, it may be a better choice in situation where the computational cost is more of a concern.

Hybrid similarity measure. The hybrid similarity measure computes the similarity between labels of two entities considering both string-based and semantic-based similarity measures mentioned earlier, and returns the highest one. The objective of this hybrid similarity measure is to obtain the best result when comparing similarities between entities from different data sources, since applying either of the two aforementioned label similarity measures (i.e., string-based and semantic-based) alone may not be sufficient to deal with the terminological heterogeneity [75] that may exist between these sources. More specifically, labels of entities extracted from web pages typically are not designed for machine to process (e.g., tokenization). Consequently, the semantic-based similarity measures $Sim^L_{OMBM}(e_1,e_2)$ may return low similarity between labels of entities e_1 and e_2 even though the two entities are very similar. Consider entities "waterjet cutting" and "WaterJetCutting" as an example. It is clear to us that the two entities refer to the exact same manufacturing capability concept. However, $Sim^L_{OMBM}(e_1,e_2)$ returns a similarity score of only 0.64 between those two entities. This is because $Sim^L_{OMBM}(e_1,e_2)$ is not able to tokenize the string "waterjet" as two words. The hybrid similarity measure helps us alleviate such issues. We define the hybrid similarity measure as follow:

Definition 4.3: Given the Sim^{L}_{ngram} and $Sim^{L}_{OMBM}(e_1, e_2)$, the *hybrid similarity measure* is defined as:

$$Sim^{L}_{hybrid}(e_1, e_2) = \max(Sim^{L}_{ngram}(e_1, e_2), Sim^{L}_{OMBM}(e_1, e_2))$$

The choice of the most suitable measure among these three for calculating label similarity largely depends on the nature of linguistic features of entities to be compared, the particular task, and the expected response time. When computing label similarity between entities from the same data source, the string-based label similarity measure is usually applied. For computing label similarity between entities from heterogeneous data sources (e.g., mapping entities extracted from web pages to entities defined in the manufacturing domain ontology), semantic-based or hybrid similarity measures are more suitable.

4.2 Mapping Functions

Label similarity measures are concerned with measuring similarity between labels of two entities. They serve as basis for establishing mappings between entities from different (or the same) data sources. In this section, we will define an entity mapping function that maps one entity from a set to one entity from another set and an entity-set mapping function that maps a set of entities to another set of entities. We will also define measures to calculate mapping scores of these functions.

Definition 4.4: Given an entity $e_1 \in E_1$, the *entity mapping function* that identifies the counterpart of e_1 in entity set E_2 is defined as:

$$m_{\delta}(e_1, E_2) = \underset{e_2 \in E_2}{\arg\max}\, \delta(e_1, e_2)$$

where $\delta(e_1, e_2)$ is a generic similarity measure of the similarity between e_1 and e_2 in terms of certain aspects of the two entities. It can be a label similarity measure defined in §4.1 or any other more complex similarity measure based on different use cases. In the rest of this chapter, we will consistently use the symbol δ to refer to such generic similarity measure. In the following chapters, we will replace δ with specific similarity measures in different scenarios and different stages of integration. Based on Definition 4.4, the entity-set mapping function is defined as follow.

Definition 4.5: Given two sets of entities E_1 and E_2 and a similarity measure δ, the *entity-set mapping function* that identifies a set of directed correspondences from entities in E_1 to entities in E_2 is defined as:

$$M_\delta(E_1,E_2)=\{(e_1,e_2)|e_1\in E_1 : e_2=m_\delta(e_1,E_2) \text{ and } \delta(e_1,e_2)>d\}$$

Notice that each entity from E_1 is mapped to at most one entity from E_2. This means that we consider two entities mapped only when their similarity beyond a predefined threshold d.

The entity-set mapping function is directed and does not achieve 1-to-1 mapping. These properties are useful in identifying mapping between two sets of entities that entities in one set are not the exact paraphrases of entities in another [76]. The entity-set mapping function suits situations where one tries to find a reference for each entity of one set in the other set. In these cases, it is useful to map multiple entities from one set with similar or related semantics to a single entity in the other.

In the remainder of this chapter, we will define three functions used in different scenarios for computing mapping score between two sets of entities. The first two functions are non-symmetric measures: one is average mapping score and another is weighted mapping score. The last function is a symmetric measure called alignment score.

Definition 4.6: Given two sets of entities E_1 and E_2 and a similarity measure δ, the *average mapping score* between these two sets is defined as:

$$MS_\delta^A(E_1,E_2)=\frac{1}{|E_1|}\sum_{e_1\in E_1}\delta(e_1,m_\delta(e_1,E_2))$$

Definition 4.6 assumes all entities from E_1 are equally important. In certain situation, however, this may not be the case. For example, we associate each instance n_i extracted from manufacturing web sites with a context that represents the meaning of instance n_i. Such context contains a set of concepts each of which is associated with a score reflecting the relevance of that concept to n_i, We want to compute the score of a mapping established between the set of concepts and a set of classes defined in the manufacturing domain ontology considering the relevance of each concept to n_i. In this scenario, the concepts are not equally important with respect to instance n_i. To deal with this scenario, a weighted mapping score is required.

Definition 4.7: Given two sets of entities E_1 and E_2 and a similarity measure δ, the *weighted mapping score* between the two sets is defined as:

$$MS_{\delta}^{W}(E_1, E_2) = \sum_{e_1 \in E_1} \beta_{e_1} \delta(e_1, m_{\delta}(e_1, E_2))$$

where β_{e_1} denote the weight of entity e_1.

Definition 4.6 and 4.7 define two non-symmetric similarity measures that compute scores for directed mappings. Definition 4.8 defines a symmetric similarity measure between two sets of entities, dispite the asymmetric nature of the mappings. Definition 4.8 satisfies all the constraints that a symmetric similarity measure should satisfy, including (1) *positiveness,* (2) *maximality* and (3) *symmetry.* We call this symmetric similarity measure *cross mapping score*.

Definition 4.8: Given two sets of entities E_1 and E_2 and a similarity measure δ, the *cross mapping score* of the two sets of entities is defined as:

$$CS_{\delta}(E_1, E_2) = \frac{\sum_{e_1 \in E_1} \delta(e_1, m_{\delta}(e_1, E_2)) + \sum_{e_2 \in E_2} \delta(e_2, m_{\delta}(e_2, E_1))}{2 \cdot \max(|E_1|, |E_2|)}$$

4.3 Summary

In this chapter, we defined three label similarity measures, an entity mapping function and an entity-set mapping function. These measures and functions serve as building blocks for algorithms we developed to annotate and integrate manufacturing service capability instances from heterogeneous sources. In the next three chapters, we will explain in detail the approaches we take to extracting manufacturing service capability instances from web sites of manufacturers, annotate these instances with vocabularies defined in a manufacturing domain ontology, and correct misannotated instances, and see how these similarity measures and mapping functions applied in these processes.

Chapter 5.

Extracting Manufacturing Service Capability Instances

In this chapter, we elaborate the approach of extracting manufacturing service capability (MSC) instances from HTML web sites of manufacturers. As opposed to ontologies or many other data models that are able to explicitly define concepts and their relationships (*terminological knowledge*) as well as instances and their attributes (*assertional knowledge*) [77, 78]. HTML provides few constructs to fulfill such purposes. To facilitate the annotation of extracted instances with semantics defined in a manufacturing domain ontology and the integration of those instances under the ontology, we developed an instance extractor that, on one hand, can identify the context of each instance that helps to understand the meaning of such instance and on the other hand, is able to handle various forms of presenting instance data in manufacturing web sites. The instance extractor takes as input a manufacturing web site and outputs an instance description model that serves as a lightweight local semantic model of instances extracted from that web site. The instance description model (IDM) describes an instance n_i with a context consisting of a set of concepts that are related to but have broader meaning than n_i, and a set of relations relating n_i to other instances or data values. Intuitively, concepts of an instance can be considered as a set of tags indicating the categories that instance may belong to and relations describing the characteristics of that instance. Thus, the IDM describes instances in a structural way in term of concepts and relations. As a result, the IDM facilitates the annotation of manufacturing capability instances and the integration of these instances into OWL-based manufacturing domain ontologies.

This remainder of this chapter is structured as follows. In §5.1, we first define the XHTML tree model, instance description model and some related terminology that allows us to represent the manufacturing capability instance extraction problem. From

§5.2 to §5.6, we elaborate the approaches of extracting manufacturing capability instances and constructing the instance description model. We summarize this chapter in §5.7.

5.1 Definitions

In this section, we define the XHTML tree model of web pages from where we extract manufacturing service capability (MSC) instances and the instance description model that describes the extracted MSC instances.

5.1.1 XHTML Tree Model

A web page is processed and presented by a web browser through parsing the source code associated with that web page. The source code is typically written in Extensible Hypertext Markup Language (XHTML) [80], which is an application of the Extensible Markup Language (XML) [81], and it is organized in a tree structure. Therefore, we call source code of a web page *XML source tree* and define it as follow:

Definition 5.1: *XML source tree* is a XML serialization of a web page. It represents a web page in a machine-processable way. We define it as a 3-tuple:

$$X = (r, N, E)$$

where r is the root of XML source tree and N is a finite set of nodes, each of which is a descendant of r and has zero or more child nodes and one parent node. E is a finite set of edges, each of which connects a parent node to a child node.

The XML source tree has two types of nodes: leaf node and, inner node. We define them as follow:

Definition 5.2: Any node in N that has no child is called *leaf node.* Leaf nodes are also called atomic nodes since they are the smallest unit and cannot be further divided. We define a leaf node l in N of a *XML source tree* as 3-tuple:

$$l = (id, t, e)$$

where *id* uniquely identifies node *l*, *t* is the XHTML tag associated with node *l*, and *e* is an entity representing the textual content wrapped by tag *t*.

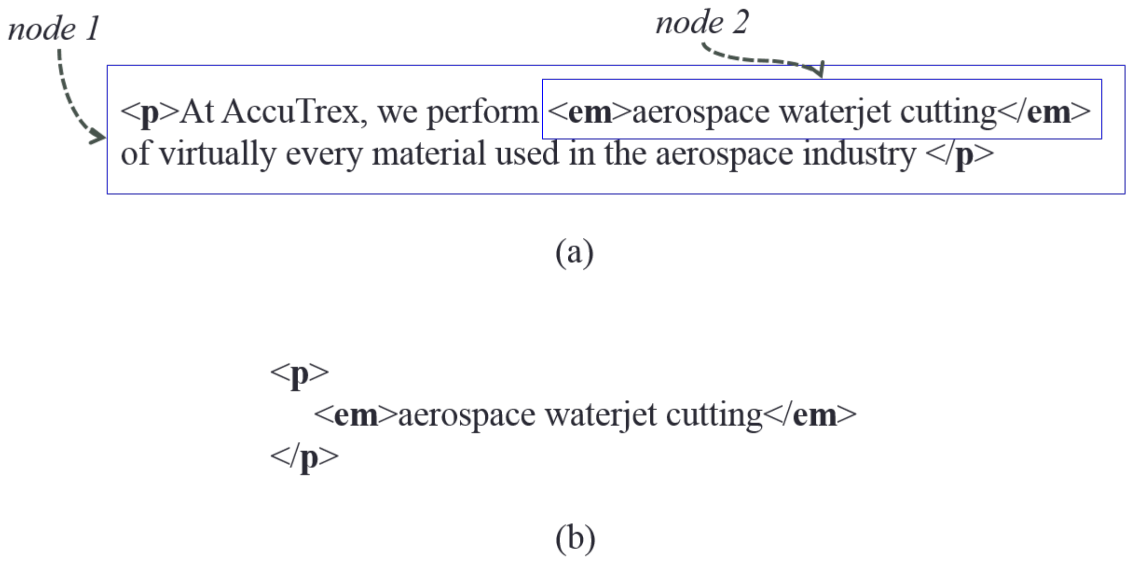
An entity can be a word, a phrase, a sentence, a paragraph or numerical data values or description (e.g., 6” x 4” x 1” wall thickness) that does not contain other tags. In practice, we limit the length (i.e., number of tokens) of entities to be extracted to a predefined threshold because we are only interested in manufacturing service capabilities expressed in short sentences or phrase, not in detailed descriptions of these capabilities or other irrelevant content.

Definition 5.3: An *inner node* is a XHTML node that has at least one child node that can be either a leaf node or other inner node. We define an inner node *m* in *N* of *XML source tree* as 3-tuple:

$$m = (id, t, X)$$

where *id* uniquely identifies node *m, t* is the XHTML tag associated with node *m*, and *X* is a set of node pointers that each is pointing to a leaf node or other inner node that is a direct child of *m*.

Definitions 5.2 and 5.3 impose restriction on the structure of XML source tree such that only leaf nodes contain entities of interest. In reality, however, an inner node *m* may also contain entities that are often intertwined with tags of children of *m*. Consider following paragraph marked by XHTML tags (i.e., <p> and <em>) as an example shown in Figure 5.1 (a). According to Definition 5.2 and 5.3, node 2 is a leaf node since it does not contain any child node but the entity “aerospace waterjet cutting”, while node 1 is an inner node since it contains node 2 as its child node.



```
<p>At AccuTrex, we perform <em>aerospace waterjet cutting</em>
of virtually every material used in the aerospace industry </p>
```

```
<p>
    <em>aerospace waterjet cutting</em>
</p>
```

Figure 5.1 Example of leaf node and inner node

We will not consider the text content (i.e., At AccuTrex, we perform) nested between tags <p> and <em> as well as text content (i.e., of virtually every material used in the aerospace industry) nested between </em> and </p> as entities of node 1 although they are wrapped by the tag (i.e., <p> and </p>) of node 1. Figure 5.1 (b) shows the diagram of the two nodes (i.e., node 1 and node 2) when the two pieces of texts are ignored. This restriction simplifies our instance extraction approach since we only consider leaf node when extracting entities. However, we may lose some information by ignoring the textural content nested among different XHTML tags. In the future, we will apply more sophisticated approaches to extract entities of interest from those texts.

XHTML tags play a crucial role in marking up data publishing on web pages and serve various purposes. To suit our purpose of forming contexts of instances, we group XHTML tags into three categories based on their intended usage. The three categories are listed as follow:

- *Topic tags:* A topic tag *t* is a XHTML tag that indicates an entity *e* wrapped by *t* is of certain importance and such entity *e* is typically a topic (e.g., a MSC instance, a relation or a category of MSC instances). The detailed characteristics or values of a topic are typically listed below that topic in the same web page. In our research, entities contained in topic tags are the main resources we explore to form the contexts of instances. Some of topic tags include <caption>, <strong>, <em>, <b>, <i>, <h1> to <h6>, <cite>, <dt> and <th>.

- *Function tags:* A functional tag is a XHTML tag that serves as certain functions such as triggering events, taking user inputs and transferring these inputs to web servers, or is a segment of code or script that drive these functions. In other words, functional tags aim at functionality of a web site rather than the content presentation. Thus, when we parse the XML source tree, we will ignore all the contents contained in function tags. Some of the function tags include <form>, <input>, <textarea>, <button>, <select>, <option>, <optgroup>, <fieldset>, and <label>.
- *Other tags:* Any tag that does not belong to the two categories listed above belongs to this category. We treat tags (e.g., <table>, <tr>, <ul>, <ol> and <span>) of this category as components constituting the XML source tree. The nested (i.e., hierarchical) structure of these tags determines the relationship between entities resided in these tags.

Since web browsers display a web page by parsing its XML source tree, the order in which a web browser (or XML parse tools) renders elements of the XML source tree matters with respect to constructing entity paths from our XML source tree (§5.2). We assume that all XML parsers render elements of a XML document following the *document order* [82], which is the order in which nodes appear in the XML serialization of a XML document. We define the *preceding siblings* of a node *n* (or a tag *t*) as the siblings that appear earlier than *n* (or *t*) by the document order. As shown in Figure 5.2, for example, node 2 and 3 are inner nodes; node 1, 4, 5, 6, 7, 8 and 9 are leaf nodes; the preceding siblings of node 3 are node 1 and node 2; the preceding siblings of node 6 are node 4 and node 5. Similarly, the preceding siblings of node 9 are node 7 and node 8.

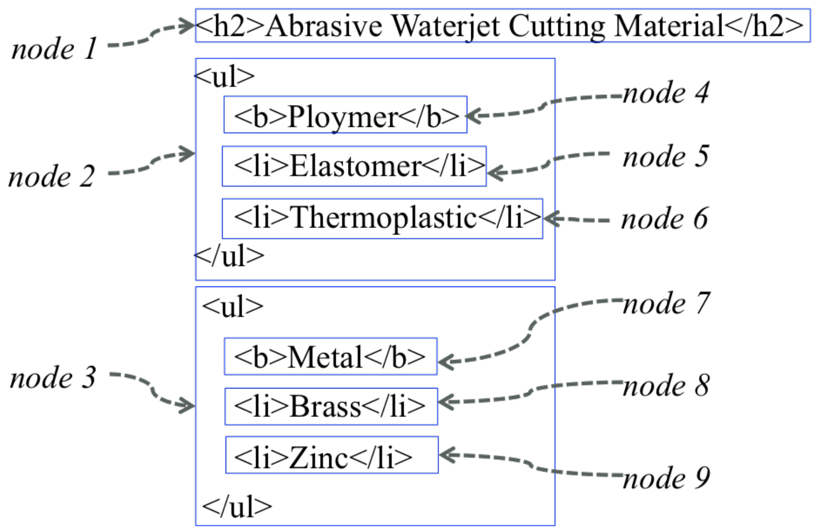


Figure 5.2 Example of nodes and their preceding siblings in XML source tree

We will parse and analyze XML source trees of web pages to extract entities of interest and connect related ones. These entities can be instances, relations that connect related instances or data values. To facilitate the annotation of extracted MSC instances, we designed an instance description model to describe instances and their relationships.

5.1.2 Instance Description Model

Instance description model (IDM) describes manufacturing service capability instances extracted from web pages of manufacturers in terms of contexts of these instances. The design goal of IDM is to facilitate the annotation of manufacturing service capability instances with semantics defined in a formal manufacturing domain ontology (MDO) and thus the integration of these annotated instances into MDO. We formally define the instance description model as follow:

Definition 5.4: Given a web site *W*, the *instance description model* of *W* is a 6-tuple (*I, K, R^o, R^d, N, D*), where:

$I = \{n_i\}$, the set of instances extracted from *W*.

$N = \{m_k\}$, the set of numerical data values.

$D = \{d_x\}$, the set of numerical data types extracted from *N*. All data types of members in *N* constitute this numerical data type set *D*. In §5.4.2, we will explain how the function $D(m)$ determines the numerical data type of $m \in N$ work.

$R^o = \{r_j\}$, the set of object relations that each connects two (related) instances, with $r_j = \{(s,o)|s \in I, o \in I\}$, $DS_{r_j} = \{s|(s,o) \in r_j\}$ is the domain of r_j and $RS_{r_j} = \{o|(s,o) \in r_j\}$ is the range of r_j. The label of r_j is denoted as λ_{r_j}.

$R^d = \{r_z\}$, the set of data type relations that each connects an instance with a numerical data value, with $r_z = \{(s,o)|s \in I, o \in N\}$, $DS_{r_z} = \{s|(s,o) \in r_z\}$ is the domain of r_z and $RS_{r_z} = \{D(o)|(s,o) \in r_z\}$ is the range of r_z. The label of r_z is denoted as λ_{r_z}.

$R = R^o \cup R^d$, the set of all relations

$K = \{K_{n_i}\}$, a set of contexts of instances;

where $K_{n_i} = (R_{n_i}, T_{n_i})$ is the context of instance n_i

where $T_{n_i} = \{t_{n_i,q}\}$ is the concept set of n_i,

$t_{n_i,q} = (\ell_{n_i,q}, \partial_{n_i,q})$ is the qth concept of n_i; $\ell_{n_i,q}$ is the label of $t_{n_i,q}$; $\partial_{n_i,q}$ is the relevance score of $t_{n_i,q}$ with respect to n_i

where $R_{n_i} = \{r_{n_i,q}\}$ is the relation set of n_i such that $n_i \in DS_{r_{n_i,q}}$, $r_{n_i,q}$ is the qth relation of n_i

Definition 5.4 defines components describing instances extracted from a manufacturing web site. Intuitively, these components can be considered as terminological knowledge on extracted instances. In the following, we define the assertional knowledge based on these components. This assertional knowledge together with the terminological knowledge constitute the knowledge base of a manufacturing web site. The objective of the semantic resolution framework presented in this dissertation is to integrate the knowledge bases extracted from a large amount of manufacturing web sites into a manufacturing domain ontology (MDO).

Definition 5.5: Given the instance description mode IDM of a web site *W*, the *assertions* of IDM is a 2-tuple (A^o, A^d), where:

$A^o = \{(s, r^o, o) | r^o \in R^o, s \in DS_{r^o}, o \in RS_{r^o}\}$, the set of assertions in the form of subject-predicate-object triple, in which the predicate is an object relation

$A^d = \{(s, r^d, o) | r^d \in R^d, s \in DS_{r^d}, D(o) \in RS_{r^d}\}$, the set of assertions in the form of subject-predicate-object triple, in which the predicate is a data type relation

$A = A^o \cup A^d$, the set of all assertions

I, R^o, R^d, and *N* are signature of IDM. They are disjoint sets and each is a set of entities identified from a manufacturing web site. Members of other sets (e.g., A^o, A^d and *K*) are formed by members from the four sets. A^o and A^d are sets of assertions that contain facts about the instances of IDM. IDM does not explicitly describe the type information of instances or the domains and ranges of relations. This is because the XML source tree of web pages from where instances and relations were extracted contains no explicit type information on instances and relations.

IDM describes types of instances through the contexts of these instances. The context of an instance n_i can be used to disambiguate and determine the meaning of n_i, thereby helping identify a class label defined in the MDO for instance n_i. The context of n_i, denoted as K_{n_i}, is represented by a set of instances, denoted as T_{n_i}, where each instance is related to but typically has boarder meaning than n_i, and a set of relations, denoted as R_{n_i}, where each relation connects n_i to other instances or numerical data values. Each instance in T_{n_i} is also referred to as a concept of n_i since it can be a topic that is described by n_i or a category that n_i may belong to. Thus, we call T_{n_i} the concept set of instance n_i. Each concept $t_{n_i,q} \in T_{n_i}$ is associated with a relevance score $\partial_{n_i,q}$ representing the degree of relevance of that concept to instance n_i. Table 5.1 show an example of the context for instance **Aerospace Waterjet Cutting** extracted from web site of AccuTrex Products, Inc[14].

Table 5.1 Context of instance Aerospace Waterjet Cutting

Instance: Aerospace Waterjet Cutting			
Context	**Concepts**:	Aerospace Waterjet Cutting	$\partial = 1.0$
		Abrasive Waterjet Cutting	$\partial = 0.8$

14 http://www.accutrex.com/

		Capability	$\partial = 0.64$
	Relations:	material include	$\partial = 0.8$
		waterjet cutting benefits	$\partial = 0.8$

Notice that each concept in the concept set of an instance is an instance itself, thus, the context of which would be formed accordingly. Put another way, an instance in the IDM has two roles: one is an instance with a context and the other is a concept appearing in the contexts of other instances. Acting as the first role, an instance will be annotated with a class label in the instance annotation process (Chapter 6). Acting as the second role, an instance will help us to identify class labels for other instances.

A relation *r* of IDM connects two related instances if *r* is object relation or connects an instance to a numerical data value if *r* is data type relation. IDM describes domain of *r*, denoted as DS_r, as a set of instances described by relation *r*. Since an object relation connects an instance to another instance while a data type relation connects an instance to a numerical data value, IDM treats the ranges of the two types of relations differently. Specifically, the range of an object relation r^o, denoted as RS_{r^o}, is represented by a set of instances that are values of *r*. While the range of a data type relation r^d, denoted as RS_{r^d}, is represented by a set of data types determined based on numerical data values of r^d. Notice that we only allow the value of a data type relation to be a numerical data value or numerical description such as ".001 in - 1/4 in" and "Up to 144 in". Any other types of values of a relation are considered as instances. The reason for this consideration is that it typically has no clue at this instance extraction stage that can help determine whether an entity published in a web page is a string or an instance of certain type (i.e., class). Thus, we treat all entities that were not identified as numerical data values or relations (§5.4) as instances of certain types. Then, at the instance annotation stage (Chapter 6), we assign each instance with a class label.

Assertion set *A* of IDM are facts on instances and in the form of subject-predicate-object triple. The predicate of an assertion is a relation that relates an instance (i.e., subject) to another instance or a numerical data value (i.e., object). When the predicate of an assertion *a* is an object relation, *a* belongs to A^o. Otherwise, *a* belongs to A^d.

5.2 Constructing Entity Path

According to Definition 5.4, to construct instance description model (IDM) from web site of a manufacturer, we need to identify members of the signature of IDM (i.e., instances, relations and numerical data values) from the contexts of instances as well as domains and ranges of relations. However, HTML provides no constructs to explicitly distinguish and connect these members. To address this challenge, we extract and connect related entities appearing in the same web page in the form of *entity paths* based on the structure of the XML source tree. An entity path is a directed path such that each edge is directed from child entity to its parent. Each entity in the entity path can be an instance, a relation or a numerical data value. Based on the observations, which will be given shortly, a child entity *e* can be either a characteristic of or a specific instance of its parent. In the former case, the entity *e* is a relation of its parent and the value of such relation is the child of *e*. In the latter case, the parent of entity *e* is a concept of *e* (i.e., a member of the concept set in the context of *e*). Notice that the same entity *e* may appear in multiple entity paths constructed from different web pages of the same web site and thus the context of *e* should comprise concepts and relations of *e* extracted from all entity paths that *e* appears in. Thus, we merge all entity paths of a web site by fusing similar entities to form *entity graph* (§5.3) that connect related entities on the site basis. The structure of entity graph enables us to construct the context for each instance.

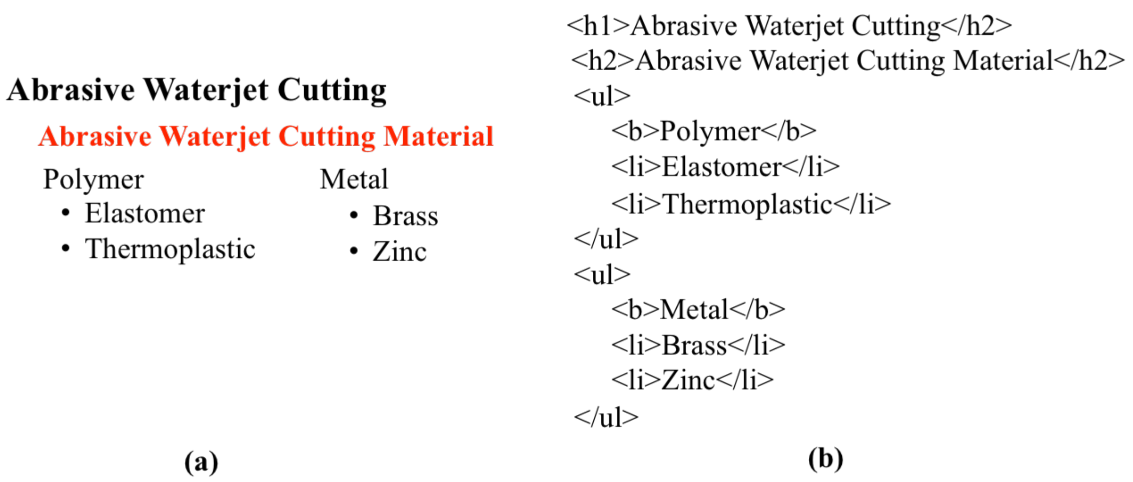
The approach of extracting entity paths is based on the following two observations on the structure of manufacturing web pages and the XML source tree.

1. A manufacturing web site typically dedicates one or more description pages to describe the features of manufacturing service capabilities it offers. In each page, the description of a MSC is typically presented in a hierarchical (tree) structure such that an entity appearing at lower level of the hierarchy typically describes a specific aspect or is a specific instance of entities appear at higher level of the same hierarchy. In other words, entities that appear at higher level than entity *e*

does typically have broader meaning than *e* and should be included in the context of *e*. Figure 5.3(a) illustrates a snippet of the web page that describes the materials used by Abrasive Waterjet Cutting. For example, it shows that Elastomer is an instance of Polymer which in turn is an instance of material used by Abrasive Waterjet Cutting.

Note that the first observation is based on the hierarchical structure of entities presented at the web browser. Such hierarchical structure is not the same as the structure of entities encoded in the XML source tree of web page that is processed by computer programs, although they share some commonalities. The second observation helps to deal with this problem.

2. In the XML source tree of a manufacturing web page, an entity e_1 residing in a node n_1 typically describes a specific aspect of or is a specific instance of an entity e_2 residing in a node n_2 that is n_1's nearest preceding sibling with a topic tag. Thus, e_2 is a topic described by e_1 and is added as parent of e_1 to the entity path. If we cannot find the topic of e_1 among the preceding siblings of n_1, we will recursively search preceding siblings of ancestors of n_1 for such topic. For example, as shown in Figure 5.3(b), **Thermoplastic**'s topic (i.e., **Polymer**) is found at its nearest preceding sibling with topic tag <b> while **Polymer**'s topic (i.e., **Abrasive Waterjet Cutting Material**) is found at its parent's nearest preceding sibling with topic tag <h2>. In turn, **Abrasive Waterjet Cutting Material**'s topic (i.e., **Abrasive Waterject Cutting**) is found at its nearest preceding sibling with topic tag <h1>. This observation is crucial for the instance extractor to extract entity path that each entity is a topic for its child, thereby helping form the contexts of instances.



Abrasive Waterjet Cutting

Abrasive Waterjet Cutting Material

Polymer
- Elastomer
- Thermoplastic

Metal
- Brass
- Zinc

(a)

```
<h1>Abrasive Waterjet Cutting</h2>
<h2>Abrasive Waterjet Cutting Material</h2>
<ul>
    <b>Polymer</b>
    <li>Elastomer</li>
    <li>Thermoplastic</li>
</ul>
<ul>
    <b>Metal</b>
    <li>Brass</li>
    <li>Zinc</li>
</ul>
```

(b)

Figure 5.3 A snippet of a manufacturing web page and its corresponding XML source tree.

Based on these two observations, our instance extractor constructs an entity path starting from a leaf node of XML source tree. Experiments show that our instance extraction algorithm based on these two observations works well. However, by no means do we assume that all manufacturing web pages follow the same structure as stated above and all concepts are residing in topic tags. Occasionally, a child entity may have a more general meaning than its parent and topics may locate in other types of tags. The issues of missing potential topics or collecting incorrect ones for certain entities can be alleviated by forming an entity graph (§5.3) from entity paths of the whole web site. This is because entity graph collects concepts of entities at the site level and it may locate the missing concepts of entities in different web pages. The algorithm for constructing entity path is given bellow.

Algorithm 5.1: Given the XML source tree of a web page, *the entity path extraction algorithm* works in three steps:

Step 1. Extract leaf nodes of the XML source tree of a given web page following the document order and store these leaf nodes in list *L*.

Step 2. Fetch a leaf node *l* from *L*, create an empty entity path *EP* and add entity of *l* to *EP* as the leaf entity of *EP*. Scanning upwards the XML source tree for *l*'s preceding sibling *g* that has a topic tag that is not the same as the tag of node *l* (if the siblings *g* and *l* have the same tag, we consider them in the same level. Thus, *g* will not be considered as the parent of *l*). If *g* is found, add entity of *g* to *EP* as parent of *l*. Then go to *Step 3*.

Step 3. Go one level up to the parent *p* of the current node and scan upwards the XML source tree for *p*'s preceding sibling *g'* that contains a topic tag that is not the same as the tag of node *p*. If *g'* was found, add entity of *g'* to *EP* as the parent of *p*. If reaches the top of the XML source tree, Go to *Step 2.* Otherwise, go to *Step 3.*

Algorithm 5.1 describes an approach of extracting entity paths based on the hierarchical structure of MSC information published in web pages. With small revision, Algorithm 5.1 is able to parse tables with headers of data records presented in the first column of each row. We call these headers *row headers* and table with row headers *row header table*. A row header table can be considered as a special hierarchical structure of presenting MSC information. The row header of a data record typically suggests the type information of or the topic described by data items in that record. Thus, when constructing an entity path, we extract the row header of a row *r* as the parent entity of data items presented in row *r*. As shown in Table 5.2, the first column presents row headers indicating topics described by data items presented in the second column.

Table 5.2 Example of a simple row header table

Fabrication Method	Abrasive Water Jet Cutting Flame Cutting Shearing
Materials	Stainless Steel Aluminum Abrasion Resistant Plate
Additional Services Provided	Reverse Engineering CAD Drafting Assembly

Figure 5.4 shows the XML source tree of Table 5.2. In this XML source tree, each row header resides in the first *table data tag* <td> of its corresponding row that in turn resides in *table row tag* <tr>, while data items reside in the following <td> tags (in this particular example, all data items are presented in the second column). To

enable Algorithm 5.1 to parse row header tables and extract entity paths from them, we revise the *Step 3* of Algorithm 5.1 as follow:

Step 3. Go one level up to the parent *p* of the current node. If node *p* contains a tag of <td>, directly go to the farthest preceding sibling (or the first preceding sibling in document order) of *p* and search inside for node *d* containing a topic tag. Otherwise, scan upwards the XML source tree for *p*'s preceding sibling *g'* that contains a topic tag. If either node *d* or node *g'* was found, add the entity of *d* or *g'* to *EP*. If reaches the top of the XML source tree, Go to *Step 2*. Otherwise, go to *Step 3*.

Intuitively, the revised *step 3* first checks whether the node *p* is a data item contained in a table. If it is, identify the row header of the row that *p* is residing in. if it is not, follow the normal way of finding the topic of *p*. From our investigation of the 45 manufacturing web sites we used to conduct the experiment, 39 out of the 45 web sites (86.7%) utilize row header tables to describe detailed information of manufacturing service capabilities. Thus, we assume that the revised version of Algorithm 5.1 is able to deal with the majority of tables used in manufacturing web sites.

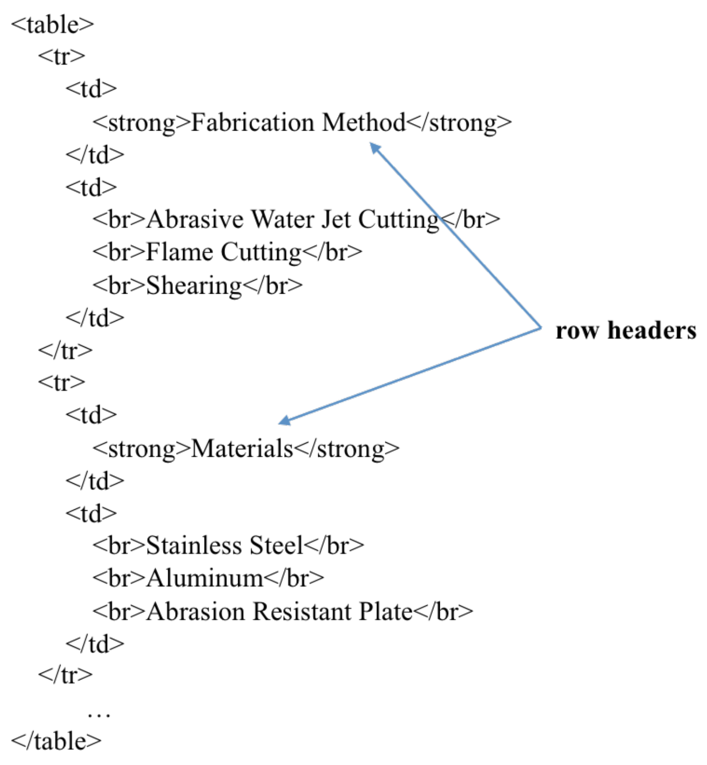

```
<table>
  <tr>
    <td>
      <strong>Fabrication Method</strong>
    </td>
    <td>
      <br>Abrasive Water Jet Cutting</br>
      <br>Flame Cutting</br>
      <br>Shearing</br>
    </td>
  </tr>
  <tr>
    <td>
      <strong>Materials</strong>
    </td>
    <td>
      <br>Stainless Steel</br>
      <br>Aluminum</br>
      <br>Abrasion Resistant Plate</br>
    </td>
  </tr>
      …
</table>
```

Figure 5.4 XML source tree of Table 5.2 (Only first two rows are presented)

One issue with the entity path extraction algorithm is that an (sub-) entity path may be generated multiple times, since a leaf entity of an entity path may also appear as inner entities in other entity paths. For example, Figure 5.5 shows three entity paths constructed from three leaf nodes Polymer, Elastomer and Thermoplastic presented in the Figure 5.3(b).

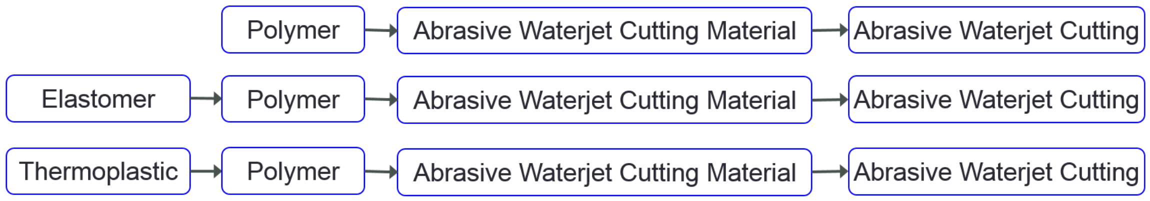

Polymer
Abrasive Waterjet Cutting Material
Abrasive Waterjet Cutting
Elastomer
Polymer
Abrasive Waterjet Cutting Material
Abrasive Waterjet Cutting
Thermoplastic
Polymer
Abrasive Waterjet Cutting Material
Abrasive Waterjet Cutting

Figure 5.5 Example of repetitively constructed entity paths

The first entity path is a sub-path of the second and third ones and it has been constructed three times. Repetitively constructing the same sub-path decreases the efficiency of the entity path extraction algorithm. To address this issue, after an entity path *p* is constructed, we cache *p* in the form of key/value pair. The key is the *id* of the leaf node from where *p* was constructed and the value is the entity path *p*. In this particular example, the first entity path is cached in the form as follow.

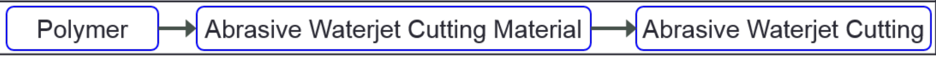
key	value
#Polymer	

Figure 5.6 Example of an entity path catch entry

The next time when the instance extractor comes across a node with an *id* that exists in the cache, the instance extractor directly fetches the value (i.e., entity path) corresponds to that *id* from the cache instead of constructing it from scratch. This way, each entity path will be constructed only once.

5.3 Constructing Entity Graph

While an entity path connects related entities appearing in the same web page, an entity graph of a web site connects related entities across the entire web site. Intuitively, entity graph are formed by merging similar entities (i.e., entities with similar labels) in entity paths constructed from that web site. More specifically, we implement entity paths as ordered lists in which each edge is directed from child node to parent node. Then, we navigate each entity path from tail (i.e., leaf node) to head and add two adjacent nodes with their directed edge to the entity graph at a time. The label of each entity is normalized before added to the entity graph. Entities with the same normalized label are merged as one entity node in the entity graph. Algorithm 5.2 explains in detail how entity paths are merged into entity graph and Figure 5.7 illustrates how Algorithm 5.2 works.

Algorithm 5.2: Given a set of entity paths *EP* extracted from a web site, *the entity graph construction algorithm* forms an entity graph *EG* by performing following three steps:

Step 1. Fetch an entity path *p* from *EP*, denote the index of entity in *p* as *i* and initialize index *i* to 0 (pointing to the tail of *p*).

Step 2. Fetch two adjacent entities e_i and e_{i+1} from *p* starting from leaf node of *p* and denote the pair as an edge $<e_i, e_{i+1}>$ in which e_i is directed to e_{i+1}. Normalize the label of each entity in $<e_i, e_{i+1}>$ and then add $<e_i, e_{i+1}>$ to *EG* by performing following set operation.

$$EG = EG \cup \{< e_i, e_{i+1} >\}$$

Increase index *i* by 1. If index *i* is smaller than the length of *p*, go back to *Step 2*. Otherwise, go to *Step 3*.

Step 3. If no entity path exists in *EP*, terminate the algorithm. Otherwise, go to *Step 1*.

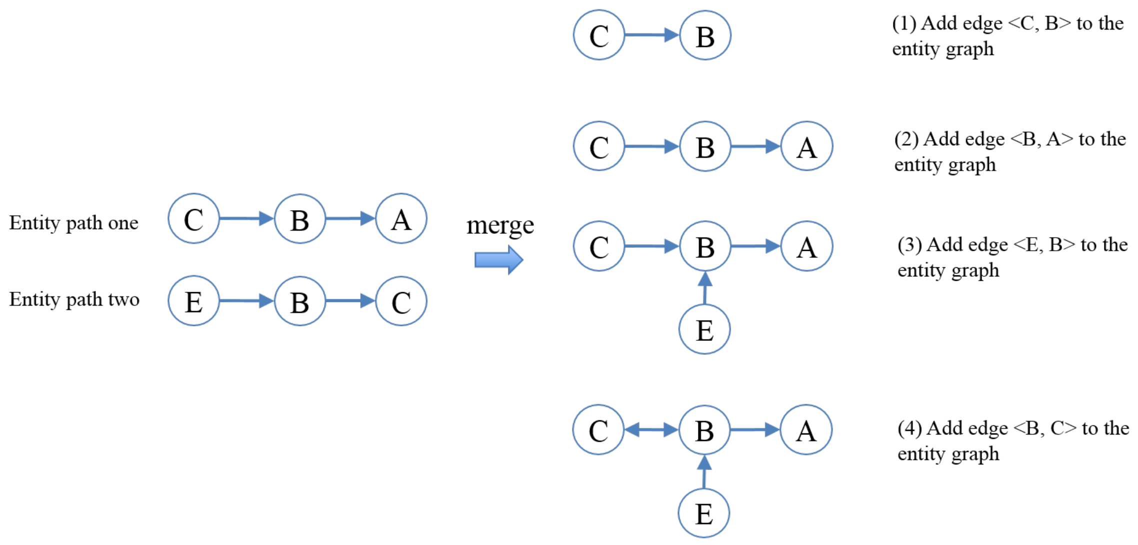

Entity path one
C
B
A
Entity path two
E
B
C
merge
(1) Add edge <C, B> to the entity graph
(2) Add edge <B, A> to the entity graph
(3) Add edge <E, B> to the entity graph
(4) Add edge <B, C> to the entity graph

Figure 5.7 Example of merging two entity paths to form the entity graph

Entities that appear in multiple web pages of a web site may describe the terminological knowledge (e.g., concepts of instances and relations connecting instances) of that web site. Thus, by constructing the entity graph of a web site we are able to identify relations and concepts of instances and thus form contexts of instances. Figure 5.8 and 5.9 show two concrete examples of merging entity paths into a sub-entity-graph. These two sub-entity-graphs illustrate two typical structures of entities from where relations and concepts of instances can be identified.

We interpret the merged sub-entity-graph shown in Figure 5.8 as relational structure of entities. In this structure, we consider **benefits include** as a relation that connects specific waterjet cuttings with benefits of these waterjet cuttings. These waterjet cuttings form the domain of **benefits include** while benefits form the range of **benefits include**. In §5.4, we will elaborate our approach of identifying relations from the entity graph.

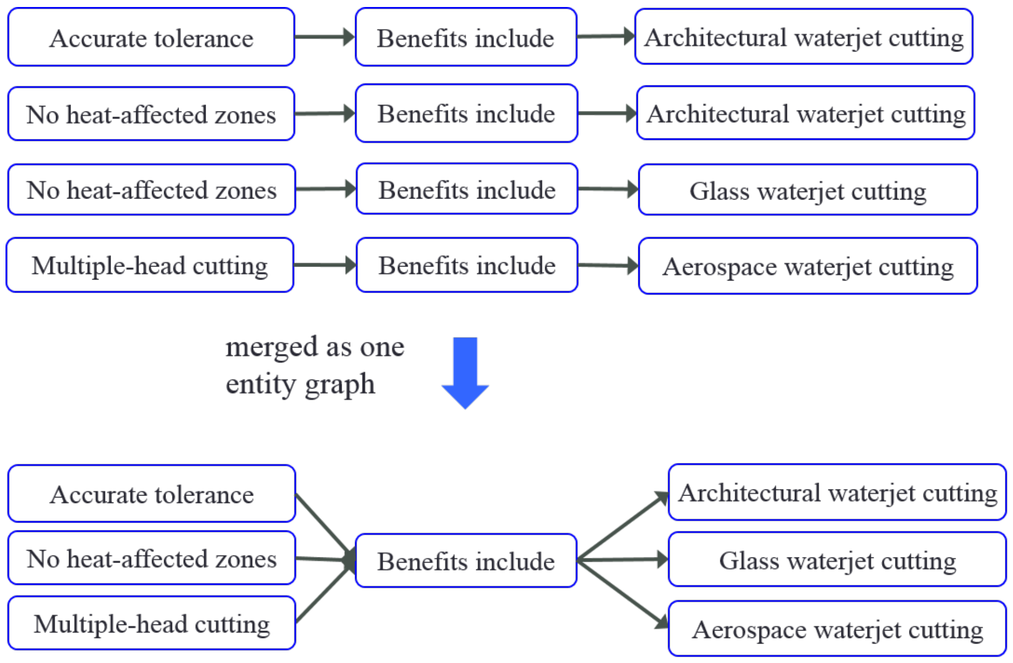


Figure 5.8 Example of sub-entity-graph of relational structure

Although by forming entity graph, we are able to identify potential relations that connect instances, entity graph lost the information of how specific instances are related via relations. Such information is coded in entity paths. Therefore, we will store both entity graph and entity paths constructed from a web site to form the whole IDM of that web site. We will elaborate this point in §5.6.

According to the Definition 5.4, the relations identified for an instance partially form the context of that instance. The rest of the context is formed by concepts of that instance. The sub-entity-graph shown in Figure 5.9 illustrates the hierarchical structure of entities from where the concepts of entities can be identified. Entities at higher level of the hierarchical structure can be seen as concepts of entities at lower level since they typically have broader meaning than entities at the lower level. Thus, in this particular example, **Capability** is a concept of **Abrasive waterjet cutting** and **Laser cutting**. **Capability** and **Abrasive waterjet Cutting** are concepts of **Architectural waterjet Cutting**, **Glass waterjet cutting** and **Aerospace waterjet cutting** while **Capability** and **Laser cutting** are concepts of **CNN laser cutting.**

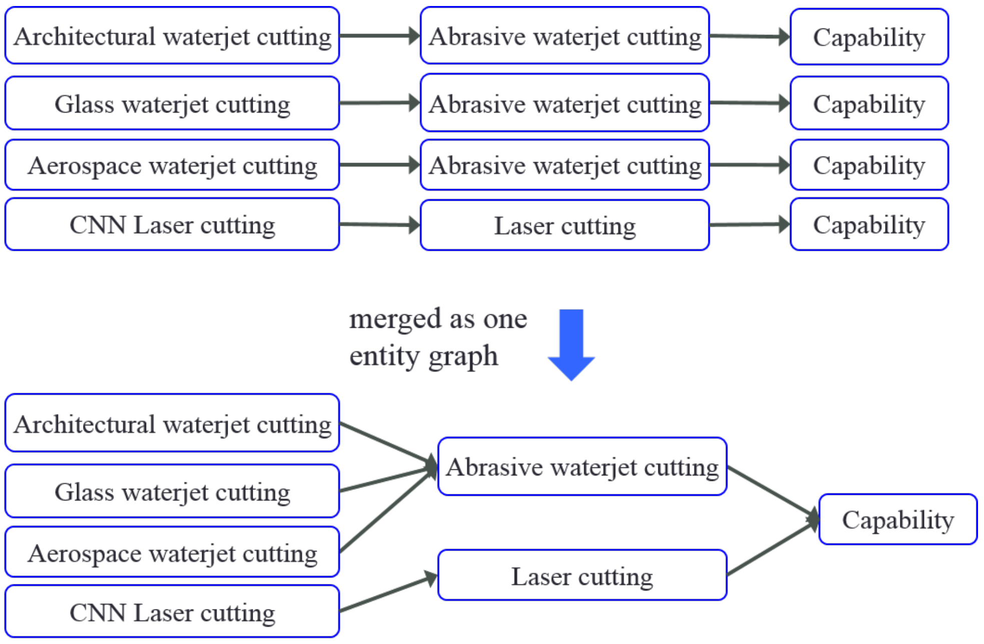

Architectural waterjet cutting
Abrasive waterjet cutting
Capability
Glass waterjet cutting
Abrasive waterjet cutting
Capability
Aerospace waterjet cutting
Abrasive waterjet cutting
Capability
CNN Laser cutting
Laser cutting
Capability
merged as one entity graph
Architectural waterjet cutting
Glass waterjet cutting
Aerospace waterjet cutting
CNN Laser cutting
Abrasive waterjet cutting
Laser cutting
Capability

Figure 5.9 Example of sub-entity-graph of hierarchical structure

Figure 5.10 depicts a snippet of the entity graph constructed from the web site of Accutrex Products, Inc. It includes, among other things, the two sub-entity-graphs shown in Figure 5.8 and Figure 5.9.

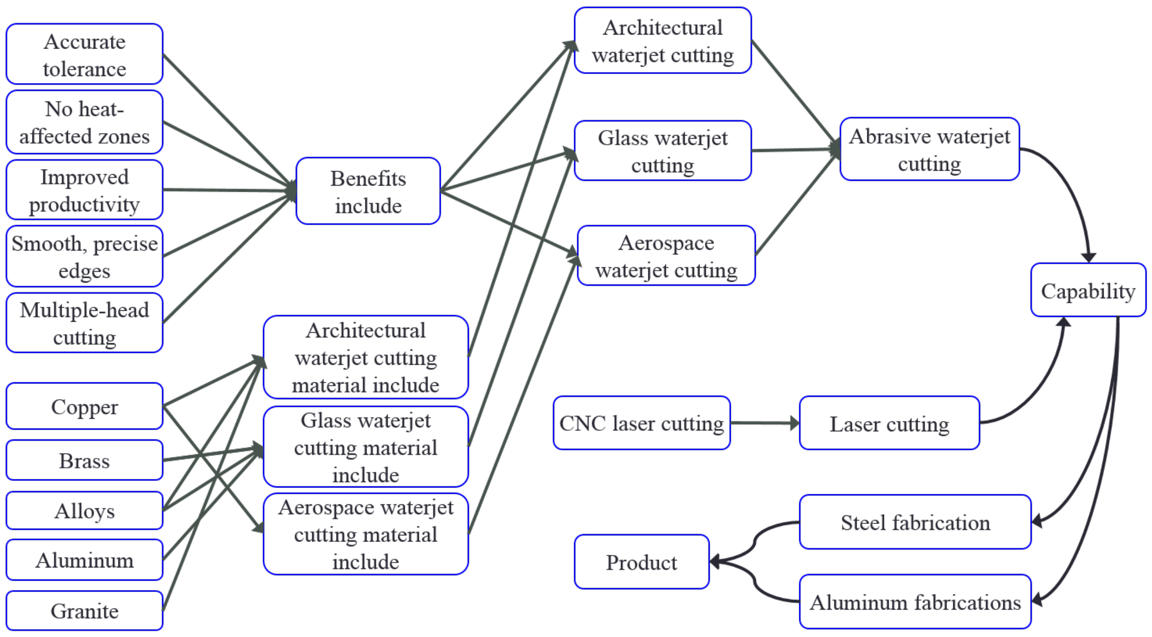


Figure 5.10 Example of an entity graph.

We create the instance description model (IDM) of a web site based on entity graph constructed from that web site. Each entity in an entity graph belongs to one of the four sets of IDM: *I, R^o, R^d* and *N*. Specifically, we first identify relations (i.e., R^o and R^d), instances (i.e., *I*) and numerical data values (i.e., *N*) from the entity graph, and then form contexts (i.e., *K*) of instances as well as assertions (i.e., A^o and A^d) on instances according to the Definition 5.4. In the next three sections, we will elaborate each of these processes in detail.

5.4 Identifying Relations

To identify relations from entity graph, we apply the heuristic that a relation applied in a web site describes multiple distinct instances and has multiple distinct values. In other words, a relation should have the cardinality of m-to-n. Based on this heuristic, entities with multiple ancestors and descendants in an entity graph can be considered as relations. However, directly applying this heuristic may miss out many potential relations. This is because a relation may appear multiple times across the web site with some variations of its label and each variation may be dedicated to describing only one particular instance. For example, as shown in Figure 5.10, **Architectural waterjet cutting materials include** and **Glass waterject cutting materials include** are two specific relations that each states that a particular waterject cutting includes some materials. Those two entities have a cardinality of 1-to-n instead of m-to-n and thus they will not be identified as relations based on the heuristic stated above. To address this issue, we first locate entities in the entity graph that have the cardinality of 1-to-n or m-to-n and consider them as specific relations. Notice that we assume that a relation should have at least two values and thus we do not consider entities with the cardinality of 1-to-1 or m-to-1 as relations. The rationale behind this consideration is two-fold. On the one hand, it is more natural and intuitive that a valid relation should have multiple distinct values [83]. On the other hand, it would narrow down the search space for valid relations when only relations that have multiple values are considered.

To more accurately identity valid relations presented in web pages and efficiently establish mappings from relations extracted from web pages to properties that are typically applied to a class or a set of classes defined in an ontology, we further abstract these specific relations by clustering them based on their labels, applied instances and values. Abstract relations are extracted from clusters that satisfy predefined criteria.

5.4.1 Clustering Specific Relations

Clustering algorithms come in various shapes and forms. They can be categorized on the basis of many different criteria [84]. For example, they can be categorized based on the structure of the cluster (hierarchical and partitional), based on the type and structure of the data (grid-based and categorical) and based on the size of the data set. The choice of clustering algorithm largely depends on the characteristics of the data set and the particular task.

Our clustering task is to group similar entities that are considered as specific relations based on their labels, ancestors and descendants. The number of clusters keeps changing during the process of clustering until no similar clusters can be merged. Based on these requirements, agglomerative clustering [85] fits our purpose. Intuitively, the agglomerative clustering algorithm starts with clusters each of which contains only one member. Then, the algorithm iteratively merges two existing clusters that are most similar to each other to form a new cluster. The algorithm terminates when there are no clusters that are similar enough to be merged.

In traditional agglomerative clustering algorithm, each cluster is represented by the centroid of its members and each centroid is typically the mean of its members calculated based on certain numerical measure. Such measure does not suit our clustering task, since each cluster member (i.e., entity representing a specific relation) in our relation-clustering task is represented by categorical data (i.e., label, domain and range). Therefore, we need an alternative way to calculate the centroid of cluster members. We formally define each cluster member as:

$$e = (\lambda_e, DS_e, RS_e)$$

λ_e is the label of entity e, DS_e is a set of ancestors of e in the entity graph and represents the domain of e, and RS_e is a set of descendants of e and represents the range of e. The definition of e conforms to that of a relation in Definition 5.4 since we treat e as a specific relation in this particular clustering task. Bellow, we formally define the centroid of members of a cluster:

Definition 5.5: A *centroid* of members of a cluster i is defined as

$$\mu_i = (\lambda_{\mu_i}, DS_{\mu_i}, RS_{\mu_i})$$

in which:

$$\lambda_{\mu_i} = LCS(\{\lambda_e | e \in cluster_i\})$$

$$DS_{\mu_i} = \bigcup_{e \in cluster_i} DS_e$$

$$RS_{\mu_i} = \bigcup_{e \in cluster_i} RS_e$$

The label of the centroid of a cluster is defined as the longest common sequence (LCS) [73] of labels of members in that cluster while the domain (or range) of the centroid of a cluster is defined as the union of domains (or ranges) of those members. The rationale behind Definition 5.5 is that the centroid of a cluster is a relation abstracted from the specific relations in that cluster (each member in a cluster represents a specific relation). Thus, the label of the abstract relation is the common name of those specific relations and the domain (or range) of the abstract relation contains all the members in the domains (or ranges) of those specific relations. The similarity between two centroids is a combination of similarities along three dimensions: label, domain and range, and it is defined as follow:

$$\begin{aligned} Sim^{\mu}(\mu_1, \mu_2) = {} & w_1 \cdot Sim^{L}_{hybrid}(\lambda_{\mu_1}, \lambda_{\mu_2}) + \\ & w_2 \cdot CS_{Sim^{L}_{hybrid}}(DS_{\mu_1}, DS_{\mu_2}) + \\ & w_3 \cdot CS_{Sim^{L}_{hybrid}}(RS_{\mu_1}, RS_{\mu_2}) \end{aligned} \quad (5.1)$$

where Sim^{L}_{hybrid} is the hybrid similarity measure defined in Definition 4.3 and it computes the label similarity between two entities. $CS_{Sim^{L}_{hybrid}}$ is the cross mapping score function defined in Definition 4.8 and it computes the similarity between two sets of entities. w_1, w_2 and w_3 are weights associated with each dimension and $w_1+w_2+w_3=1$. Following gives the algorithm of clustering entities that represent specific relations.

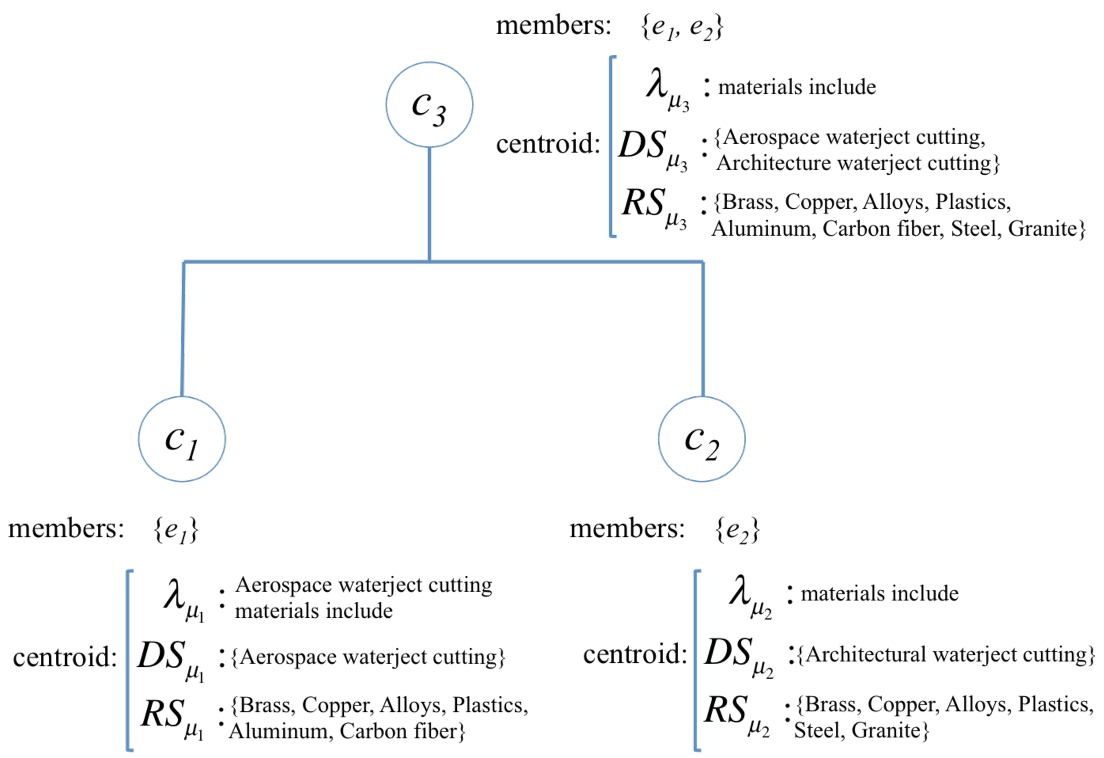
Algorithm 5.3: Given a list of entities *E* where each entity in *E* is a specific relation, the *relation agglomerative clustering algorithm* works as follows:

Step 1. Assign each entity in *E* to an empty cluster and add these clusters to the cluster set *C*.

Step 2. Search the cluster set *C* for two clusters, say c_i and c_j, that have the centroids most similar to each other among all the possible cluster pairs according to (5.1). If the similarity between c_i and c_j is beyond a predefined threshold, merge c_i and c_j as a new cluster c_z and compute the centroid μ_z of c_z. Remove c_i and c_j from *C* and add c_z to *C*.

Step 3. If only one cluster left in *C* or there were no cluster formed in *Step 2*, terminate the algorithm. Otherwise go back to *Step 2*.

For each cluster c_i in *C*, the centroid μ_i that satisfies the condition that $|DS_{\mu_i}| > 1$ and $|RS_{\mu_i}| > 1$ is considered as a relation r_i. Such relation r_i will be added to the relation set *R* of the instance description model (IDM) as well as the relation set R_{n_j} of each instance n_j belonging to the domain of r_i. All members in cluster c_i are specific variations of the abstract relation represented by r_i. In other words, all members in cluster c_i are actually the same relation as the abstract relation r_i but with more specific names. Thus, the label of each of those members will be revised to be consistent with the label of r_i. Figure 5.11 shows a simple example of merging two clusters c_1 and c_2 as a single cluster c_3.



members: $\{e_1, e_2\}$
c_3
centroid:
λ_{μ_3} : materials include
DS_{μ_3} : {Aerospace waterject cutting, Architecture waterject cutting}
RS_{μ_3} : {Brass, Copper, Alloys, Plastics, Aluminum, Carbon fiber, Steel, Granite}
c_1
c_2
members: $\{e_1\}$
centroid:
λ_{μ_1} : Aerospace waterject cutting materials include
DS_{μ_1} : {Aerospace waterject cutting}
RS_{μ_1} : {Brass, Copper, Alloys, Plastics, Aluminum, Carbon fiber}
members: $\{e_2\}$
centroid:
λ_{μ_2} : materials include
DS_{μ_2} : {Architectural waterject cutting}
RS_{μ_2} : {Brass, Copper, Alloys, Plastics, Steel, Granite}

Figure 5.11 Example of clustering potential relations

As shown in Figure 5.11, in this particular example, the clustering algorithm starts with two clusters c_1 and c_2 that each contains only one member and it ends up with a cluster c_3 merged from the two clusters. The label of the centroid of the merged cluster c_3 is **material include**, which is the longest common word sequence of the labels of centroids of clusters c_1 and c_2. The domain (or range) of the centroid of c_3 is the union of domains (ranges) of centroids of c_1 and c_2. Because the centroid of c_3 satisfy the condition that $|DS_{\mu_3}| > 1$ and $|RS_{\mu_3}| > 1$, **material include** is identified as a relation.

5.4.2 Data type Relation vs. Object Relation

In §5.4.1, we described a generic approach of clustering entities that are potential relations. However, according to Definition 5.4, there are two types of relations: data type relation and object relation. A data type relation has values that are numerical data types while an object relation has values that are instances. To make relations to be clustering compatible, we cluster the two types of relations separately. We differentiate relations by examining their values. Specifically, if above half of values of a relation *r* are numerical data values, *r* is considered a data type relation. Otherwise, it is an object relation. We used two tools – NERClassifierCombiner and RegexNERAnnotator – that are part of the coreNLP software package[15] developed by Stanford natural language processing group to determine whether a value of a relation is a numeral data type (or numerical description). NERClassifierCombiner is trained by built-in natural language corpora and it recognizes numerical entities using a rule-based system. RegexNERAnnotator implemented a rule-based named entity recognizer over token sequences using Java regular expressions. It allows us to develop RegexNER rules incorporating named entity labels that are not annotated in the built-in natural language corpora.

The simplest RegexNER rule has two tab-separated fields. The first field contains a sequence of one or more space-separated regular expressions. If the regular expressions match a sequence of tokens, the tokens will be labeled as the category specified in the second field. Following shows a simple RegexNER rule that checks whether a numeral description is of type inch:

```
(up|Up) to (^-*[0-9,\.]+$) (in|inch)      INCH
```

This rule contains four regular expressions. The first one checks if the first token of an inputted entity matches string of "up" or "Up", the second one checks if the second token exactly matches string of "to", the third regular expressions checks if the third token matches a number and the last one checks if the forth token matches string of "in" or "inch". "Up to 144 in" is an example of an entity that matches the rule presented above. Consequently, the entity is labeled as a numerical data type of INCH. Other possible numerical data types such as FEET, TON and LBS (i.e., pound) can be incorporated similarly into the RegexNER rules.

Relations grouped into each of the two types – data type relation and object relation – are clustered separately by applying Algorithm 5.3. Values of identified data type relations are labeled as numerical data values and added to the numerical data value set *N*, while values of identified object relations are labeled as instances and added to the instance set *I*.

[15] http://nlp.stanford.edu/software/corenlp.shtml

5.5 Identifying Instances and their Concepts

Having entities belonging to sets *R^o, R^d and N* been determined, the entities left in the entity graph are labeled as instances and added into set *I*. At this point, categories of all entities in entity graph have been settled. According to Definition 5.4, only three components– *A^o, A^d* and *K* – are left undetermined. *K* is a set of contexts of instances of IDM. Each instance's context contains two parts: a set of concepts and a set of relations. In §5.4.1, we explained how the set of relations of an instance is constructed. In this section, we explain the procedure of forming the concept set for an instance. In §5.6, we will form assertions *A^o* and *A^d* of IDM.

In §5.3, we briefly talked about how the concept set of an instance n_i is formed. Basically, we breadth-first navigate ancestors of instance n_i and record them as concepts of n_i based on the observation (§5.2) that ancestors of an instance typically have broader meaning than that instance. Each concept of n_i is associated with a score representing its relevance to n_i.

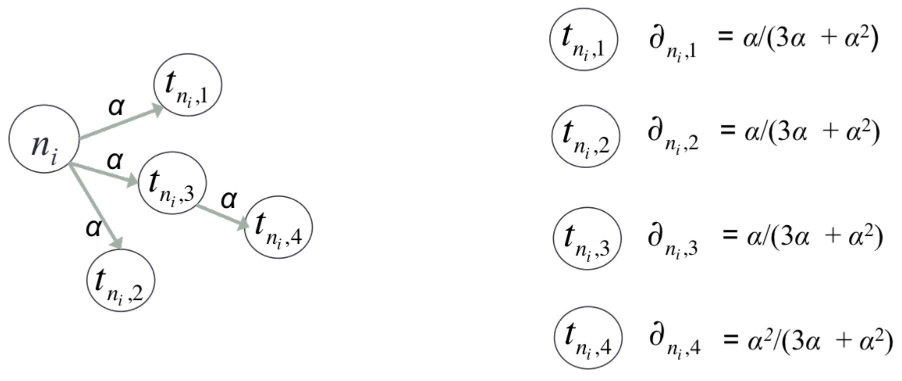
Algorithm 5.4: The algorithm of identifying concepts for instance n_i goes as follow:

Step 1. Identify n_i's ancestors at level *s* (i.e., *s* is the number of edge from these ancestors to n_i). Initially, set $s = 1$;

Step 2. For each ancestor *e* at level *s*, compute the relevance score α^s from e to n_i, where $\alpha \in (0,1]$ is the relevance decay factor; add *e* and its relevance score to the concept set T_{n_i}

Step 3. If the level of traversal reaches a predefined threshold (e.g., 3), terminate the traversal. Otherwise, set $s = s + 1$ and go to *Step 2*.

Figure 5.12 (a) shows an example of visited ancestors (i.e., concepts) of instance n_i. There are four distinct concepts for instance n_i. Figure 5.12 (b) shows the relevance score of each concept of instance n_i.


α
n_i
$t_{n_i,1}$
$t_{n_i,2}$
$t_{n_i,3}$
$t_{n_i,4}$
$\partial_{n_i,1} = \alpha/(3\alpha + \alpha^2)$
$\partial_{n_i,2} = \alpha/(3\alpha + \alpha^2)$
$\partial_{n_i,3} = \alpha/(3\alpha + \alpha^2)$
$\partial_{n_i,4} = \alpha^2/(3\alpha + \alpha^2)$

(a) visited ancestors (i.e., concept) of n_i. α is the relevance decay factor

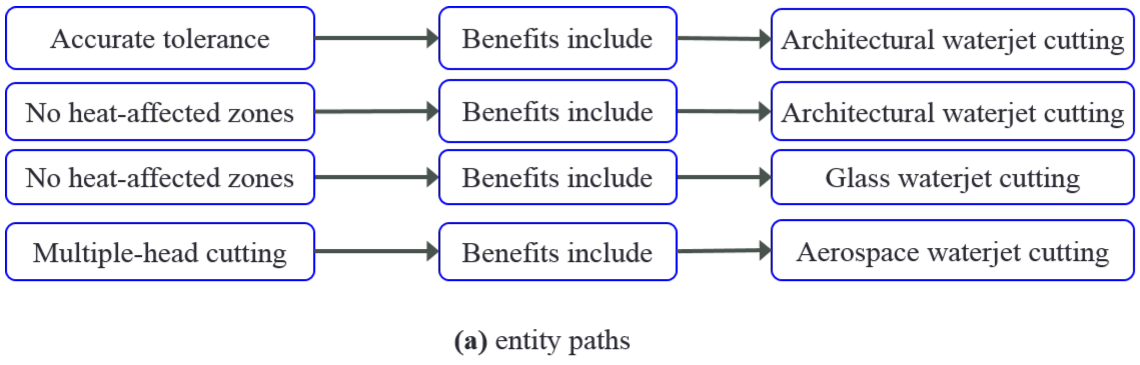
(b) relevance scores for merged concepts in the context of n_i

Figure 5.12 Example of forming context of instance n_i

For each instance n_i, the concept set T_{n_i} constructed in this section along with relation set R_{n_i} constructed in §5.4 form the context K_{n_i} of n_i. The context K_{n_i} helps determine the class label of n_i. In Chapter 6, we will map K_{n_i} of n_i to classes defined in the manufacturing domain ontology to locate a class label for n_i.

5.6 Extracting Assertions

As stated before, the objective of constructing an entity graph is to identify relations and form contexts of instances. In other words, it aims to identify the terminological knowledge of instances. However, the information of how specific instances are related via relations is lost when constructing entity graph. Such assertional knowledge can be found in entity paths. Figure 5.13 illustrates this point.



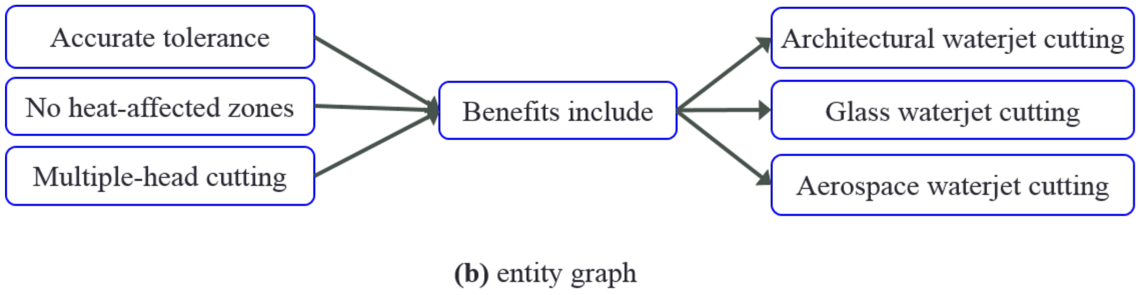


Figure 5.13 Example of entity path and entity graph presenting different level of knowledge published on web site

Figure 5.13 (b) shows the sub-entity-graph formed by entity paths shown in Figure 5.13 (a). Based on Algorithm 5.3, we consider **benefits include** as a relation that connect specific waterjet cuttings to benefits of these cuttings. However, from this sub-entity-graph, we cannot determine which benefit is related to which waterjet cutting via the relation **benefits include**. This information is encoded in the entity paths. For example, from the first entity path, we can tell that **Accurate tolerance** is a benefit of **Architectural waterject cutting**. Therefore, we need to navigate all entity paths to extract assertions. According to Definition 5.5, an assertion is a subject-predicate-object triple in which the subject is an instance, predicate is a relation and the object is an instance or numerical data value. Thus, we extract triples from entity paths and check whether the three parts of these triples satisfy the definition of an assertion.

Algorithm 5.5: Given an entity path *p*, the algorithm of extracting assertions from *p* goes as follow:

Step 1. Denote the index of entity in p as i and initialize index i to 0;

Step 2. Extract three consecutive entities e_i, e_{i+1} and e_{i+2} starting from the leaf node of p. Denote the triple <e_i, e_{i+1}, e_{i+2}> as a_i. If $e_i \in I$, $e_{i+1} \in R$ and $e_{i+2} \in I$ or N (I, R and N are the set of instances, the set of relations and the set of numerical data values of IDM respectively), add a_i to the assertion set A of IDM and increase index i by 2. Otherwise, increases index i by 1. go to step 3.

Step 3. If $i+2 <$ the number of entities in p, terminate the algorithm. Otherwise, go to *Step 2*.

Notice that in *Step 2* we increase index i by 2 when the triple <e_i, e_{i+1}, e_{i+2}> is an assertion. This is because if triple <e_i, e_{i+1}, e_{i+2}> is an assertion, triple <e_{i+1}, e_{i+2}, e_{i+3}> cannot be an assertion because the entity e_{i+1} is a relation which contradict the definition of assertion. In this case, we skip triple <e_{i+1}, e_{i+2}, e_{i+3}> and go two steps further to check whether the next triple in the entity path is an assertion. Figure 5.14 illustrates how assertions are extracted from an entity path given two identified relations — **Capability** and **Benefits include**.

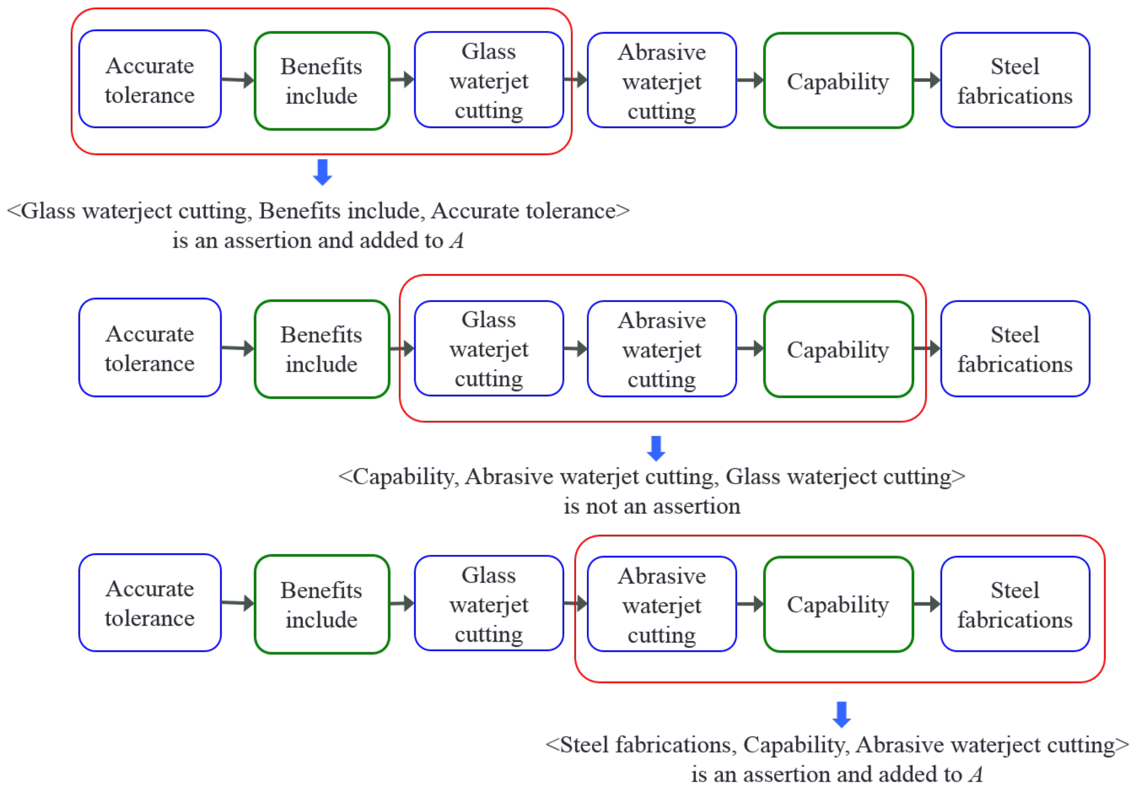

Accurate tolerance
Benefits include
Glass waterjet cutting
Abrasive waterjet cutting
Capability
Steel fabrications
<Glass waterjet cutting, Benefits include, Accurate tolerance> is an assertion and added to A
<Capability, Abrasive waterjet cutting, Glass waterject cutting> is not an assertion
<Steel fabrications, Capability, Abrasive waterject cutting> is an assertion and added to A

Figure 5.14 Example of extracting assertions from entity paths

As illustrated in Figure 5.14, the first triple <Glass waterject cutting, Benefits include, Accurate tolerance> is identified as an assertion and added to the assertion set *A* of the instance description model (IDM). Then, we go two steps further to check whether the triple <Capability, Abrasive waterject cutting, Glass waterject cutting> is an assertion and the answer is negative. Finally, we identify triple <Steel fabrications, Capability, Abrasive waterject cutting> as an assertion and add it to the assertion set *A*.

At this point, all the eight components of IDM – $I, R^o, R^d, N, D, K, A^o$ and A^d – have been formed and thus the construction of IDM of a manufacturing web site has been completed. The next step is to annotate instances of IDM with semantics defined in the manufacturing domain ontology (MDO) based on the contexts of these instances.

5.7 Summary

In this chapter, we elaborated the procedure of extracting manufacturing service capability instances from a manufacturing web site and forming an instance description model (IDM) to describe these instances. More specifically, this procedure first extracts entity paths from the web site and constructs entity graph based on these entity paths. Then it identifies relations that connect instances and forms the contexts of instances by analyzing the structure of entity graph. Finally, based on the relations identified, it extracts assertions on these instances. In next chapter, we will elaborate the approach of automatically annotating instances and relations with semantics defined in a MDO.

Chapter 6.

Annotating Manufacturing Service Capability Instances

In this chapter, we elaborate the approach of automatically annotating instances of instance description model (IDM) constructed from a manufacturing web site with semantics defined in manufacturing domain ontology (MDO). This process is the start of the semantic resolution. Specifically, it involves two subtasks: the first subtask is mapping relations of IDM to properties of MDO. The second task is identifying a class label from MDO for each instance of IDM based on the context of that instance. In §6.1, we first introduce the OWL-based manufacturing domain ontology (MDO) that defines common concepts (i.e., classes) and relationships (i.e., properties) between these concepts in the manufacturing domain. §6.2 describes in detail how correspondences between components of IDM and those of the MDO are established. Based on those correspondences, instances of IDM are annotated with classes defined in MDO. The experiment results are presented in §6.3.

6.1 Manufacturing Domain Ontology

In this section, we introduce the manufacturing domain ontology we leverage to annotate and integrate manufacturing service capability (MSC) instances extracted from manufacturing web sites.

A great number of ontologies have been developed and published along with the development of semantic web for different domains and purposes. Only a very few of them are developed for the manufacturing domains. These few include for example GoodRelations, an upper ontology for e-commerce, and MSDL (Manufacturing

Service Description Language) [86], a detailed ontology for manufacturing capabilities, especially for machining services.

We developed a manufacturing domain ontology (MDO) based on MSDL and GoodRelations for experimental use in this research. We also integrated some concepts reverse-engineered from popular manufacturing e-marketplace portals, including mfg.com, thomasnet.com[16] and globalspec.com[17] into MDO for better coverage of core concepts of manufacturing domains. MDO (1) covers the most common and important concepts of manufacturing domain, (2) is easy to use (for application), reusable, extensible, and conceptually compatible with other relevant ontologies, and (3) functions as a shared conceptualization of the manufacturing domain for resolving semantic difference among data from heterogeneous sources. It defines 845 classes and 22 of them are top classes, and each top class corresponds to a class hierarchy (§6.2.2). The core classes and properties of MDO for describing manufacturing service capabilities are shown in Figure 6.1.

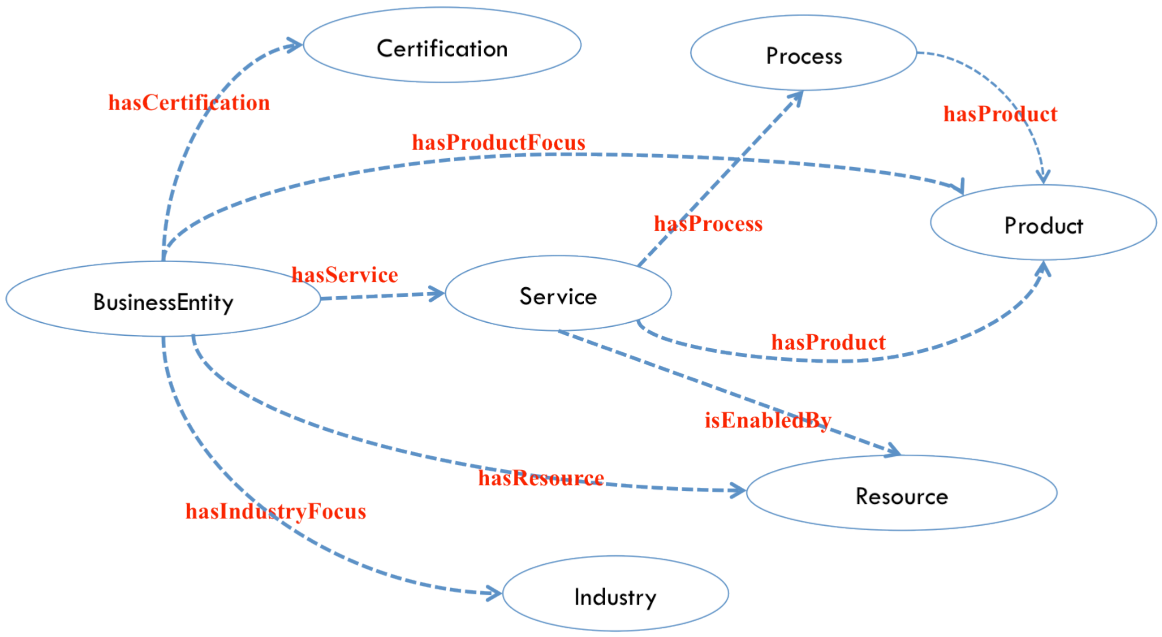


Figure 6.1 Core classes and properties of manufacturing domain ontology

16 www.thomasnet.com

17 www.globalspec.com

The MDO is written in OWL that is a combination of RDF data model and a dialect of description logic [18]. The primary advantage of RDF is that it facilitates the integration of data from heterogeneous sources, while the description logic allows one to perform logical reasoning over the ontology and its instances. Since the extracted instances are described by IDM, which is a local semantic model, in a structural way, the annotation of instances of IDM is mainly performed by exploiting the mappings established between components of IDM and those of MDO as well as the structure of those components. The main advantage of having this local semantic model (i.e., IDM) is that the MSC instances extracted from web pages can be easily and accurately annotated by (partially) applying ontology mapping techniques. We define the structure of a given manufacturing domain ontology as follows:

Definition 6.1: The structure of a given *manufacturing domain ontology* is a tuple <*C, SR, OI, P^o, P^d, U, V*>, where:

$C = \{c_q\}$ represents a set of classes. These classes are arranged with a corresponding subsumption hierarchy: $SR = \left\{(c_1, c_2) \middle| c_1 \in C, c_2 \in C, c_1 \geq_{\sup} c_2\right\}$, which is a set of 2-tuple that each represents the subclass relationship between a pair of classes. $c_1 \geq_{\sup} c_2$ denotes that c_2 is a subclass of c_1

$OI = \bigcup_{c_q \in C} OI_{c_q}$, the set of ontology instances of MDO, where $OI_{c_q} = \left\{n_{c_q,j}\right\}$ is the set of instances belong to class c_q

$P^o = \{p_y\}$, a set of object properties that each connects two (related) instances with $p_y = \left\{(s,o) \middle| s \in OI_{c_1}, c_1 \in DS_{p_y}; o \in OI_{c_2}, c_2 \in RS_{p_y}\right\}$, $DS_{p_y} \subseteq C$ is the domain of p_y and $RS_{p_y} \subseteq C$ is the range of p_y. The label of p_y is denoted as λ_{p_y}.

$P^d = \{p_z\}$, a set of data type properties that each connects an instance with a data value with $p_z = \left\{(s,o) \middle| s \in OI_{c_1}, c_1 \in DS_{p_z}; o \in V, data\ type\ of\ o \in RS_{p_z}\right\}$, $DS_{p_z} \subseteq C$ is the domain of p_z and $RS_{p_z} \subseteq U$ is the range of p_z, where V is the set of all data values of data type properties in P^d and U is the set of primitive data types that can be assigned to values of data type properties. The label of

p_z is denoted as λ_{p_z}.

$P = P^o \cup P^d$, the set of all properties

The assertional knowledge of manufacturing domain ontology is defined below. Having been annotated, instances of IDM were transformed as instances of MDO. Consequently, assertions of IDM were integrated into the MDO.

Definition 6.2: Given a manufacturing domain ontology, the *assertions* of this ontology is a 2-tuple <OA^o, OA^d>, where:

$OA^o = \{(s, p^o, o) | p^o \in P^o, s \in OI, o \in OI\}$, the set of assertions in the form of subject-predicate-object triple, in which the predicate is an object property

$OA^d = \{(s, p^d, o) | p^d \in P^d, s \in OI, o \in V\}$, the set of assertions in the form of subject-predicate-object triple, in which the predicate is a data type property

$OA = OA^o \cup OA^d$, the set of all assertions

The definition of the MDO and that of the IDM are analogous (Definitions 5.4 and 5.5), since our objective of developing IDM is to model manufacturing service capabilities published in manufacturing web sites in a structural way that it can facilitate the annotation of these capabilities with semantics defined in MDO. Defining the two models in similar way helps achieve such objective. The main difference between the two models is that MDO explicitly defines the type information of instances through classes as well as domains and ranges of properties while IDM represents the type information of an instance by a set of terms related to that instance (i.e., context). This is because MDO is written by formal ontology language designed for explicit knowledge representation and meticulously developed by ontology engineers while IDM is automatically constructed from web pages that are mainly for visually consumption by humans rather than machine to process. In Table 6.1, we present the correspondences between components of MDO and those of IDM.

Table 6.1 The comparison between components in MDO and IDM

MDO	IDM	Description
C	*K*	Each member of *C* is a formally defined class while each member of *K* is the context describing the meaning of an instance n_i . The main objective of the instance annotation is to identify a class from *C* as the class label for n_i based on the context of n_i.
OI	*I*	Each member of the two sets refers to an instance. One subtask of the instance annotation is to transform members of *I* as members of *OI* by assigning each member a class label defined in MDO.
P^o	R^o	Each member of the two sets is a relation that relates an instance to another. In general, domains and ranges of members of P^o are formally defined as a set of classes while those of members of R^o are interpreted as a set of instances. Mapping each member of R^o to a member of P^o is another subtask of the instance annotation
P^d	R^d	Each member of the two sets is a relation that relates an instance to a primitive data value. Ranges of members of P^o are defined as a set of primitive data types while ranges of members of R^o are defined as a set of numerical data types. Mapping each member of R^o to a member of P^o is another subtask of the instance annotation
U	*D*	Each member of set *U* is a predefined primitive data type while each member of set *D* is a numerical data type determined based on members in *N*.
V	*N*	Each member of set *V* is a primitive data value while each member of set *N* is a numerical data value. In IDM, data values that are not numerical types are treated as instances.
OA^o	A^o	Each member of the two sets is a triple in which a subject is related to an object through a relation or a property. Both subject and object are instances
OA^d	A^d	Each member of the two sets is a triple in which a subject is related to an object through a relation or a property. Subject of each triple is an instance while object is a primitive data value.
SR		*SR* is a set of subsumption relationships defined between members (i.e., classes) of *C*. We will exploit *SR* to help identify class labels for instances of IDM. Because IDM is constructed from web pages, no subsumption relationship

		between entities in IDM was defined.

Our approach for annotating IDM with semantics of MDO is given in the next section.

6.2 Semantic Annotation of the Instance Description Model

The objective of semantically annotating instances of instance description model (IDM) is to describe these instances with vocabularies (i.e., classes and properties) formally defined in the manufacturing domain ontology (MDO). To this end, three tasks are involved:

- Mapping relations in R of IDM to properties in P of MDO
- Mapping concepts in context K_{n_i} of each instance n_i of IDM to a set of classes C_{n_i} defined in MDO.
- Identifying a class label for instance n_i based on C_{n_i}.

For Task 1, we map each relation r of IDM to a property p of MDO. We separate the establishment of mappings between relations and properties from the establishment of mappings between context of each instance and classes defined in MDO. This is because relations connect instances (Definition 5.4) while properties are applied to classes (Definition 6.1). In order to more accurately map a relation and a property, we need to leverage relation's domain (and/or range) that is defined as a set of instances rather than a specific instance. Thus, we consider relation as a separate component rather than a member in the context of a particular instance when establishing mappings. After the mappings have been established, the classes in the domain of p will be added as new concepts to the concept sets of instances belonging to DS_r, the domain of r. This added semantic information will help improve the accuracy of identifying class labels for those instances. Task 2 maps concepts of each particular instance n_i to a set of classes defined in MDO. Based on these mappings, in Task 3, we then identify the class label of instance n_i. We will explain Task 1 in §6.2.1 and Task 2 and 3 in §6.2.2. The result of each of the three tasks affects the result of the other. To achieve a global optimal result of the semantic annotation, the three mapping tasks should be considered as whole. In our case, however, fast computing time is preferred over global optimal matching result. For one thing, the mappings are typically performed at real time. For another, incorrect mappings can be corrected when needed by using semantic resolution knowledge base as will be discussed in Chapter 7. Therefore, we adopt a greedy approach that solves the three mapping tasks sequentially.

6.2.1 Annotating Relations of IDM

Given a set of relation R of an instance description model (IDM) and a set of properties P (both P^o and P^d) of the manufacturing domain ontology (MDO), the semantic annotation of relations of IDM is the problem of finding a counterpart for each $r \in R$ in P. Each $r \in R$ maps to at most one $p \in P$ while each $p \in P$ can be mapped by multiple relations from R. The counterpart (i.e., property) of relation $r \in R$ in P can be identified by applying the entity mapping function defined in Definition 4.4 (§4.2).

$$m_{Sim^R}(r, P) = \underset{p \in P}{\operatorname{argmax}}\, Sim^R(r, p) \tag{6.1}$$

where $Sim^R(r, p)$ is the similarity measure that computes the similarity between a relation r and a property p. Similar to formula (5.1) that computes similarity between two centroids of two clusters of relations as discussed in §5.4.1, we compare r and p along three dimensions: labels, domains and ranges between r and p:

$$\begin{aligned} Sim^R(r, p) = {} & w_1 \cdot Sim^L_{hybrid}(\lambda_r, \lambda_p) + \\ & w_2 \cdot MS^A_{Sim^L_{hybrid}}(DS_r, DS_p) + \\ & w_3 \cdot MS^A_{Sim^L_{hybrid}}(RS_r, RS_p) \end{aligned} \tag{6.2}$$

Where λ_r and λ_p are labels of r and p respectively; DS_r and DS_p are domains of r and p respectively; RS_r and RS_p are ranges of r and p respectively. We compute similarity between λ_r and λ_p by applying the hybrid label similarity measure defined in Definition 4.3 (§4.1). Similarity between domains (or ranges) of r and p is computed by applying the average mapping score function defined in Definition 4.6 (§4.2). The set of mappings from all relations in R to properties in P is established by applying the entity-set mapping function $M_{Sim^R}(R, P)$ defined in Definition 4.5 (§4.2).

One issue with (6.2) of computing similarity between r and p is that the IDM represents the domain of a relation as a set of instances, while the MDO represents the domain of a property as a set of classes (typically one or two classes). This may cause granularity heterogeneity because IDM may provide a more detailed description of a domain than MDO does. To address this issue, when establishing mappings between members in DS_r and those in DS_p, we extend domain DS_p that originally contains a set of classes C to a set of terms that involves subclasses and instances of classes in C:

$$DS_p \leftarrow DS_p \cup \left\{ n_i \middle| n_i \in OI_{c_q} \text{ and } c_q \in DS_p \right\} \cup \left\{ c_j \middle| (c_q, c_j) \in SR \text{ and } c_q \in DS_p \right\}$$

where SR is the subsumption hierarchy defined in Definition 6.1. The reason that we consider subclasses and instances of classes in C as part of the domain of a property p is twofold: (1) the original classes contained in C may be insufficient to describe the domain of p in the context of comparing domains between relations of IDM and properties of MDO; (2) subclasses and instances are useful descriptions of the domain of p. The same extension process is also applied to the ranges of object type properties.

6.2.2 Annotating Instances of IDM

Given a set of instances I of instance description model (IDM) where each instance n_i is associated with a context $K_{n_i} = (R_{n_i}, T_{n_i})$ and a set of classes C defined in the manufacturing domain ontology (MDO), the instance annotation is the problem of identifying a class label $c \in C$ for each instance $n_i \in I$ based on K_{n_i}. We decompose this problem into two steps: (1) identifying the top class hierarchy closest to n_i based on K_{n_i}. The identified top class hierarchy is denoted as H_k. (2) Choosing the class label for n_i from H_k. These two steps are described in detail in the following.

6.2.2.1 Identifying Class Hierarchy for Instance

The first step of identifying a class label from the MDO for an incoming instance n_i is identifying the top class hierarchy that is closest to instance n_i based on context K_{n_i} of n_i. The basic idea behind this process is that we first narrow down the search space to the area where the correct class label for instance n_i most likely resides in. A top class hierarchy, as we will define shortly, is a set of classes that describes a specific category of the MDO. Therefore, identifying the top class hierarchy closest to instance n_i is equivalent to identifying the sub-ontology of MDO that is specific to describing the category that instance n_i most likely belongs to. We formally define the top class and top class hierarchy as follow:

Definition 6.3: Given the set of classes C of an OWL-based manufacturing domain ontology O, A class $g \in C$ is a *top class* if and only if there exists no other class in C that is superclass of g.

Note that the class set C stated in Definition 6.2 contains only domain specific classes created by manufacturing domain experts. It does not include built-in classes (i.e., *Thing* and *Nothing*) of OWL-based ontologies.

Definition 6.4: A *top class hierarchy* H_j of a top class g, is a rooted graph denoted as:

$$H_j = (g, \Omega, S),$$

where g is the root of H_j , Ω is the set of all subclasses of g and S is a finite set of 2-tuples that each represents a subclass relationship between two classes in Ω:

$$S = \{(c_1, c_2) | c_1 \in \Omega, c_2 \in \Omega, c_1 \geq_{\sup} c_2\}$$

The closeness of an instance n_i to a top class hierarchy H_j is measured based on the label similarities between concepts in the concept set T_{n_i} of n_i and classes in H_j , as well as the relevance score $\partial_{n_i,q}$ of each concept $t_{n_i,q}$ in T_{n_i} to instance n_i. Concepts in the T_{n_i} play a crucial role in measuring the closeness from n_i to H_j . However, since the concept set T_{n_i} is extracted from the web site that instance n_i resides in, it is likely that some of, or even most of, the concepts in T_{n_i} cannot find counterparts (i.e., classes) in MDO because of the terminological or conceptual heterogeneity. To improve the chances that concepts in T_{n_i} can be mapped to classes in MDO, we extend the T_{n_i} to include the classes in domains of properties that has been mapped to relations in the relation set R_{n_i} of instance n_i:

$$T_{n_i} \leftarrow T_{n_i} \cup \left\{ (c_j, \partial_j) \middle| r_{n_i,q} \in R_{n_i}, (r_{n_i,q}, p) \in M(R,P) \text{ and } c_j \in DS_p \right\}$$

where $M(R,P)$ is the set of mappings established between relations in R of IDM and properties in P of MDO according to (6.1). ∂_j in (c_i, ∂_j) is the relevance score of c_i to instance n_i. To identify the class hierarchy that is closest to instance n_i, we first map each concept $t_{n_i,q}$ in T_{n_i} to a class in MDO that is closest to $t_{n_i,q}$ by (6.3)

$$c_{j,z} = \max_{c \in C} Sim^{L}_{hybrid}(t_{n_i,q}, c) \tag{6.3}$$

where $Sim^{L}_{hybrid}(t_{n_i,q}, c)$ measures the similarity between labels of concept $t_{n_i,q}$ and class c. The class $c_{j,z}$ is the closest class among all the classes C defined in MDO to $t_{n_i,q}$ and it is the zth class in class hierarchy H_j. Based on (6.3), we denote the set of mapping pairs established between concepts in T_{n_i} and classes in H_j as follow.

$$HM_j = \left\{ \begin{array}{l} (t_{n_i,q}, c_{j,z}) \middle| t_{n_i,q} \in T_{n_i}, c_{j,z} = \max_{c \in C} Sim^{L}_{hybrid}(t_{n_i,q}, c) \\ \qquad\qquad\qquad\qquad \text{for } Sim^{L}_{hybrid}(t_{n_i,q}, c) > d \text{ and } c_{j,z} \in H_j \end{array} \right\} \tag{6.4}$$

Note that we consider concept $t_{n_i,q}$ and class $c_{j,z}$ are mapped only when their similarity is beyond the threshold d. We denote H as the set of class hierarchies that each includes one or more class mapped to concepts in T_{n_i}. Then, for each class hierarchy H_j in H, we compute the closeness score between T_{n_i} and H_j by (6.5).

$$closeness\,(T_{n_i}, H_j) = \sum_{(t_{n_i,q}, c_{j,z}) \in HM_j} \partial_{n_i,q} \cdot Sim^{L}_{hybrid}(t_{n_i,q}, c_{j,z}) \tag{6.5}$$

The top class hierarchy that is closest to n_i is computed as follow:

$$H_k = \underset{H_j \in H}{\operatorname{argmax}} \, closeness(T_{n_i}, H_j) \tag{6.6}$$

Figure 6.2 shows an illustrative example on identifying the closest top class hierarchy for instance n_i. The three blue circles are concepts from T_{n_i} of instance n_i and each concept $t_{n_i,q}$ is associated with a relevance score $\partial_{n_i,q}$ to n_i. The two graphs composed of green circles are top class hierarchies defined in MDO. The label similarities between each concept in T_{n_i} and a class in one of the two top class hierarchies are given to the right of two hierarchies. The two sets of mapping pairs (i.e., HM_1 and HM_2) are also given on the right of Figure 6.2. According to (6.5), the closeness from T_{n_i} to H_1 is 0.36, to H_2 is 0.64. Thus, closest top hierarchy to instance n_i is H_2 according to (6.6).

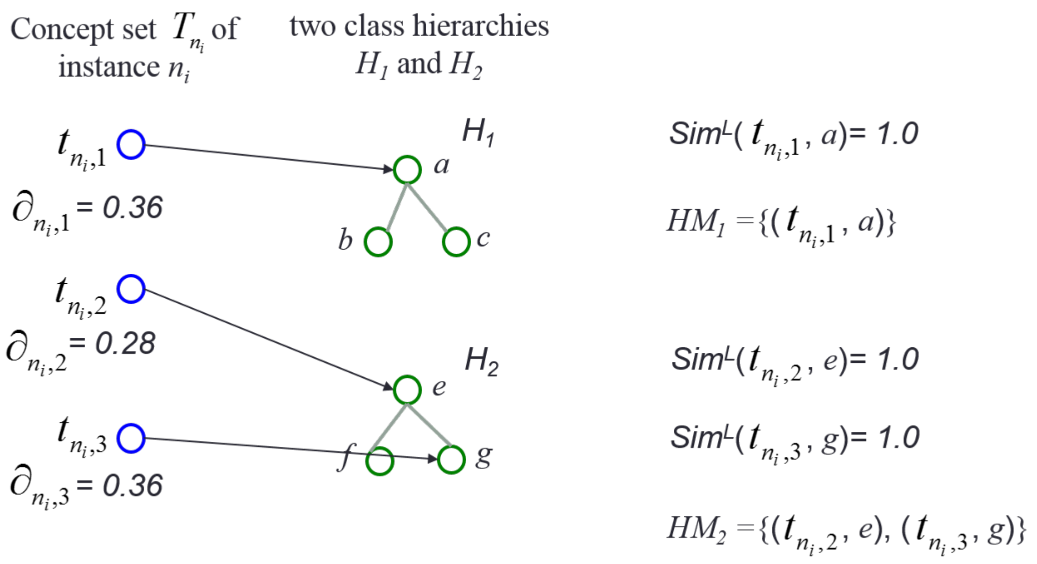

Concept set T_{n_i} of instance n_i
two class hierarchies H_1 and H_2
$t_{n_i,1}$
$\partial_{n_i,1}$ = 0.36
$t_{n_i,2}$
$\partial_{n_i,2}$ = 0.28
$t_{n_i,3}$
$\partial_{n_i,3}$ = 0.36
H_1
a
b
c
H_2
e
f
g
$Sim^L(t_{n_i,1}, a)$= 1.0
$HM_1 = \{(t_{n_i,1}, a)\}$
$Sim^L(t_{n_i,2}, e)$= 1.0
$Sim^L(t_{n_i,3}, g)$= 1.0
$HM_2 = \{(t_{n_i,2}, e), (t_{n_i,3}, g)\}$

Figure 6.2 An example of identifying the closest top class hierarchy for instance n_i

After the top class hierarchy H_k that is closest to instance n_i has been identified, we then choose a class label from H_k for instance n_i.

6.2.2.2 Identifying Class Label for Instance

We choose a class label from H_k for an instance n_i based on the classes in H_k that have been mapped to concepts in T_{n_i} of n_i. In other words, the class label of n_i is chosen from HM_k established in (6.4). This HM_k serves as a translator that translates the description of instance n_i in terms of a set of manufacturing concepts expressed in manufacturing web site to the description of n_i in terms of a set of classes defined in MDO. The set of mapped classes for describing instance n_i is denoted as:

$$C_{n_i} = \left\{ c_{j,z} \middle| (t_{n_i,q}, c_{j,z}) \in HM_k \right\} \tag{6.7}$$

We choose a class c from H_k that best represents the meaning of C_{n_i} as the class label for instance n_i. Such class c is called the *best representative class* of C_{n_i} and is chosen based on two criteria: (1) the best representative class should be able to resolve the *semantic conflicts* that occurs when two or more classes in C_{n_i} have no subsumption relationships among them. (2) Among classes that satisfy the first criteria, denoted as C'_{n_i}, the most specific class in terms of the subsumption relationships defined in H_k will be chosen as the class for instance n_i. This is because the most specific class carries the most semantics among classes in C'_{n_i}. Choosing this class as the class label for n_i, we maintain the most possible semantics represented by C_{n_i} of n_i. We develop the algorithm for solving this problem based on two scenarios that each corresponds to a way of finding the best representative of C_{n_i} .

The first scenario, as shown in Figure 6.3, occurs when there exists a *convergence* among classes in C_{n_i}. More specifically, there exists a class $k \in C_{n_i}$ such that it is the subclass of all other classes in C_{n_i}. In this case, class k inherits semantics of all members in C_{n_i} and thus it is chosen as the best representative class of C_{n_i}.

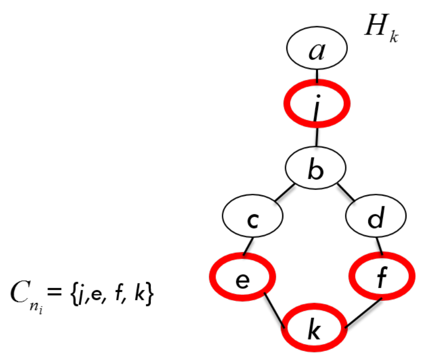


Figure 6.3 The first scenario of identifying best representative of C_{n_i}. In this scenario, C_{n_i} = {*j, e, f, k*} and the best representative of C_{n_i} is class *k*.

If such a class does not exist, as illustrated in Figure 6.4, two or more classes in C_{n_i} diverge from their common parent. In other words, there exists a *divergence* among classes in C_{n_i}. In this case, semantic conflict occurs among divergent classes. The semantic conflict caused by diverging classes (e.g., *e* and *f*) can be resolved by finding common ancestors (e.g., *a*, *j* and *b*) of these classes. This is because common ancestors of diverging classes define semantics shared by those divergent classes. Put another way, instances of divergent classes that have common ancestors are also instances of these ancestors. The lowest common ancestor (e.g., *b*) among all common ancestors (in other words, the lowest common subsumer of divergent classes) is chosen as the best representative class of C_{n_i} since the lowest common ancestor maintains the most possible semantics of those divergent classes.

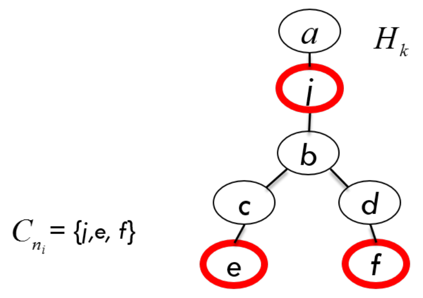


Figure 6.4 The second scenarios of identifying best representative of C_{n_i}. In this scenario, C_{n_i} = {*j, e, f* } and the best representative of C_{n_i} is *b*.

Algorithm 6.1: Given a class set C_{n_i} containing classes mapped by concepts of n_i, the algorithm of identifying the best representative class of C_{n_i} works as follow:

Step 1. Identify the class *k* that is the subclass of all the members in C_{n_i} . If such class *k* exists, *k* is chosen as the class label of n_i and terminate the algorithm. Otherwise, identify divergent classes among members in C_{n_i} and store these classes to an empty class set *F*, and go to *Step 2*.

Step 2. Identify common ancestors of members in *F*. Chose the lowest one as the class label of n_i and end the algorithm.

The algorithm first checks if C_{n_i} contains class that is the subclass of all its members. If there exists no such class, C_{n_i} contains divergent classes. Thus, we identify common ancestors of these divergent classes in order to resolve the semantic conflict. The lowest one is chosen as the class label of instance n_i.

6.3 Experimental Evaluation

We randomly selected 45 manufacturing web sites from Thomasnet to evaluate the quality of the contexts formed for manufacturing service capability (MSC) instances extracted from these web sites and the effectiveness of our approach for automatically annotating these instances based on their contexts. We did not evaluate if we extracted all instances from each manufacturing web site considered. This is mainly because (1) our major objective is not to extract all instances but as many as possible and accurately annotate these instances with classes defined in the MDO and (2) the evaluation of whether all the instances of a manufacturing web site have been extracted is extremely time-consuming and labor-intensive since we need to go over each web page of the manufacturing web sites to check whether all instances were extracted. We leave a thorough evaluation of this part to future research and did a simple sampling to get a sense of how many instances can be extracted. This was done by selecting 5 pages for each of the 45 web sites and compute the ratio of instances (including relations) extracted from these pages to all instances of interest presented in these pages determined by human. The ratio obtained is 0.76, which demonstrates that the instance extractor works fairly well in extracting MSC instances.

To evaluate our approach for automatically annotating extracted manufacturing service capability instances, we created the ground truth by manually annotating 14894 instances of interest extracted from 45 web sites with classes defined in MDO. The correctness of these annotated instances is then validated by myself and Yunsu Lee, who is a researcher at the National Institute of Standards and Technology (NIST) and familiar with both the manufacturing domain and OWL-based ontology. We adopt the relaxed version of the precision and recall [87] to evaluate our instance annotation approach since the traditional ones cannot discriminate between a totally wrong annotation and an almost correct annotation. As a result, they are not able to accurately reflect the effectiveness of our approach. [87] defines the relaxed precision and relaxed recall as follows:

$$\text{relaxed precision} = \frac{\sum_{i=1}^{m} \omega(EC_{n_i}, RC_{n_i})}{m},$$

$$\text{relaxed recall} = \frac{\sum_{i=1}^{m} \omega(EC_{n_i}, RC_{n_i})}{n} \tag{6.8}$$

m is the number of instances assigned class labels, and n is the number of instances defined in the ground truth. EC_{n_i} is the expected class (defined in the ground truth) of instance n_i, while RC_{n_i} is the assigned class (assigned by instance annotator) of instance n_i. The $\omega(EC_{n_i}, RC_{n_i})$ measures the proximity between the expected class and real class of n_i, and it is defined as follow:

$$\omega(EC_{n_i}, RC_{n_i}) = \begin{cases} \beta^{\sigma(EC_{n_i}, RC_{n_i})} & \text{if } EC_{n_i} \text{ and } RC_{n_i} \text{ have subsumtion relationship} \\ 0 & \text{otherwise} \end{cases} \tag{6.9}$$

β is set to 0.8 if EC_{n_i} is superclass of RC_{n_i}, otherwise set to 0.6. The reason we assign higher score to β for the first condition is because that an instance n_i of type RC_{n_i} is also an instance of type EC_{n_i} and thus it is still correct when we classify n_i to EC_{n_i} rather than RC_{n_i}. If EC_{n_i} is subclass of RC_{n_i}, an instance n_i of type RC_{n_i} is not necessarily an instance of type EC_{n_i}. Thus, the class label EC_{n_i} assigned to instance n_i by the instance annotator is not strictly correct. However, we still give credits for EC_{n_i} and RC_{n_i}being in the same class path. $\sigma(EC_{n_i}, RC_{n_i})$ is the number of edges in the shortest path between EC_{n_i} and RC_{n_i}. The experimental results are given in Table 6.2.

Table 6.2 Experimental results for Instance Annotator

Web Site	Pre.	Rec.	F_1	Miss
Schulermfg	0.78	0.59	0.67	0.25
Soundmfg	0.76	0.59	0.66	0.23
numericalconcepts	0.8	0.7	0.75	0.13
Basssettinc	0.84	0.64	0.72	0.24
Aerostarmfg	0.91	0.79	0.71	0.22
magnamachine	0.79	0.45	0.57	0.43
Ashleyward	0.93	0.55	0.69	0.32
Accutrex	0.94	0.82	0.88	0.12
Weaverandsons	0.9	0.73	0.831	0.19
Astromfg	0.71	0.59	0.65	0.16
Wisconsinmetalparts	0.77	0.69	0.73	0.10
Tmfincorporated	0.92	0.83	0.87	0.10
Robersontool	0.83	0.77	0.8	0.08
Spmfg	0.96	0.76	0.85	0.21
Smccontractmfg	0.79	0.65	0.71	0.18
Swiftglass	0.96	0.88	0.92	0.09
Ameristarmfg	0.94	0.89	0.91	0.06
Jnmetalproducts	0.91	0.82	0.86	0.10
Nobleindustries	0.85	0.73	0.78	0.14
Aalloy	0.92	0.89	0.91	0.04
Portersfab	0.93	0.84	0.88	0.10
Californiabrazing	0.94	0.78	0.85	0.13
Columbiamanufacturing	0.91	0.75	0.82	0.17
Mantecservicesinc	0.82	0.7	0.76	0.16
Kilgoremfg	0.93	0.63	0.75	0.32
Elgeprecision	0.97	0.59	0.74	0.39
Dynamicprecision	0.94	0.76	0.84	0.19
Pbandm	0.82	0.67	0.74	0.18
Schenketool	0.89	0.85	0.87	0.05
Pierceindustries	0.87	0.6	0.71	0.39
Westfieldmachine	0.99	0.6	0.75	0.28

Ardelengineering	0.83	0.6	0.72	0.18
Hhsmm	0.66	0.55	0.6	0.17
Wiegeltoolworks	0.91	0.77	0.84	0.16
Adaptplastics	0.89	0.64	0.74	0.28
Mpi-dms	0.95	0.78	0.86	0.18
Contract-mfg	0.89	0.71	0.79	0.20
Hloeb	0.94	0.82	0.87	0.13
Milacronmachining	0.78	0.56	0.65	0.26
cuttingexperts	0.82	0.42	0.55	0.48
hds-usa	0.82	0.57	0.67	0.30
madisontoolinc	0.86	0.75	0.8	0.12
Duratrack	0.57	0.43	0.49	0.24
nationgrinding	0.92	0.8	0.86	0.13
Schmidtool	0.82	0.74	0.78	0.11
Average	**0.86**	**0.69**	**0.76**	**0.19**

The first column of Table 6.2 lists the web sites of manufacturers. In each site, more than 50 web pages were crawled and on average 331 instances were extracted from each site. The column 2, 3, and 4 give the numbers of relaxed precision, relaxed recall and F_1 measure respectively. The last column gives the ratio of the numbers of instances that were not annotated (assigned a class label) to the numbers of instances that were extracted. We call such ratio the *miss rate* since it measures the percentage of instances that were not assigned a class label. The relaxed precision measures the accuracy of our approach assigning class labels to instances that were annotated, while the relaxed recall measures the accuracy of our approach assigning class labels to all instances that were extracted (including instances that were annotated and unannotated). As shown in the second column, the 87% relaxed precision demonstrates that our approach is quite effective for accurately assigning class labels to instances provided that the concepts in the contexts of these instances can be mapped to classes defined in MDO (§6.2). However, the 69% relaxed recall shows that our approach has difficulty in assigning class labels to all instances extracted. This mainly because: (1) some MSC terms or concepts published on manufacturing web sites is not covered by the MDO, and (2) our label similarity measure is insufficient to map all concepts in contexts of instances to classes defined in MDO typically because of the terminological heterogeneity. The 0.19 miss rate also demonstrates this point.

6.4 Summary

In this chapter, we introduced the manufacturing domain ontology (MDO) and described in detail the approach of automatically annotating instances and relations of the instance description model (IDM) with semantics defined in the MDO. Specifically, we mapped each relation of IDM to a property defined in MDO and identified a class label from MDO for each instance of IDM. The experiment result shows that our approach is quite effective in accurately annotating instances but still has potential for improvement. To further improve the annotation quality, we developed an approach that utilizes domain knowledge learned from both the human experience and the automatic instance annotation process and stored in a semantic resolution knowledge base (SR-KB). In Chapter 7, we will discuss this SR-KB in detail.

Chapter 7.

Semantic Resolution Knowledge Base

The instance annotator based on the approach described in Chapter 6 can automatically assign class labels to manufacturing service capability instances with reasonably high accuracy as demonstrated by the experimental results. However, it has the following limitations:

- It has problem in dealing with terminological heterogeneity [75] when mapping concepts in the context of an instance n_i to classes defined in manufacturing domain ontology (MDO). This is because the label similarity measures used to map concepts and classes utilize WordNet to map synonyms (or words with similar senses) existing in labels of concepts and classes. However, WordNet is a general-purpose lexicon and it contains only the most commonly used words, and synonyms. Thus, it is unable to map synonyms specific to the manufacturing domain such as “capability” to “service” and “shearing” to “die cutting”. The WordNet also cannot map domain-specific phrases to their corresponding abbreviations (and vice versa), such as “electrical discharge machining” to “EDM”, “computer-aided manufacturing” to “CAM” and “metal inert gas” to “MIG”.
- It is insufficient to address the issue of conceptual heterogeneity especially when a manufacturing web site describes certain portion of the manufacturing domain that is not covered by the MDO or it describes certain manufacturing domain concepts that are defined in the MDO but with different perspectives. For example, some manufacturing web sites treat their services as product, while some others treat their services as capability.
- It lacks learning ability that can avoid the same mistakes made before and prevent similar mistakes from happening in the future.

These limitations can be addressed with the help of domain expertise. However, human efforts are often costly in terms of time and effort. Thus, a mechanism that can address these issues while at the same time minimize human intervention is needed. To this end, we developed a semantic resolution knowledge base (SR-KB) that iteratively enriches itself through annotated instances validated by domain experts. Specifically, the SR-KB includes (1) a *manufacturing concepts mapping repository* (MCMR), (2) a *Naïve Bayes-based annotation corrector* (NBAC) and (3) an *instance validation platform* (IVP). MCMR helps the instance annotator solve the issue of terminological heterogeneity when annotating instances of IDM. NBAC is to correct misannotated instances (i.e., assigned with wrong or inaccurate class label) by resolving conceptual heterogeneity. It is trained by annotated instances and their features (§7.2.1) validated by domain experts with the assistance of IVP. As the instance annotation process goes by, both MCMR and NBAC will evolve and improve their performance. Consequently, the requirement for human intervention will be gradually reduced. Figure 7.1 shows how the three components of SR-KB interact with each other.

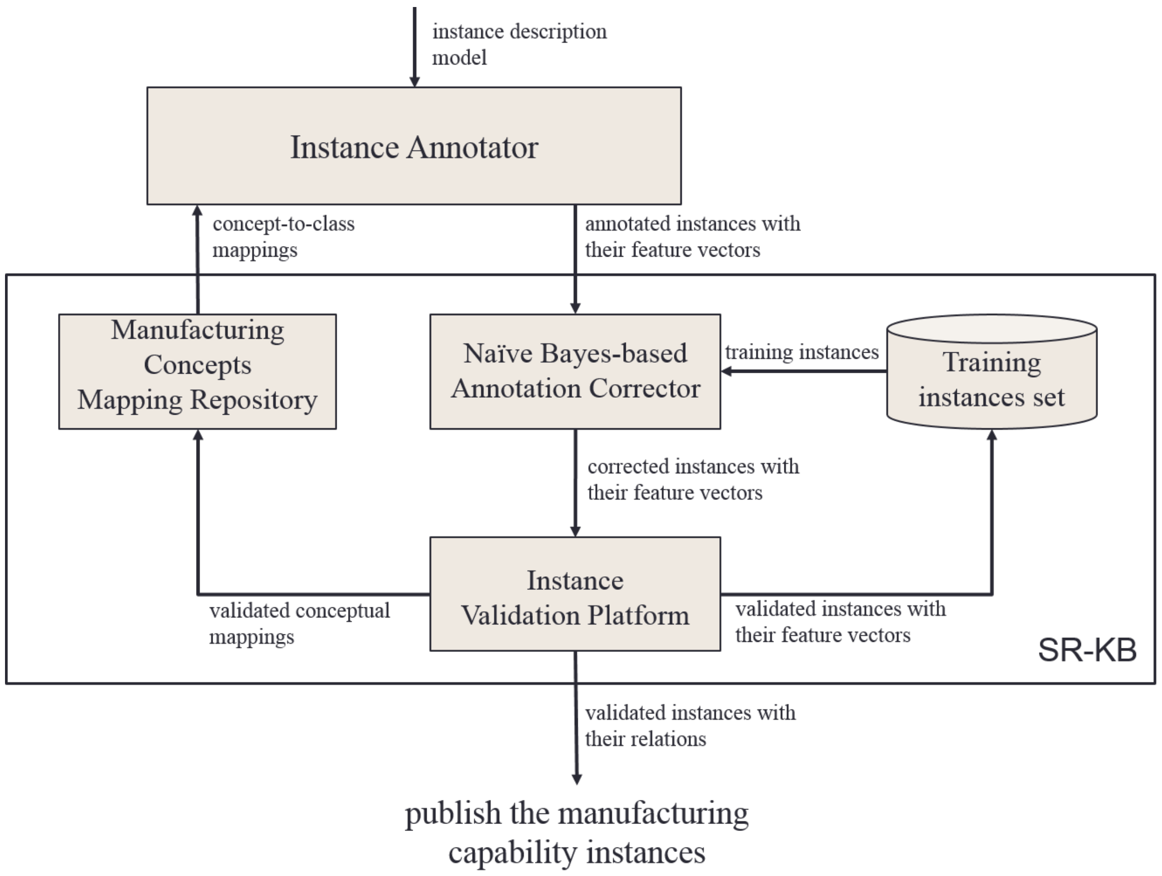

instance description model
Instance Annotator
concept-to-class mappings
annotated instances with their feature vectors
Manufacturing Concepts Mapping Repository
Naïve Bayes-based Annotation Corrector
training instances
Training instances set
corrected instances with their feature vectors
Instance Validation Platform
validated conceptual mappings
validated instances with their feature vectors
SR-KB
validated instances with their relations
publish the manufacturing capability instances

Figure 7.1 Interactions between components of SR-KB

In the following three sections, we will introduce these components and explain each of them in detail. In §7.4, we present the experimental evaluation on the effect of applying the semantic resolution knowledge base to instance annotation.

7.1 Manufacturing Concepts Mapping Repository

As discussed earlier, instance annotator cannot sufficiently deal with terminological heterogeneity (e.g., entities with similar meaning but have low similarity in their labels) when it maps concepts extracted from proprietary manufacturing web sites to classes defined in MDO because the instance annotator maps concepts to classes solely based on label similarity measures. The manufacturing concepts mapping repository (MCMR) comes to address this issue and help the instance annotator more effectively identify class labels for manufacturing service capability instances.

MCMR only stores mappings between concepts and classes that may have similar meaning but share low similarity in their labels (i.e., similarity below a predefined threshold). In other words, if a concept and a class are similar in their labels, the mapping between them will not be considered in MCMR. The rationale behind this consideration is twofold. First, it is not necessary to store mappings of this kind since they can be established by instance annotator at run time with any label similarity measure at low cost. Secondly, MCMR is constructed from a large amount of manufacturing web sites before it is applied to the instance annotator. That is to say once the similarities between concepts and classes are computed and stored in MCMR, they are expensive to change (e.g., if we want to change the label similarity measure from n-gram to token-based string similarity, we have to reconstruct the MCMR). However, in experiments or real applications, we might change label similarity measures from time to time according to different situations. Thus, it is more suitable to maintain the flexibility in choosing label similarity measures in different scenarios rather than adopting only one of them for all situations.

Intuitively, the MCMR can be considered as a simplified WordNet in the manufacturing domain that each entry in MCMR is a pair of synonyms or terms with similar word sense. More specifically, each entry is a 3-tuple <t, c, s> where t is a manufacturing concept extracted from web pages, c is a class defined in MDO that is mapped to t and s is the semantic relatedness score associated with the mapping between t and c, and it statistically measures the confidence that t and c are in fact semantically related. In the next section, we explain how this MCMR is constructed.

7.1.1 Constructing Manufacturing Concepts Mapping Repository

The MCMR is constructed from concept sets in the contexts of instances defined in IDM based on an approach similar to that of annotating manufacturing service capability instances (§6.2.2). We assume that concepts in the same concept set of an instance are semantically related. Thus, for each concept t in a concept set T_{n_i} of an instance n_i of IDM, if t cannot be mapped to any class defined in MDO based on label similarity measures, we will map t to a class c that is collaboratively determined by other members in T_{n_i}.

Specifically, for each concept set T_{n_i} (associated with instance n_i) defined in an IDM, we first locate the top class hierarchy H_k closest to T_{n_i} according to (6.6).

$$H_k = \underset{H_j \in H}{\operatorname{argmax}}\, closeness(T_{n_i}, H_j)$$

H is the set of class hierarchies that each includes more than one class mapped to concepts in T_{n_i}. The mapping from each concept $t_{n_i,q}$ in T_{n_i} to a class $c_{j,z}$ in $H_j \in H$ was established according to (6.4), which is recited below.

$$HM_j = \left\{ \begin{array}{l} (t_{n_i,q}, c_{j,z}) \Big| t_{n_i,q} \in T_{n_i}, c_{j,z} = \max_{c \in C} Sim^{L}_{hybrid}(t_{n_i,q}, c) \\ \qquad\qquad\qquad\qquad \text{for } Sim^{L}_{hybrid}(t_{n_i,q}, c) > d \text{ and } c_{j,z} \in H_j \end{array} \right\}$$

Based on this formula, the set of classes in H_k mapped by concepts in T_{n_i} is denoted as $C_{n_i} = \left\{ c_{k,z} \middle| (t_{n_i,q}, c_{k,z}) \in HM_k \right\}$ while the set of concepts that each is mapped to a class in any $H_j \in H$ is denoted as T'_{n_i} $= \left\{ t_{n_i,q} \middle| (t_{n_i,q}, c_{j,z}) \in HM_j \text{ and } c_{j,z} \in H_j \text{ and } H_j \in H \right\}$. Then, we identify the class c that best represents the semantics of C_{n_i} according to Algorithm 6.1. Last, we map each concept t in $T''_{n_i} = T_{n_i} \backslash T'_{n_i}$ (concepts in T_{n_i} but not in T''_{n_i}) to class c. Figure 7.2 shows a concrete example that illustrates this approach.

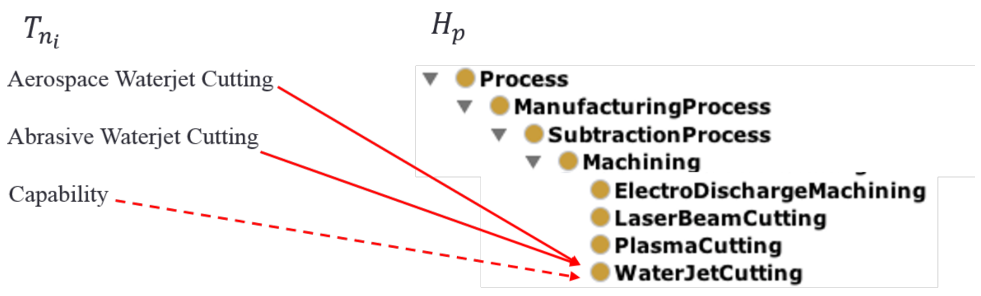


Figure 7.2 Example of creating an entry of the MCMR

As shown in Figure 7.2, **Capability**, **Abrasive waterjet cutting**, and **Aerospace waterjet cutting** are concepts in the same concept set of instance n_i denoted as T_{n_i}.

The top class hierarchy H_p is the one closest to T_{n_i} among all the top class hierarchies defined in MDO. Both concepts **Abrasive waterjet cutting** and **Aerospace waterjet cutting** were mapped to class **WaterJetCutting**, while concept **Capability** was failed to be mapped to any class defined in H_p based on label similarity measures. According to Algorithm 6.1, class **WaterJetCutting** is identified as the best representative of T_{n_i}. Thus, in this particular example, we map **Capability** to class **WaterJetCutting**. As we will explain shortly, the concept **Capability** may come from different manufacturing web sites and be mapped to many other classes defined in MDO.

The MCMR can be constructed from arbitrary amount of manufacturing web sites. It is highly likely that certain manufacturing concepts would appear in multiple web sites constructed based on different domain conceptualization. Consequently, those manufacturing concepts may be mapped to different classes defined in the MDO. For example, the concept **Capability** may be mapped to **Service**, **Process** or some other classes. We denote all the classes mapped by the same concept t as a class set C_t. Intuitively, each class in C_t can be considered representing a different sense or a synonym of t. We assign each class $c \in C_t$ with a score indicating the semantic relatedness between c and t. Such semantic relatedness score of mapping $<t, c>$ is measured according to (7.1) shown below. $f_{<t,c>}$ is the frequencies (i.e., counts) of mapping $<t, c>$ appearing in the entire MCMR where f_t is the frequencies of t appearing in mappings of MCMR, and d_t serves as the normalization factor that is the biggest semantic relatedness score among all the semantic relatedness scores between t and $c \in C_t$:

$$r(t,c) = \frac{f_{<t,c>}}{d_t \cdot f_t} \tag{7.1}$$

As will be explained in the next section, the MCMR will be applied to the instance annotation process to help map concepts of instances to classes defined in MDO. Thus, the semantic relatedness score of a mapping $<t, c>$ would be compared with the label similarity between t and c at the same scale. Without the normalization factor d_t, however, the semantic relatedness scores between t and $c \in C_t$ would be extremely small when the concept t was mapped to multiple classes. As a result, these

MCMR mappings may make no difference when applied to the instance annotation process (§7.1.2). For example, the instance **Marine Hardware** has a concept **Military** that was mapped to nine classes according to MCMR approach. Among the nine, class **Industry** has the highest semantic relatedness score with **Military** and such score is 0.27 without applying the normalization factor. The semantic relatedness score between **Military** and **Industry** is so small that it may have little impact on the result of annotating **Marine Hardware** with a class of MDO. After applying the normalization factor d_t, the semantic relatedness score between **Military** and **Industry** is increased to 1.0. Thus, the instance **Marine Hardware** will highly likely be classified into **Industry.** Figure 7.3 shows a concrete example of some mappings stored in MCMR constructed from 45 manufacturing web sites. For presentational purpose, the format of MCMR mappings shown below is different from the one defined in Definition 7.1. In Figure 7.3, the first term presented at the beginning of each row is a concept t that is followed by a set of classes C_t mapped by t and each class $c \in C_t$ is associated with a semantic relatedness score between c and t.

precision manufacturing: Equipment 0.89, ManufacturingService 1.0
MIG: Service 0.50, ManufacturingProcess 1.0
engraving: Service 1.0, Product 1.0, Process 1.0
crimp assembly: ToolAssembly 1.0
supplementary operation: AssemblyService 0.53, Joining 1.0
bystronic bystar: Service 0.38, LaserBeamCutting 0.47, System 1.0
abrasive blasting: Service 0.62, Process 1.0
substrate: ManufacturingProcess 0.40, Material 1.0
lubrication: Machining 0.79, Product 1.0
fixturing: Service 0.23, Process 0.27, Fixture 1.0
general capability: ToolAssembly 0.20, Part 0.53, Process 0.82, Service 1.0
military: Product 0.14, Equipment 0.17, Machining 0.20, Aluminum 0.74, Industry 1.0
short long run: InjectionMolding 0.99, ManufacturingService 1.0
slotting: Service 0.67, Process 1.0
highvacuum annealing: Process 1.0
plastic fabrication: Service 0.89, Machining 1.0
architectural: Machining 0.39, Industry 1.0
production powder coating service: Service 0.33, PowderCoating 1.0
aircraft: Product 0.53, Industry 1.0
milling capability: Material 0.96, Machining 1.0
laser cutting capability: Part 0.21, Service 0.44, LaserBeamCutting 1.0

Figure 7.3 Example of some concept-to-class mappings stored in the MCMR

The MCMR stores a set of concept-to-class mappings where each mapping between a manufacturing concept *t* and a class *c* is established based on the contexts of *t* and *c* rather than the label similarity between them. Consequently, MCMR is able to help the instance annotator to match concept and class that are semantically related but have low similarity in their labels, thereby improving the instance annotation accuracy. In §7.1.2, we will explain how the MCMR is applied to assisting the annotation of manufacturing service capability instances.

7.1.2 Using Manufacturing Concepts Mapping Repository

As stated at the beginning of this chapter, one of the major issues of our instance annotation approach is that it is sometimes inadequate in mapping concepts in the context of an instance to classes defined in MDO because of terminological heterogeneity. Consequently, some instances may be assigned incorrect class labels or fail to be assigned one. The experiment results shown in §6.3 demonstrated this point.

MCMR helps address the terminological heterogeneity occurring in the instance annotation process. It is applied to steps that require establish mappings between concepts in the concept set T_{n_i} and classes defined in MDO. Specifically, the place to where we apply the MCMR is identifying a class *c* in *C*, the set of classes defined in MDO, that is closest to T_{n_i} of n_i based on (6.3) defined in §6.2.2, which is recited below:

$$c_{j,z} = \max_{c \in C} Sim^{L}_{hybrid}(t_{n_i,q}, c).$$

Formula (6.3) measures the similarity between a concept $t_{n_i,q}$ in T_{n_i} and a class c in C solely based on the label similarity measure that as discussed before is not capable of dealing with the terminological heterogeneity. This can be addressed with the help of MCMR as in the revised formula below.

$$c_{j,z} = \max_{c \in C} \max\left\{Sim^{L}_{hybrid}(t_{n,q}, c), \mathrm{MCMR}(t_{n,q}, c)\right\} \tag{7.2}$$

MCMR is also applied to formula (6.5) defined in §6.2.2 in order to keep the similarity score between a concept $t_{n_i,q}$ and a class c consistent across the whole instance annotation process. The purpose of applying MCMR($t_{n_i,q}, c$) in (7.2) is to establish correspondence between $t_{n_i,q}$ and c that are synonyms or share similar word senses but have low label similarity. In other words, when $t_{n_i,q}$ and c share low similarity in their labels, they still can be mapped if they have high semantic relatedness score. As a result, more concepts in T_{n_i} can be mapped to classes defined in MDO, thereby improving the chance that the class label of instance n_i can be effectively identified. This is demonstrated by the experiment results shown in §7.4. Briefly, having MCMR been applied, the relaxed recall of the annotated instances improved significantly. However, the relaxed precision only slightly decreased. Among other things, this is mainly because that both the original instance annotator and the one with MCMR incorporated have difficulty in dealing with conceptual heterogeneity between proprietary manufacturing web sites and the manufacturing domain ontology. Consequently, certain instances were assigned incorrect class labels. The Naïve Bayes-based annotation corrector is developed to alleviate such issue.

7.2 Naïve Bayes-based annotation corrector

The primary objective of the instance annotator is to assign manufacturing service capability instances extracted from proprietary manufacturing web sites with class labels defined in the manufacturing domain ontology (MDO). To this end, these extracted instances should conform to the conceptualization of the MDO. However, because of conceptualization heterogeneity, conceptual conformation may fail during instance annotation. Although MCMR helps alleviate the issue of terminological heterogeneity existing in the instance annotation, it is insufficient in addressing the semantic heterogeneity. In §2.2.4, we have identified and discussed several types of semantic heterogeneity. With respect to the problem of instance annotation, we mainly focus on the issue of conceptual heterogeneity on instances in this section.

Conceptual heterogeneity occurs when the class label of an instance assigned by instance annotator is different from the one determined by domain experts. More specifically, if the two mismatched classes have no subsumption relationship between them, we consider the class label assigned by the instance annotator incorrect. For example, **sheet metal fabrication** was classified into class **Product** by the instance annotator, but it actually should be classified into class **ManufacturingProcess** that is in a different class hierarchy from that of **Product**. In this case, we consider **sheet metal fabrication** was incorrectly classified (i.e., assigned a wrong class label). On the other hand, if the two mismatched classes are on the same class path (one class is a superclass of the other), we consider the class label assigned by the instance annotator was inaccurate. For example, the **cnc laser cutting** was classified into **Process** class by the instance annotator, but it actually should be classified into **LaserCutting** class that is a subclass of **Process**. In this case, we consider the class label **Process** assigned to **cnc laser cutting** was correct but inaccurate.

The Naïve Bayes-based annotation corrector (NBAC) is trained to address the issue of class mismatch. It is iteratively trained by annotated instances validated by domain experts and applied to correct incorrect or inaccurate class labels of newly annotated instances. This iterative process of applying NBAC is illustrated in Figure 7.4 below.

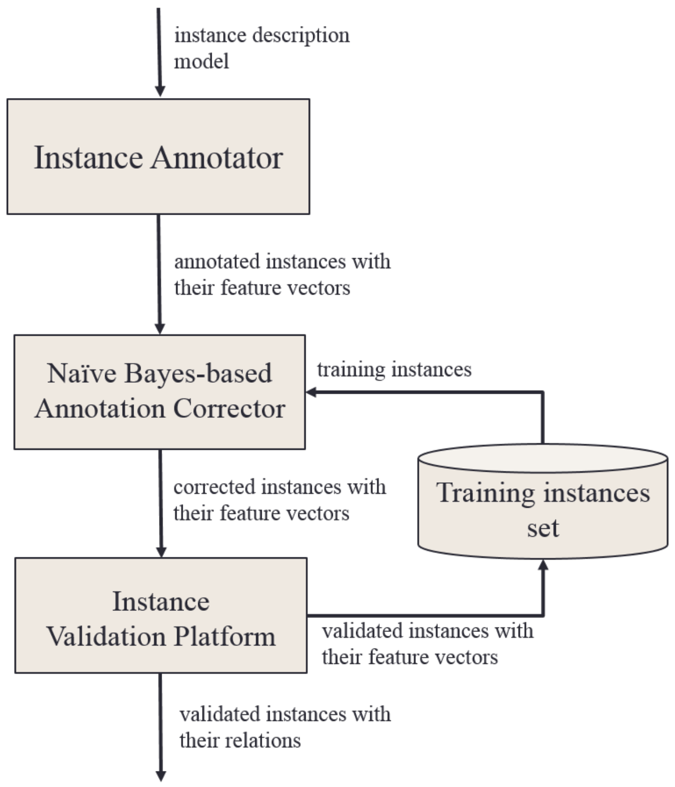

instance description model
Instance Annotator
annotated instances with their feature vectors
Naïve Bayes-based Annotation Corrector
training instances
corrected instances with their feature vectors
Training instances set
Instance Validation Platform
validated instances with their feature vectors
validated instances with their relations

Figure 7.4 The iterative procedure of applying Naïve Bayes-based annotation corrector

As shown in Figure 7.4, the NBAC takes as input annotated instances and features associated with these instances, and outputs instances with corrected class labels. Then, these corrected instances and their features are passed to the instance validation platform for validation. After the class labels of instances have been validated by domain experts with the assistance of the instance validation platform (§7.3), the validated instances and their associated features are stored in the training instance set that will be used for further training of the NBAC; Validated instances and their relations are stored in local repository.

7.2.1 Training Naïve Bayes-based Annotation Corrector

To train the Naïve-Bayes-based annotation corrector, we represent each instance n_i as a feature vector denoted as F_{n_i} created based on the output of the instance annotator. There are two types of features:

- *Lexical features* – a set of n-grams created from the label of n_i by the n-gram generator of the instance annotator.
- *Conceptual mapping features* – mappings established from concepts of n_i to classes of MDO or from relations of n_i to properties of MDO. Both concepts and relations are from the context of instance n_i and those mappings are established during the instance annotation process. We call the former mappings *concept-to-class mapping features* denoted as *CC* while the latter mappings *relation-to-property mapping features* denoted as *RP*. A conceptual mapping feature is in the form of <e_1, e_2>, where e_1 is either a concept or a relation of n_i while e_2 is either a class or a property defined in MDO.

The following is an example showing the features of instance **Aerospace Waterjet Cutting** which has a context including concepts **Aerospace Waterjet Cutting**, **Abrasive Waterjet Cutting** and **Capability**, and a relation **material include**:

Table 7.1 Features of instance Aerospace Waterjet Cutting

Features	**Lexical Features**:		n-grams in the label of Aerospace Waterject Cutting (In practice, we use 3-grams)
	Conceptual Mapping Features:	***CC***:	<Aerospace Waterjet Cutting, WaterJetCutting>
			<Abrasive Waterjet Cutting, WaterJetCutting>
			<Capability, Product>
		RP:	<material include, hasMaterial>

The union of features of instances belonging to a class *c* forms the features of class *c*. The lexical features of a class *c* indicate what n-grams or terms that instances of class *c* might have. The conceptual mapping features of a class *c* represent (both explicit and hidden) relationships between class *c* and the contexts of its instances extracted from proprietary manufacturing web sites. Put another way, conceptual mapping features of classes may bridge the conceptual gap between proprietary manufacturing web sites that instances were extracted from and the manufacturing domain ontology that instances would be populated into. Figure 7.5 illustrates the process of how the class label of the second incoming instance n_2 can be corrected by the NBAC that was trained by the first incoming instance n_1 validated by domain expert. We assume that both the two instances have only one concept **Capability** (in reality, one instance typically has more than one concepts) and belong to class **Process**.

	instance n_1 With concept **Capability**	instance n_2 With concept **Capability**
Instance annotation stage	Capability → Product ⬇ infer n_1 is of class **Product**	Capability → Product ⬇ infer n_2 is of class **Product**
Annotation correction stage by NBAC	For $\forall c \in C$ and c ≠ Product: P(Capability→Product | c) = 0 and only P(Capability→Product | Product) ≠ 0 So: The class label of n_1 was not changed	P(Process) = b P(Product) = d P(Process | n_2) ∝ P(Capability→Product | Process) P(Process) $\propto a \cdot b$ P(Product | n_2) ∝ P(Capability→Product | Product) P(Product) $\propto c \cdot d$ Assuming $a \cdot b > c \cdot d$, So: n_2 is changed to be an instance of class **Process**
Instance validation stage	n_1 is changed by domain experts to be of class **Process**	The class label (i.e., Process) of n_2 is correct
Retrain NBAC	add the conceptual mapping feature Capability→Product to class **Process** and compute the probability P(Capability→Product | Process) = a	increase the probability of P(Capability→Product | Process)

Figure 7.5 Example of how a conceptual mapping feature helps correct class label of instance

In the instance annotation stage as shown in Figure 7.5, the concept **Capability** of instance n_1 was mapped to class **Product** and thus the instance annotator outputted instance n_1 with class label **Product** and a conceptual mapping feature **Capability→Product**. In the annotation correction stage, the class label of n_1 was unchanged since only class **Product** has the feature **Capability→Product** (as we will explain shortly, we temporarily add features of newly incoming instance to NBAC before the annotation process to avoid annotation/classification bias). In the instance validation stage, domain expert changed the class label of n_1 from **Product** to **Process**. Thus, the **Capability→Product** was added as a new conceptual mapping feature of class **Process** in the stage of retraining NBAC. The instance annotator annotated the second incoming instance n_2 with class label **Product** for the same reason when annotating instance n_1. However, in the annotation correction stage, NBAC changed the label of n_2 from **Product** to **Process** since the **Capability→Product** is more likely the feature of class **Process** than that of **Product**. Then, a domain expert validated the class label of instance n_2 and proved that the class label is correct. As a result, the probability that an instance with feature **Capability→Product** is of class **Process** was increased in the last stage.

Intuitively, the conceptual mapping feature **Capability→Product** of class **Process** can be seen as an evidence indicates that a newly incoming instance n_i may belong to class **Process**. The rationale behind the idea that we utilize conceptual mappings of instances as features of classes rather than concepts of instances is that the same concept can be extracted from different web sites and mapped to different (e.g., concept **Capability** can be mapped to classes **Product**, **WaterJetCutting**, **Service** and **Process**) and more importantly these conceptual mapping of instances can either be reconcile or conflict with the conceptualization model of MDO. Either case, these features should be captured by the NBAC and serves as evidences for annotating newly incoming instances.

We interpret the probability of each feature *f* appearing in class *c* as a measure of how much evidence *f* contributes that instance n_i belongs to class *c* if n_i possesses the feature *f*. The probability of an instance n_i being in class c is computed as:

$$P(c|n_i) \propto \hat{P}(c) \prod_{f \in F_{n_i}} \hat{P}(f|c) \qquad (7.4)$$

Here, we use $\hat{P}$ instead of P because we do not know the true values for the parameter, but estimate them from the training data set (i.e., annotated instances validated by domain expert). Each conditional parameter $\hat{P}(f|c)$ is a weight that indicates the relative frequency of feature *f* occurs in class *c* while the prior $\hat{P}(c)$ is a weight that indicates the relative frequency of class *c*. Thus, the multiplication of the prior $\hat{P}(c)$ and $\hat{P}(f \mid c)$ of features is a measure of how much evidence there is for the instance n_i being in the class *c*.

We use the *maximum likelihood estimate* (MLE) [88] to estimate the prior parameter and conditional parameter. For the priors, the estimate is:

$$\hat{P}(c) = \frac{N_c + 1}{\sum_{c' \in C} (N_{c'} + 1)} \tag{7.5}$$

where N_c is the number of instances in class *c*. The number 1 in the formula is to eliminate zero prior probability of classes that have no instances. This technique is referred to as *Add-one smoothing* [89] and can be interpreted as a uniform prior that each class has one instance. In other words, when there is no instance for all (or most) of classes *C* defined in the MDO, we treat the prior probability of these classes equally. Thus, $\hat{P}(c)$ is the ratio of number of instances in class *c* to total number of instances belong to classes in *C*.

We estimate the conditional probability $\hat{P}(f \mid c)$ as the relative frequency of feature *f* occurring in instances belonging to class *c*:

$$\hat{P}(f \mid c) = \frac{K_{cf} + 1}{\sum_{f' \in F} (K_{cf'} + 1)} \tag{7.6}$$

where K_{cf} is the number of occurrences of feature *f* in instances from class *c*. Similarly, the number 1 in this formula aims to eliminate the zero conditional probability of certain features that do not occur in class *c*. *F* contains features of all classes defined in MDO.

One issue with the estimation of $\hat{P}(c)$ and $\hat{P}(f \mid c)$ shown in (7.5) and (7.6) is that when computing N_c, K_{cf} and *F*, they did not take into consideration evidences *e* for instance n_i being in the original class c_o that was assigned by the instance annotator to n_i. Consequently, they tend to underestimate $\hat{P}(c_o)$ and $\hat{P}(f|c_o)$. As stated before, the goal of the Naïve Bayes-based annotation correction algorithm is to correct the original class label c_o of an instance n_i that is mis-annotated (i.e., assigned with a wrong or inaccurate class label) by instance annotator. In other words, only when there exist stronger evidences, compared to the evidences for n_i being in the original class label c_o, that prove a new class c_n is more likely to be the class label of n_i should we change the class label from c_o to c_n. Thus, the absence of evidences e may lead to the originally correct class label of n_i being changed to a wrong one.

For this reason, we temporally add the instance n_i of class c_o to the training data when estimating the parameters of the Naïve Bayes-based annotation correction algorithm. F, N_{c_o} and $K_{c_o f}$ are updated as below.

$$\begin{aligned} &F = F \cup F_{n_i} \\ &N_{c_o} = N_{c_o} + 1 \\ &K_{c_o f} = K_{c_o f} + 1 \qquad \textit{for } \forall f \in F_{n_i} \end{aligned} \tag{7.7}$$

To permanently add an instance n_i to the training data set, the class label of n_i must be validated by domain experts assisted by the instance validation platform. In §7.3, we will present the instance validation platform and explain how a domain expert can validate class labels of instances.

As shown in Figure 7.4, having been trained, NBAC was applied to the instance annotator to correct class labels of instances that were mis-annotated.

7.2.2 Using Naïve Bayes-based Annotation Corrector

The procedure of correcting the class label of an instance involves following two steps:

Step 1: Obtain a set of potential class labels C for instance n_i. For computational efficiency, we will not compute the $P(c|n_i)$ over all classes defined in MDO. Instead, we compute $P(c|n_i)$ only over classes that are potential class labels of n_i. These potential class labels are identified based on the corrections performed by domain experts assisted by the instance validation platform (IVP). More specifically, if the class label c_o, identified by the instance annotator, of an instance n_i was corrected by domain expert to a new class label c_n, this new class label will be considered as one of the potential class labels for any new incoming instance that was assigned with class label c_o by instance annotator. We denote the set of potential class labels for instance n_i as C_p.

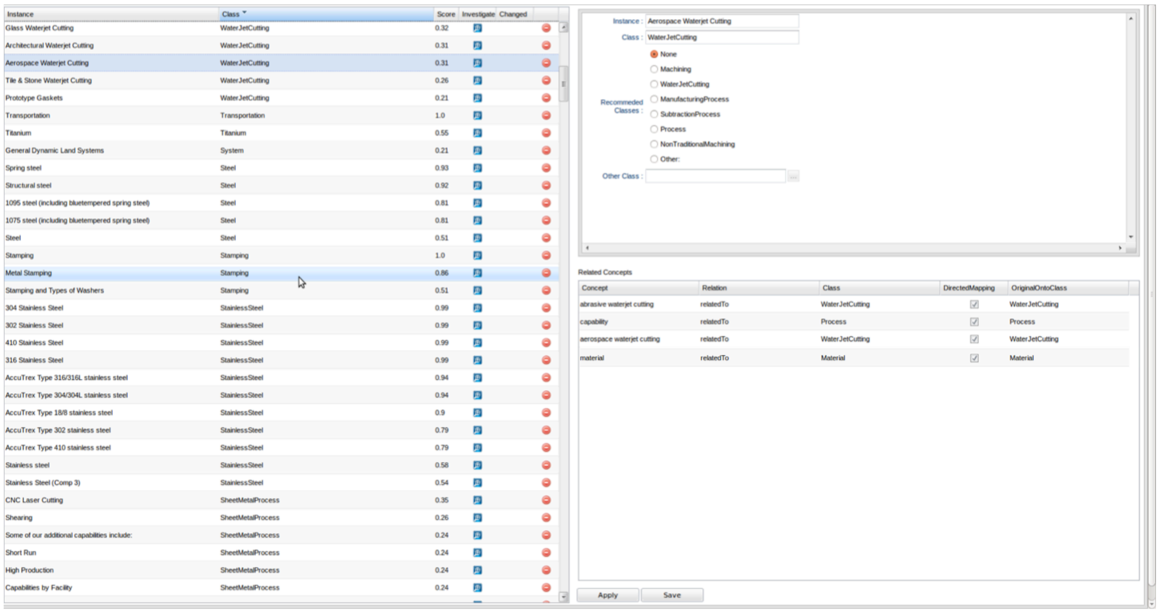
Step 2: Choses a class from C_p as class label for n_i based on the Naïve Bayes algorithm shown below:

$$c_{n_i} = \underset{c \in C_p}{\operatorname{argmax}} \hat{P}(c) \prod_{f \in F_{n_i}} \hat{P}(f|c) \tag{7.8}$$

The incoming instances along with their class labels, possibly corrected by NBAC, will be passed to the instance validation platform for validation. Then, validated instances will be added to the training instance pool for retraining the NBAC. In the next section, we will present the instance validation platform and its core functionalities.

7.3 Instance Validation Platform

The purpose of the instance validation platform (IVP) is to assist domain experts to validate class labels of inputted instances and make changes if necessary. IVP also allows domain experts to validate and change conceptual mapping features of instances. Figure 7.6 shows the main user interface of the IVP.



Instance	Class	Score	Investigate	Changed	
Glass Waterjet Cutting	WaterJetCutting	0.32	◘		⊖
Architectural Waterjet Cutting	WaterJetCutting	0.31	◘		⊖
Aerospace Waterjet Cutting	WaterJetCutting	0.31	◘		⊖
Tile & Stone Waterjet Cutting	WaterJetCutting	0.26	◘		⊖
Prototype Gaskets	WaterJetCutting	0.21	◘		⊖
Transportation	Transportation	1.0	◘		⊖
Titanium	Titanium	0.55	◘		⊖
General Dynamic Land Systems	System	0.21	◘		⊖
Spring steel	Steel	0.93	◘		⊖
Structural steel	Steel	0.92	◘		⊖
1095 steel (including bluetempered spring steel)	Steel	0.81	◘		⊖
1075 steel (including bluetempered spring steel)	Steel	0.81	◘		⊖
Steel	Steel	0.51	◘		⊖
Stamping	Stamping	1.0	◘		⊖
Metal Stamping	Stamping	0.86	◘		⊖
Stamping and Types of Washers	Stamping	0.51	◘		⊖
304 Stainless Steel	StainlessSteel	0.99	◘		⊖
302 Stainless Steel	StainlessSteel	0.99	◘		⊖
410 Stainless Steel	StainlessSteel	0.99	◘		⊖
316 Stainless Steel	StainlessSteel	0.99	◘		⊖
AccuTrex Type 316/316L stainless steel	StainlessSteel	0.94	◘		⊖
AccuTrex Type 304/304L stainless steel	StainlessSteel	0.94	◘		⊖
AccuTrex Type 18/8 stainless steel	StainlessSteel	0.9	◘		⊖
AccuTrex Type 302 stainless steel	StainlessSteel	0.79	◘		⊖
AccuTrex Type 410 stainless steel	StainlessSteel	0.79	◘		⊖
Stainless steel	StainlessSteel	0.58	◘		⊖
Stainless Steel (Comp 3)	StainlessSteel	0.54	◘		⊖
CNC Laser Cutting	SheetMetalProcess	0.35	◘		⊖
Shearing	SheetMetalProcess	0.26	◘		⊖
Some of our additional capabilities include:	SheetMetalProcess	0.24	◘		⊖
Short Run	SheetMetalProcess	0.24	◘		⊖
High Production	SheetMetalProcess	0.24	◘		⊖
Capabilities by Facility	SheetMetalProcess	0.24	◘		⊖

Instance : Aerospace Waterjet Cutting

Class : WaterJetCutting

Recommeded Classes : [✓ None] Machining WaterJetCutting ManufacturingProcess SubtractionProcess Process NonTraditionalMachining Other:

Other Class :

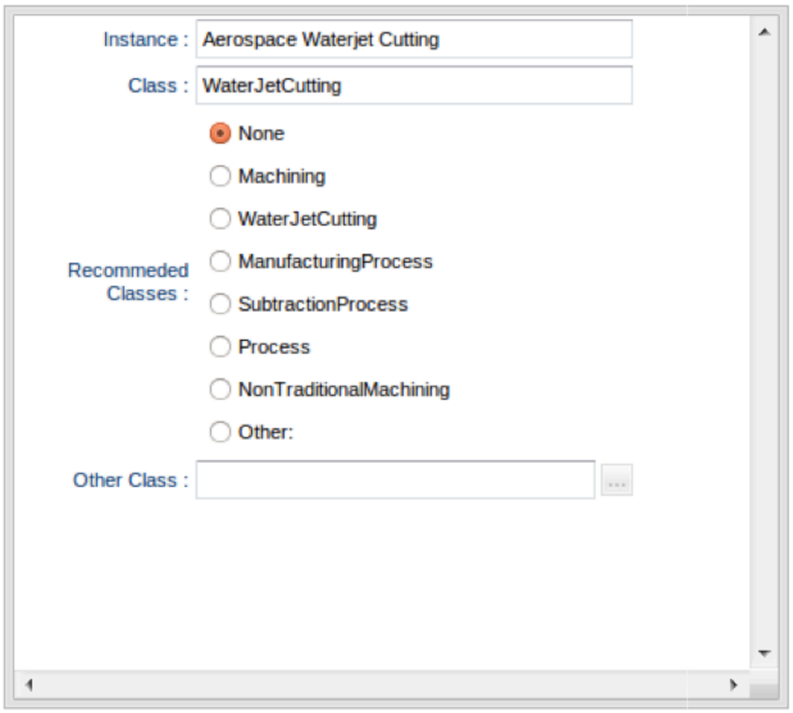
Related Concepts

Concept	Relation	Class	DirectedMapping	OriginalOntoClass
abrasive waterjet cutting	relatedTo	WaterJetCutting	✓	WaterJetCutting
capability	relatedTo	Process	✓	Process
aerospace waterjet cutting	relatedTo	WaterJetCutting	✓	WaterJetCutting
material	relatedTo	Material	✓	Material

Figure 7.6 Main user interface of instance validation platform

This graphical user interface is designed for domain experts who are not familiar with OWL-based ontologies. It hides technical details of ontologies and only shows relevant information for domain experts performing tasks at hand (e.g., change class labels or conceptual mappings of instances). On the left side of this user interface presents a list of annotated instances for quick review. When a domain expert selects an instance from the list, the detail of that instance will show on the right.

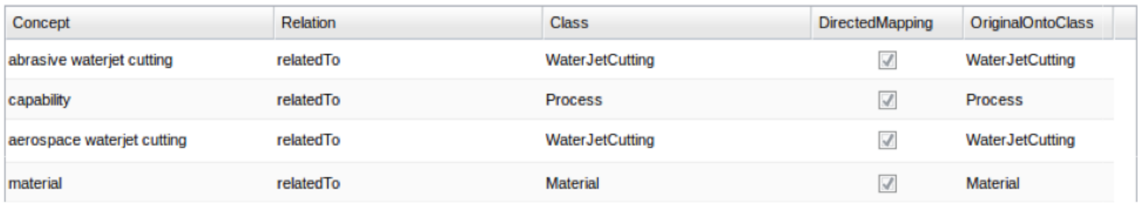
An innovative functionality of this user interface is that it provides a set of recommended classes for the selected instance. Domain experts only need to choose one of the recommended classes as the class label for the selected instance without needing to navigate the whole manufacturing domain ontology that may contains hundreds of classes. The recommended classes of an instance n_i includes subclasses of the original class label of n_i and classes from other class hierarchies that are related to the context of instance n_i. If domain experts cannot find an appropriate class among the recommended classes, they can navigate the whole ontology to find one. Figure 7.7 shows recommended classes for instance **Aerospace Waterjet Cutting**.



Instance :	Aerospace Waterjet Cutting
Class :	WaterJetCutting
Recommeded Classes :	[✓ None]
	Machining
	WaterJetCutting
	ManufacturingProcess
	SubtractionProcess
	Process
	NonTraditionalMachining
	Other:
Other Class :	

Figure 7.7 Recommended classes for instance Aerospace Waterjet Cutting

IVP also enables domain expert to validate conceptual mapping features of instances and make changes if necessary. As defined in §7.2.1, a conceptual mapping feature of an instance n_i is in the form of $<e_1, e_2>$, where e_1 is either a concept or a relation of n_i while e_2 is either a class or a property defined MDO. Domain expert is allowed to change e_2 to one of the entities (i.e., classes or properties) related to the class label c of n_i. More specifically, if e_1 is a concept, it can be changed to a superclass or subclass of c. Otherwise, it can be changed to a property of c. Figure 7.8 shows the conceptual mapping features of instance **Aerospace Waterjet Cutting**.



Concept	Relation	Class	DirectedMapping	OriginalOntoClass
abrasive waterjet cutting	relatedTo	WaterJetCutting	✓	WaterJetCutting
capability	relatedTo	Process	✓	Process
aerospace waterjet cutting	relatedTo	WaterJetCutting	✓	WaterJetCutting
material	relatedTo	Material	✓	Material

Figure 7.8 Conceptual mapping features of instance Aerospace Waterjet Cutting

As depicted in Figure 7.1, the output of IVP was divided into three parts: the validated instances along with their class labels and features (both lexical features and conceptual mapping features) will be added to the training data set used to train the Naïve Bayes-based annotation corrector; the validated conceptual mapping features of instances will be applied to update the manufacturing concepts mapping repository; the validated instances with their class labels and properties will be published as public accessible data, which can be linked to the Linked Data Cloud.

7.4 Experimental Evaluation

This section presents the performance analysis on the instance annotation approach when applying SR-KB. To better analyze the performance and effects of applying the SR-KB, we compared the experimental results over four different instance annotation strategies:

1. Annotate extracted instances without applying the SR-KB
2. Annotate extracted instances only applying the manufacturing concepts mapping repository (MCMR)
3. Annotate extracted instances only applying the Naïve Bayes-based annotation corrector (NBAC)
4. Annotate extracted instances applying both MCMR and NBAC. That is, applying the SR-KB as a whole.

The first strategy serves as the control group based on which the effects of applying SR-KB can be measured. The second and third instance annotation strategy applies MCMR and NBAC of the SR-KB individually with the purpose of demonstrating how each of the two components would affect the instance annotation approach and thus shedding more light on how the SR-KB as a whole would affect the performance of the instance annotator.

Besides using the relaxed precision, relaxed recall and F1-meausre introduced in §6.3 to measure the performance on each of the four instance annotation strategies, we use *correction rate* to measure the efforts that domain experts would take to correct incorrect or inaccurate class labels of instances. Smaller correction rate indicates lesser efforts required for domain experts to correct class labels of instances. We measure the effort of correcting the incorrect or inaccurate class label of an instance n_i as the distance from the expected class EC_{n_i} (defined in the ground truth) of instance n_i to the assigned class RC_{n_i} (assigned by instance annotator) of instance n_i. Such distance is measured as follow:

$$d(EC_{n_i}, RC_{n_i}) = 1 - \omega(EC_{n_i}, RC_{n_i}) \tag{7.9}$$

where the $\omega(EC_{n_i}, RC_{n_i})$ is defined in (6.9) and it measures the proximity between the expected class EC_{n_i} and assigned class RC_{n_i} of n_i. Thus, the correction rate for a set of annotated instances is calculated as follow:

$$\text{correction rate} = \frac{\sum_{i=1}^{n} d(EC_{n_i}, RC_{n_i})}{n} \tag{7.10}$$

We use the same ground truth introduced in §6.3 to evaluate the performance on the instance annotation approach applying SR-KB. We created the ground truth by randomly selecting 45 manufacturing websites from Thomasnet and automatically extracting totally 14894 instances of interest from those web sites. Then, we manually annotated those instances with classes defined in MDO. The correctness of these annotated instances is validated by Yunsu Lee, who is a researcher at the National Institute of Standards and Technology (NIST) and familiar with both the manufacturing domain and OWL-based ontology. Table 7.2 presents the experimental results on the four instance annotation strategies. The Pre, Rec, F1, and Cor stands for relaxed precision, relaxed recall, F1 measure and correction rate respectively.

As demonstrated in the last row of Table 7.2 (average performance over all websites), the second strategy (i.e., MCMR applied) significantly improved the relaxed recall of the instance annotator compared to the first strategy. This was to be expected since MCMR helps instance annotator establish more mappings between concepts extracted from manufacturing web sites and classes defined in the MDO, thereby improving the chance of assigning an instance with a class label. However, the relaxed precision of the MCMR slightly decreased. This is mainly because the class labels of certain newly annotated instances may not be accurate or even correct, thereby dragging down the overall relaxed precision. The third strategy applies the Naïve Bayes-based annotation corrector (NBAC) to assign new class labels to instances. More specifically, it may either correct wrongly assigned class labels of instances or assign class labels to instances that have not been assigned. Thus, it is supposed to improve both the relaxed precision and relaxed recall. However, as presented in Table 7.2 (from column 10 to column 13 on the last row), the relaxed precision of the third strategy dropped noticeably by 0.06 although the relaxed recall improved significantly (not as significantly as the second strategy) compared to the first strategy. There are two main reasons that lead to this outcome. For one thing, the NBAC applied in the third strategy assigns class labels to un-annotated instances solely based on the lexical features of these instances. As a result, it more likely assigns wrong (inaccurate or incorrect) class labels to these instances compared to the second strategy. According to (6.8), which is also recited below, newly annotated instances will always improve the relaxed recall, but they may negatively affect the relaxed precision if they were annotated with wrong class labels.

$$\text{relaxed precision} = \frac{\sum_{i=1}^{m} \omega(EC_{n_i}, RC_{n_i})}{m},$$

$$\text{relaxed recall} = \frac{\sum_{i=1}^{m} \omega(EC_{n_i}, RC_{n_i})}{n}$$

The second reason for the relatively poor performance of the third strategy in terms of relaxed precision compared to the first and second strategy is the lack of conceptual mapping features for NBAC to make accurate classification decisions. As explained in §7.2.1, NBAC choses class labels for instances based on the probability distribution of features (both lexical features and conceptual mapping features) appearing in instances of each class. Lacking conceptual mapping features may lead lexical features to dominating the decision-making process of NBAC, thereby decreasing the relaxed precision of the annotation of instances.

Based on the analysis stated above, exploiting more conceptual mapping features to train the NBAC would potentially improve the performance of the NBAC. The experimental result presented in Table 7.2 from column 14 to column 17 demonstrates this point. It shows that strategy four (SR-KB applied) has the best performance over all other three strategies. The 0.88 relaxed precision score and the 0.86 relaxed recall score on average demonstrate that both the quality and quantity of annotated instances increased significantly compared to the other three strategies. Also notice that the relaxed precision is almost identical to the relaxed recall. According to (6.8), this indicates that almost all extracted instances have been successfully assigned with class labels. Figure 7.9, Figure 7.10 and Figure 7.11 graphically compare relaxed precision, relaxed recall and F1 measure among the four instance annotation strategies across 45 manufacturing web sites. The three figures manifest that strategy four outperforms other three strategies in all the three measures. The advantages of strategy four are even more obvious at later stage of the instance annotation process. This is because, as will explain in detail shortly, that more and more validated instances are available to train the NBAC as the instance annotation process continues, thereby improving the class label correction ability of NBAC.

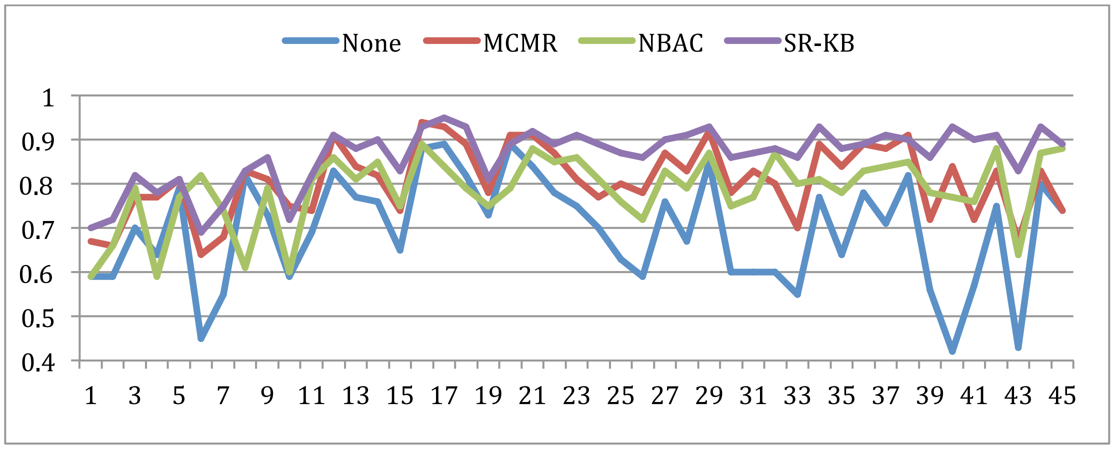


Figure 7.9 Comparison on relaxed recall among the four strategies

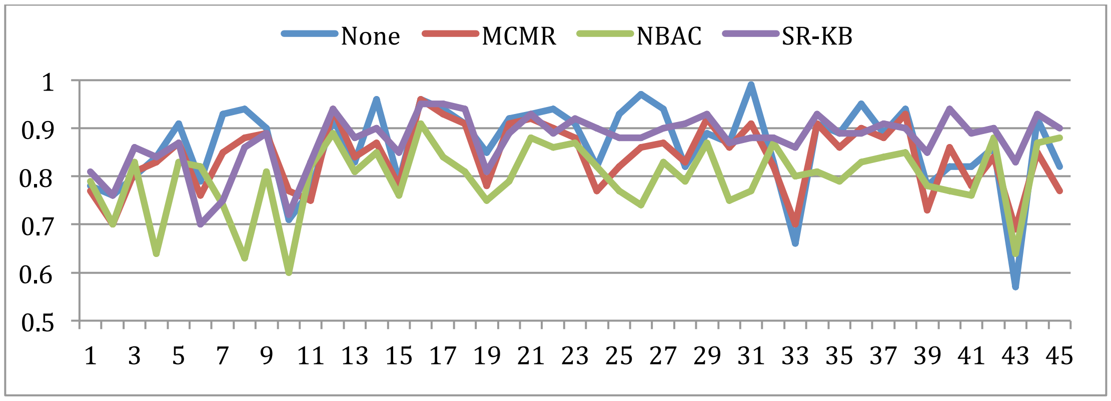


Figure 7.10 Comparison on relaxed precision among the four strategies

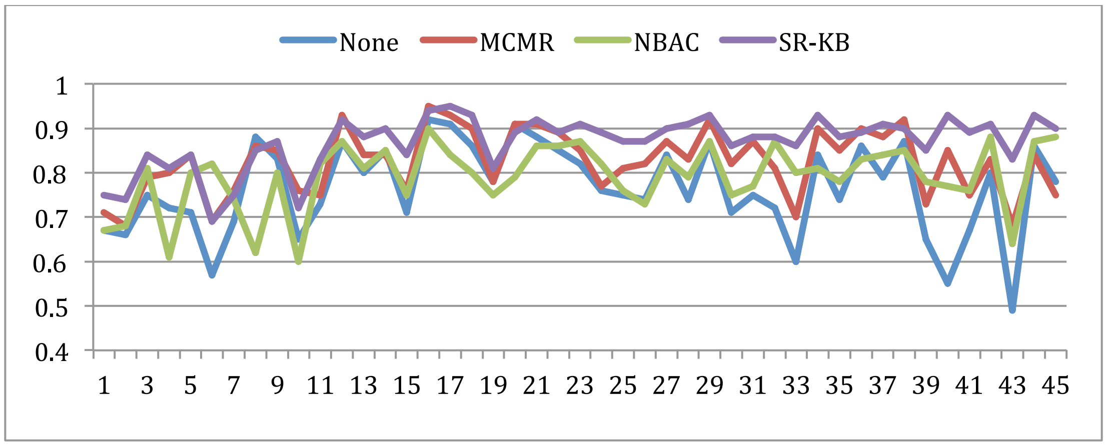


Figure 7.11 Comparison on F1 measure among the four strategies

Table 7.2 Comparison on relaxed precision, relaxed recall and correction rate among four instance annotation strategies

Web Sites	Strategy one: None				Strategy two: MCMR				Strategy three: NBAC				Strategy four: SR-KB			
	Pre.	Rec.	F1	Cor.	Pre.	Rec.	F1	Cor.	Pre.	Rec	F1	Cor.	Pre.	Rec.	F1	Cor.
Schulermfg	0.78	0.59	0.67	0.41	0.77	0.67	0.71	0.33	0.78	0.59	0.67	0.41	0.77	0.67	0.71	0.33
Soundmfg	0.76	0.59	0.66	0.41	0.7	0.66	0.68	0.34	0.7	0.66	0.68	0.34	0.76	0.72	0.74	0.28
numericalconcepts	0.8	0.7	0.75	0.3	0.81	0.77	0.79	0.23	0.83	0.79	0.81	0.21	0.86	0.82	0.84	0.18
Basssettinc	0.84	0.64	0.72	0.36	0.83	0.77	0.8	0.23	0.64	0.59	0.61	0.41	0.84	0.78	0.81	0.22
Aerostarmfg	0.91	0.69	0.71	0.31	0.87	0.81	0.84	0.19	0.83	0.77	0.8	0.23	0.87	0.81	0.84	0.19
magnamachine	0.79	0.45	0.57	0.55	0.76	0.64	0.69	0.36	0.82	0.82	0.82	0.18	0.7	0.69	0.69	0.31
Ashleyward	0.93	0.55	0.69	0.45	0.85	0.68	0.76	0.32	0.74	0.74	0.74	0.26	0.75	0.75	0.75	0.25
Accutrex	0.94	0.82	0.88	0.18	0.88	0.83	0.86	0.17	0.63	0.61	0.62	0.39	0.86	0.83	0.85	0.17
Weaverandsons	0.9	0.73	0.831	0.27	0.89	0.81	0.85	0.19	0.81	0.79	0.8	0.21	0.89	0.86	0.87	0.14
Astromfg	0.71	0.59	0.65	0.41	0.77	0.75	0.76	0.25	0.6	0.6	0.6	0.4	0.72	0.72	0.72	0.28
Wisconsinmetalparts	0.77	0.69	0.73	0.31	0.75	0.74	0.75	0.26	0.82	0.81	0.82	0.19	0.83	0.82	0.83	0.18
Tmfincorporated	0.92	0.83	0.87	0.17	0.94	0.91	0.93	0.09	0.89	0.86	0.87	0.14	0.94	0.91	0.92	0.09
Robersontool	0.83	0.77	0.8	0.23	0.84	0.84	0.84	0.16	0.81	0.81	0.81	0.19	0.88	0.88	0.88	0.12
Spmfg	0.96	0.76	0.85	0.24	0.87	0.82	0.84	0.18	0.85	0.85	0.85	0.15	0.9	0.9	0.9	0.1
Smccontractmfg	0.79	0.65	0.71	0.35	0.78	0.74	0.76	0.26	0.76	0.75	0.75	0.25	0.85	0.83	0.84	0.17
Swiftglass	0.96	0.88	0.92	0.12	0.96	0.94	0.95	0.06	0.91	0.89	0.9	0.11	0.95	0.93	0.94	0.07
Ameristarmfg	0.94	0.89	0.91	0.11	0.93	0.93	0.93	0.07	0.84	0.84	0.84	0.16	0.95	0.95	0.95	0.05
Jnmetalproducts	0.91	0.82	0.86	0.18	0.91	0.89	0.9	0.11	0.81	0.79	0.8	0.21	0.94	0.93	0.93	0.07
Nobleindustries	0.85	0.73	0.78	0.27	0.78	0.78	0.78	0.22	0.75	0.75	0.75	0.25	0.81	0.81	0.81	0.19
Aalloy	0.92	0.89	0.91	0.11	0.91	0.91	0.91	0.09	0.79	0.79	0.79	0.21	0.89	0.89	0.89	0.11
Portersfab	0.93	0.84	0.88	0.16	0.92	0.91	0.91	0.09	0.88	0.88	0.86	0.12	0.93	0.92	0.92	0.08
Californiabrazing	0.94	0.78	0.85	0.22	0.9	0.87	0.89	0.13	0.86	0.85	0.86	0.15	0.89	0.89	0.89	0.11

Web Sites	Pre.	Rec.	F1	Cor.	Pre.	Rec.	F1	Cor.	Pre.	Rec	F1	Cor.	Pre.	Rec.	F1	Cor.
Columbiamanufacturing	0.91	0.75	0.82	0.25	0.88	0.81	0.85	0.19	0.87	0.86	0.87	0.14	0.92	0.91	0.91	0.09
Mantecservicesinc	0.82	0.7	0.76	0.3	0.77	0.77	0.77	0.23	0.82	0.81	0.82	0.19	0.9	0.89	0.89	0.11
Kilgoremfg	0.93	0.63	0.75	0.37	0.82	0.8	0.81	0.2	0.77	0.76	0.76	0.24	0.88	0.87	0.87	0.13
Elgeprecision	0.97	0.59	0.74	0.41	0.86	0.78	0.82	0.22	0.74	0.72	0.73	0.28	0.88	0.86	0.87	0.14
Dynamicprecision	0.94	0.76	0.84	0.24	0.87	0.87	0.87	0.13	0.83	0.83	0.83	0.17	0.9	0.9	0.9	0.1
Pbandm	0.82	0.67	0.74	0.33	0.83	0.83	0.83	0.17	0.79	0.79	0.79	0.21	0.91	0.91	0.91	0.09
Schenketool	0.89	0.85	0.87	0.15	0.92	0.92	0.92	0.08	0.87	0.87	0.87	0.13	0.93	0.93	0.93	0.07
Pierceindustries	0.87	0.6	0.71	0.4	0.86	0.78	0.82	0.22	0.75	0.75	0.75	0.25	0.87	0.86	0.86	0.13
Westfieldmachine	0.99	0.6	0.75	0.4	0.91	0.83	0.87	0.17	0.77	0.77	0.77	0.23	0.88	0.87	0.88	0.13
Ardelengineering	0.83	0.6	0.72	0.4	0.82	0.8	0.81	0.2	0.87	0.87	0.87	0.13	0.88	0.88	0.88	0.12
Hhsmm	0.66	0.55	0.6	0.45	0.7	0.7	0.7	0.3	0.8	0.8	0.8	0.2	0.86	0.86	0.86	0.14
Wiegeltoolworks	0.91	0.77	0.84	0.23	0.91	0.89	0.9	0.11	0.81	0.81	0.81	0.19	0.93	0.93	0.93	0.07
Adaptplastics	0.89	0.64	0.74	0.36	0.86	0.84	0.85	0.16	0.79	0.78	0.78	0.22	0.89	0.88	0.88	0.12
Mpi-dms	0.95	0.78	0.86	0.22	0.9	0.89	0.9	0.11	0.83	0.83	0.83	0.17	0.89	0.89	0.89	0.11
Contract-mfg	0.89	0.71	0.79	0.29	0.88	0.88	0.88	0.12	0.84	0.84	0.84	0.16	0.91	0.91	0.91	0.09
Hloeb	0.94	0.82	0.87	0.18	0.93	0.91	0.92	0.09	0.85	0.85	0.85	0.15	0.9	0.9	0.9	0.1
Milacronmachining	0.78	0.56	0.65	0.44	0.73	0.72	0.73	0.28	0.78	0.78	0.78	0.22	0.85	0.85	0.85	0.15
cuttingexperts	0.82	0.42	0.55	0.58	0.86	0.84	0.85	0.16	0.77	0.77	0.77	0.23	0.93	0.93	0.93	0.07
hds-usa	0.82	0.57	0.67	0.43	0.78	0.72	0.75	0.28	0.76	0.76	0.76	0.24	0.9	0.9	0.90	0.1
madisontoolinc	0.86	0.75	0.8	0.25	0.84	0.83	0.83	0.17	0.88	0.88	0.88	0.12	0.91	0.91	0.91	0.09
Duratrack	0.57	0.43	0.49	0.57	0.69	0.67	0.68	0.33	0.64	0.64	0.64	0.36	0.83	0.83	0.83	0.17
nationgrinding	0.92	0.8	0.86	0.2	0.85	0.83	0.84	0.17	0.87	0.87	0.87	0.13	0.93	0.93	0.93	0.07
Schmidtool	0.82	0.74	0.78	0.26	0.77	0.74	0.75	0.26	0.88	0.88	0.88	0.12	0.9	0.9	0.9	0.1
Average	0.86	0.69	0.76	0.31	0.84	0.81	0.82	0.19	0.80	0.79	0.79	0.21	0.88	0.86	0.87	0.14

The objectives of applying the SR-KB are to help the instance annotator (1) accurately annotate as many instances as possible while at the same time (2) minimize human intervention. Thus, one important aspect of measuring the effects of SR-KB is to test whether the SR-KB helps the instance annotator achieve those objectives. The performance analysis based on the experimental results stated above demonstrated that the SR-KB successfully achieved the first objective. We use the correction rate to measure whether the SR-KB can help the instance annotator continuously decrease the human effort of correcting class labels of instances. This measurement is conducted both horizontally and vertically. That is, we compare the average correction rates as well as the trends of correction rate over 45 manufacturing web sites among the four strategies. The comparison on the average correction rates is presented in Table 7.3 while the comparison on the trends of correction rate is depicted in Figure 7.12.

Table 7.3 Comparison on average correction rates

Instance annotation strategy	Correction rate
Strategy one: None applied	0.30
Strategy two: MCMR applied	0.19
Strategy three: NBAC applied	0.21
Strategy four: SR-KB applied	0.13

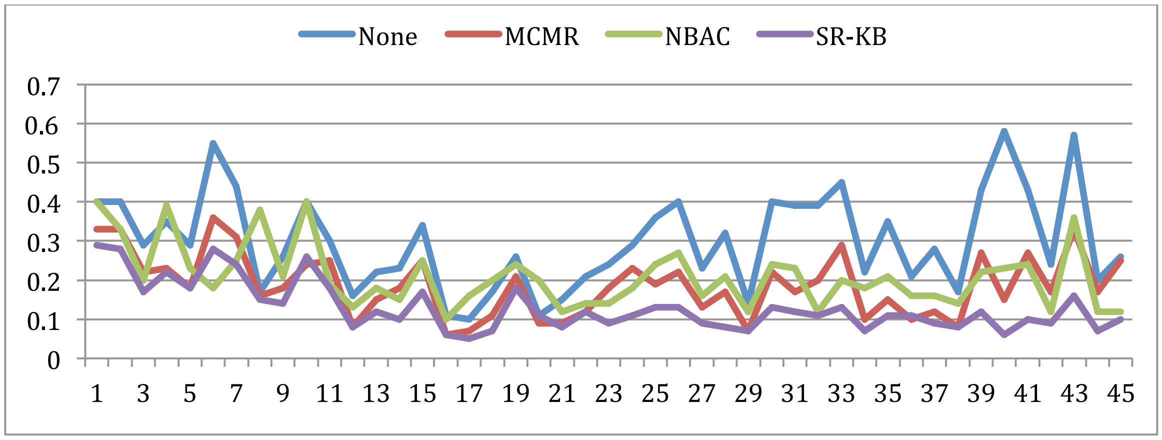


Figure 7.12 Comparison on correction rate among the four instance annotation strategies

Both Table 7.3 and Figure 7.12 clearly show that strategy four (SR-KB applied) has the best performance in terms of correction rate among the four strategies. More importantly, strategy four has a clear trend of declining correction rate as the instance annotation process goes by, as shown in Figure 7.13. This demonstrates that the SR-KB has the ability of continuously decreasing human intervention on validating annotated instances.

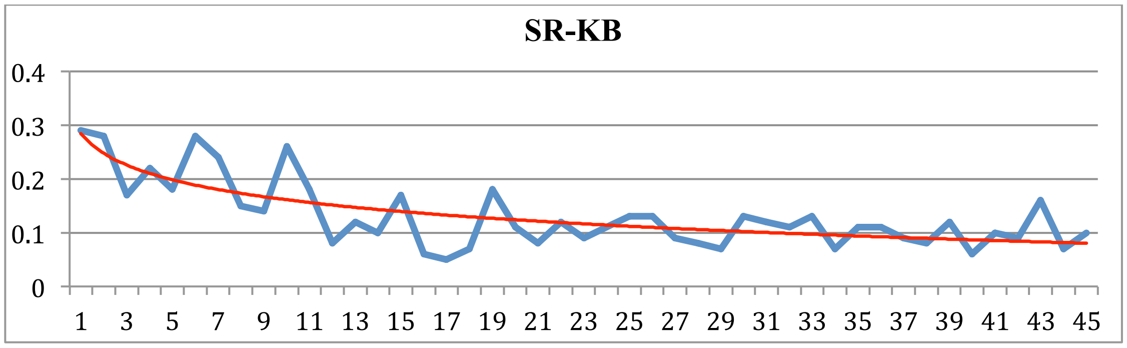


Figure 7.13 The trend of correction rate of the forth instance annotation strategy

Theoretically, both strategy three and four should be able to decrease the correction rate as the instance annotation process goes by because both of them utilize NBAC trained by validated instances to correct possibly wrong class labels of newly incoming instances. However, only strategy four shows a clear trend of declining correction rate while strategy three does not. The main reason leading to such different outcome is that the concepts of instances cannot be fully exploited in strategy three to create the conceptual mapping features because of terminological heterogeneity and as a result the class features created may not be able to well represent the relationship between classes and their instances. Consequently, the NBAC trained by these features may have high annotation error. On the other hand, strategy four exploits the MCMR to address the terminological heterogeneity issue when establishing mappings between concepts of instances and classes. As a result, more conceptual mapping features can be created to train the NBAC, thereby improving the annotation accuracy.

One issue with the previous experiment is that it is conducted in sequential order on a list of manufacturing web sites. Therefore, the outcome of annotating instances from earlier web site would affect the outcome of the subsequent ones. To test whether the performance improvement from SR-KB is robust, we need to remove its dependency on the order of web sites. In other words, we need to test whether the instance annotator is able to perform consistently on any ordering of the web-sites. For this purpose, we shuffle the 45 manufacturing web sites 10 times and conducted the same experiment on each of the 10 orderings. For each of the 10 tests, we divide the experiment into two stages. In the initial stage, we let the given strategy works through the first 40 web sites to allow the SR-KB to have opportunity to be trained sufficiently well. Thus, we test the robustness of SR-KB on the later stage. Specifically, we first compute the average of F1 measures of last five web sites for each of the 10 tests. For conciseness, we refer to the average of F1 measures as F1 average. Then, we examine the accuracy of SR-KB by comparing F1 averages of the 10 tests among the four instance annotation strategies. The result is shown in Table 7.4 and pictorially presented in Figure 7.14.

Table 7.4 Averages of F1 measures

	Strategy one: None	Strategy two: MCMR	Strategy three: NBAC	Strategy four: SR-KB
Test 1	0.708	0.816	0.788	0.892
Test 2	0.726	0.83	0.812	0.902
Test 3	0.81	0.878	0.83	0.912
Test 4	0.778	0.878	0.846	0.91
Test 5	0.762	0.872	0.818	0.898
Test 6	0.76	0.864	0.862	0.906
Test 7	0.758	0.886	0.836	0.914
Test 8	0.796	0.864	0.86	0.91
Test 9	0.722	0.84	0.832	0.898
Test 10	0.748	0.854	0.856	0.922

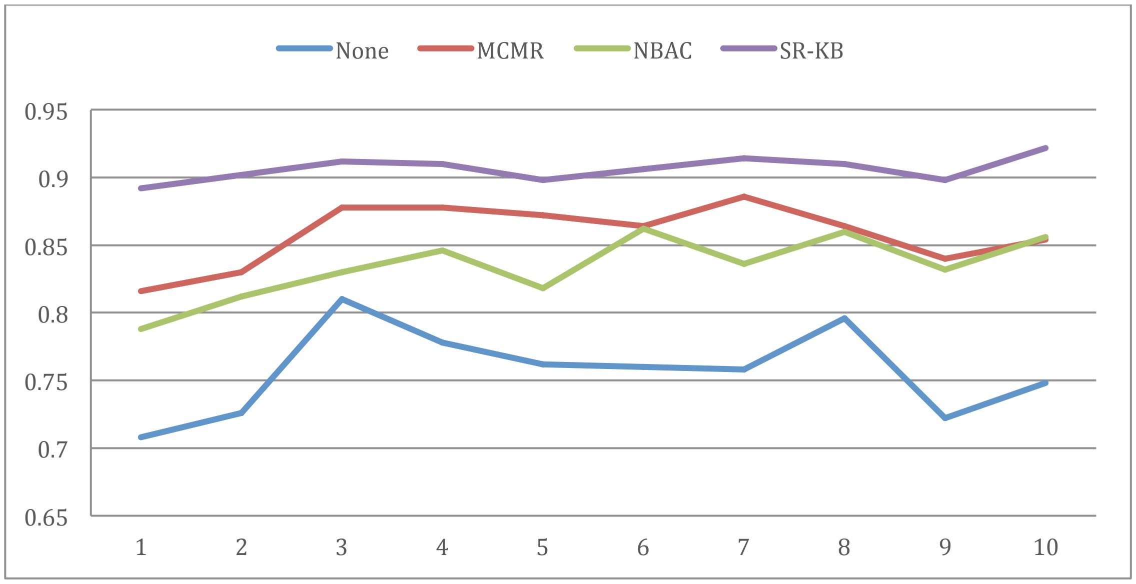


Figure 7.14 Comparison on averages of F1 measures for the 10 tests among the four instance annotation strategies

Figure 7.14 shows that the fourth strategy (SR-KB applied) has the highest F1 averages cross the 10 tests. This demonstrates that when SR-KB applied, the instance annotator can produce accurate annotation independently of order of inputted web sites. However, the F1 average along does not demonstrate that the SR-KB can help instance annotator consistently deliver accurate result that is independent of the order of web sites. In other words, we need to examine the stability of the SR-KB. To this end, we compute the standard deviation of F1 averages of the 10 tests for each of the four strategies. Then, we compare these standard deviations among the four instance annotation strategies. The result is presented in Table 7.5 and Figure 7.15.

Table 7.5 Standard deviations of F1 averages

Strategy one: None	Strategy two: MCMR	Strategy three: NBAC	Strategy four: SR-KB
0.032	0.023	0.023	0.009

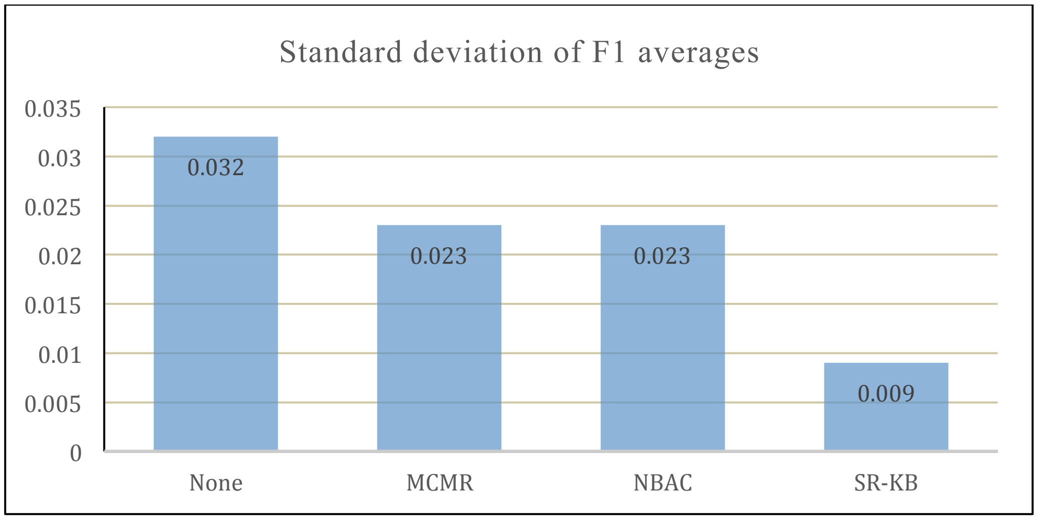


Figure 7.15 Comparison on standard deviations of F1 averages of 10 tests among the four instance annotation strategies

Figure 7.15 clearly shows that the fourth strategy (SR-KB applied) has the lowest standard deviation of F1 averages. This demonstrates that SR-KB can deal with various types of manufacturing service capabilities that different web sites may offer. In other words, the performance of the instance annotator does not change much when the order of the inputted web sites is modified. Figure 7.14 and Figure 7.15 together demonstrate that the instance annotator is robust to produce high accurate annotation when SR-KB applied.

In order to establish the base-line for the experimental evaluation on the semantic resolution framework and examine the performance of this framework under a stricter assessment criterion, we conducted an additional experiment with a more restricted version of the precision and recall. Specifically, when measuring the correctness of an annotated instances, rather than computing a proximity score between the expected class label and the assigned class label as shown in formula (6.8), the restricted version of the precision and recall returns a boolean value indicating that whether a annotation is either correct or incorrect depending on if an exact match between the system generated annotation and the grand truth. The restricted precision computes the ratio of correctly annotated instances to all instances that were automatically annotated by the instance annotator (possibly corrected by NBAC) while the restricted recall computes the ratio of correctly annotated instances to the instances extracted from websites and annotated by domain experts (serves as the ground truth). The experimental result is presented in Table 7.6.

Since the assessment criterion used here on the correctness of annotated instances is stricter than the one used in (6.8) for the relaxed precision and relaxed recall, the result shown in Table 7.6 is as expected worse than the one shown in Table 7.2. However, the difference between the two situations is not that great. In fact, the precision and recall only drop from 0.86 to 0.76 and from 0.69 to 0.62, respectively, if the knowledge base is not used (the first strategy). Moreover, the result shows a significant improvement on the restricted recall of forth strategy compared to that of the first strategy although the restricted precision does not change and the correction rate of the forth strategy still shows a trend of continuing declining as depicted in Figure 7.16.

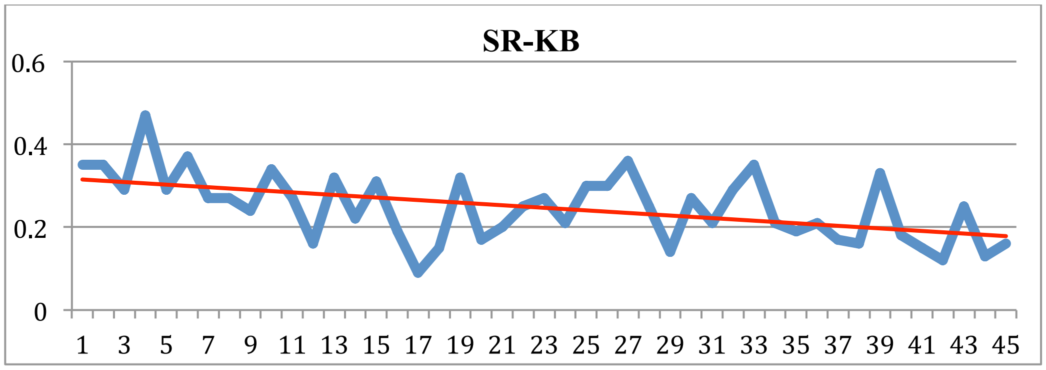


Figure 7.16 The trend of correction rate of the forth strategy with restricted precision and restricted recall applied

7.5 Summary

In this chapter, we explained in detail each component of the semantic resolution knowledge base (SR-KB) that aims to improve the annotation accuracy of instance annotator while at the same time minimize the human intervention. Specifically, manufacturing concepts mapping repository (MCMR) helps alleviate the issue of terminological heterogeneity existing in the instance annotation process, thereby improving the chance of assigning instances with accurate class labels, while Naïve Bayes-based annotation corrector (NBAC) corrects misannotated instances. The instance validation platform (IVP) is an assisting tool that helps domain experts with

Table 7.6 Comparison on restricted precision, restricted recall and correction rate among four strategies

Web Sites	Strategy one: None				Strategy two: MCMR				Strategy three: NBAC				Strategy four: SR-KB			
	Pre.	Rec.	F1	Cor.	Pre.	Rec.	F1	Cor.	Pre.	Rec	F1	Cor.	Pre.	Rec.	F1	Cor.
Schulermfg	0.74	0.56	0.64	0.44	0.75	0.65	0.7	0.35	0.74	0.56	0.64	0.44	0.75	0.65	0.7	0.35
Soundmfg	0.64	0.49	0.56	0.51	0.61	0.58	0.59	0.42	0.59	0.56	0.57	0.44	0.69	0.65	0.67	0.35
numericalconcepts	0.75	0.66	0.7	0.34	0.73	0.7	0.71	0.3	0.74	0.71	0.72	0.29	0.74	0.71	0.72	0.29
Basssettinc	0.57	0.44	0.5	0.56	0.57	0.53	0.55	0.47	0.43	0.4	0.41	0.6	0.57	0.53	0.55	0.47
Aerostarmfg	0.79	0.62	0.69	0.38	0.76	0.7	0.73	0.3	0.69	0.64	0.66	0.36	0.76	0.71	0.73	0.29
magnamachine	0.61	0.43	0.5	0.57	0.43	0.36	0.39	0.64	0.47	0.47	0.47	0.53	0.63	0.63	0.63	0.37
Ashleyward	0.63	0.45	0.53	0.55	0.57	0.57	0.57	0.43	0.54	0.54	0.54	0.46	0.73	0.73	0.73	0.27
Accutrex	0.81	0.71	0.76	0.29	0.74	0.73	0.73	0.27	0.55	0.53	0.54	0.47	0.77	0.73	0.75	0.27
Weaverandsons	0.83	0.66	0.74	0.34	0.82	0.75	0.78	0.25	0.67	0.65	0.66	0.35	0.78	0.76	0.77	0.24
Astromfg	0.47	0.39	0.43	0.61	0.59	0.58	0.58	0.42	0.69	0.64	0.66	0.36	0.66	0.66	0.66	0.34
Wisconsinmetalparts	0.67	0.6	0.63	0.4	0.73	0.73	0.73	0.27	0.71	0.7	0.71	0.3	0.73	0.73	0.73	0.27
Tmfincorporated	0.8	0.72	0.76	0.28	0.82	0.79	0.8	0.21	0.8	0.77	0.78	0.23	0.87	0.84	0.85	0.16
Robersontool	0.65	0.6	0.62	0.4	0.68	0.68	0.68	0.32	0.57	0.57	0.57	0.43	0.68	0.68	0.68	0.32
Spmfg	0.84	0.65	0.73	0.35	0.84	0.78	0.81	0.22	0.6	0.6	0.6	0.4	0.78	0.78	0.78	0.22
Smccontractmfg	0.73	0.61	0.66	0.39	0.68	0.65	0.66	0.35	0.72	0.7	0.71	0.3	0.71	0.69	0.7	0.31
Swiftglass	0.82	0.75	0.78	0.25	0.83	0.81	0.82	0.19	0.8	0.79	0.79	0.21	0.82	0.81	0.81	0.19
Ameristarmfg	0.89	0.84	0.86	0.16	0.89	0.89	0.89	0.11	0.8	0.8	0.8	0.2	0.91	0.91	0.91	0.09
Jnmetalproducts	0.86	0.77	0.81	0.23	0.88	0.86	0.87	0.14	0.78	0.77	0.77	0.23	0.87	0.85	0.86	0.15
Nobleindustries	0.76	0.65	0.7	0.35	0.68	0.68	0.68	0.32	0.67	0.67	0.67	0.33	0.68	0.68	0.68	0.32
Aalloy	0.84	0.81	0.82	0.19	0.83	0.83	0.83	0.17	0.71	0.71	0.71	0.29	0.83	0.83	0.83	0.17
Portersfab	0.8	0.72	0.76	0.28	0.77	0.76	0.76	0.24	0.82	0.8	0.81	0.2	0.82	0.8	0.81	0.2
Californiabrazing	0.79	0.65	0.71	0.35	0.76	0.73	0.74	0.27	0.69	0.67	0.68	0.33	0.77	0.75	0.76	0.25

Web Sites	Pre.	Rec.	F1	Cor.	Pre.	Rec.	F1	Cor.	Pre.	Rec	F1	Cor.	Pre.	Rec.	F1	Cor.
Columbiamanufacturing	0.74	0.62	0.67	0.38	0.77	0.71	0.74	0.29	0.66	0.65	0.65	0.35	0.75	0.73	0.74	0.27
Mantecservicesinc	0.76	0.65	0.7	0.35	0.67	0.66	0.66	0.34	0.73	0.72	0.72	0.28	0.8	0.79	0.79	0.21
Kilgoremfg	0.91	0.61	0.73	0.39	0.62	0.6	0.61	0.4	0.67	0.66	0.66	0.34	0.71	0.7	0.7	0.3
Elgeprecision	0.96	0.59	0.73	0.41	0.76	0.68	0.72	0.32	0.66	0.65	0.65	0.35	0.71	0.7	0.7	0.3
Dynamicprecision	0.66	0.54	0.59	0.46	0.64	0.64	0.64	0.36	0.57	0.57	0.57	0.43	0.64	0.64	0.64	0.36
Pbandm	0.78	0.64	0.7	0.36	0.7	0.7	0.7	0.3	0.71	0.71	0.71	0.29	0.75	0.75	0.75	0.25
Schenketool	0.8	0.76	0.78	0.24	0.88	0.88	0.88	0.12	0.81	0.81	0.81	0.19	0.86	0.86	0.86	0.14
Pierceindustries	0.78	0.58	0.67	0.42	0.74	0.68	0.71	0.32	0.66	0.66	0.66	0.34	0.73	0.73	0.73	0.27
Westfieldmachine	0.97	0.59	0.73	0.41	0.77	0.7	0.73	0.3	0.68	0.68	0.68	0.32	0.79	0.79	0.79	0.21
Ardelengineering	0.61	0.61	0.61	0.39	0.65	0.65	0.65	0.35	0.63	0.63	0.63	0.37	0.71	0.71	0.71	0.29
Hhsmm	0.59	0.59	0.59	0.41	0.59	0.59	0.59	0.41	0.53	0.53	0.53	0.47	0.65	0.65	0.65	0.35
Wiegeltoolworks	0.83	0.7	0.76	0.3	0.8	0.78	0.79	0.22	0.73	0.73	0.73	0.27	0.79	0.79	0.79	0.21
Adaptplastics	0.81	0.6	0.69	0.4	0.79	0.77	0.78	0.23	0.72	0.71	0.71	0.29	0.81	0.81	0.81	0.19
Mpi-dms	0.87	0.71	0.78	0.29	0.81	0.81	0.81	0.19	0.75	0.75	0.75	0.25	0.79	0.79	0.79	0.21
Contract-mfg	0.85	0.69	0.76	0.31	0.82	0.82	0.82	0.18	0.74	0.74	0.74	0.26	0.83	0.83	0.83	0.17
Hloeb	0.83	0.71	0.77	0.29	0.89	0.87	0.88	0.13	0.8	0.8	0.8	0.2	0.84	0.84	0.84	0.16
Milacronmachining	0.7	0.5	0.58	0.5	0.65	0.64	0.64	0.36	0.64	0.64	0.64	0.36	0.67	0.67	0.67	0.33
cuttingexperts	0.7	0.39	0.5	0.61	0.77	0.74	0.75	0.26	0.71	0.71	0.71	0.29	0.82	0.82	0.82	0.18
hds-usa	0.79	0.53	0.63	0.47	0.74	0.7	0.72	0.3	0.72	0.72	0.72	0.28	0.85	0.85	0.85	0.15
madisontoolinc	0.81	0.73	0.77	0.27	0.82	0.82	0.82	0.18	0.84	0.84	0.84	0.16	0.88	0.88	0.88	0.12
duratrack	0.57	0.43	0.49	0.57	0.62	0.62	0.62	0.38	0.6	0.6	0.6	0.4	0.75	0.75	0.75	0.25
nationgrinding	0.88	0.76	0.82	0.24	0.71	0.71	0.71	0.29	0.75	0.75	0.75	0.25	0.87	0.87	0.87	0.13
schmidtool	0.81	0.64	0.72	0.36	0.75	0.75	0.75	0.25	0.72	0.72	0.72	0.28	0.84	0.84	0.84	0.16
Average	0.76	0.62	0.68	0.38	0.73	0.71	0.72	0.29	0.68	0.67	0.67	0.33	0.76	0.75	0.75	0.25

no or little knowledge on ontology validate annotated instances and creates training data set for NBAC. The experimental results demonstrate that SR-KB not only significantly improves the quality and quantity of the annotated instance but also continuously reduces human intervention as the instance annotation process goes by.

Chapter 8.

Conclusion

The objective of this research is to develop a semantic resolution formwork (SRF) that is able to extract manufacturing service capability (MSC) instances from a broad range of manufacturing web sites that may present their MSC instances in different ways and resolve semantic heterogeneity while integrating these instances into a (or a set of) manufacturing domain ontology. To this end, our research supports two different tasks: instance extraction task and instance annotation task.

For the instance extraction task, we propose an instance extractor that builds an instance description model upon each manufacturer web site. Such IDM describes each extracted instance n_i with a context that includes concepts instance n_i might belong to and relations connecting n_i to other instances or numerical data values. Our instance extraction approach does not require manually created rules or training data. Neither does it require web pages to be built upon regular templates. It extracts MSC instances and forms their context by exploiting the common structure of and redundancy [12] in manufacturing web pages.

For the instance annotation task, we propose an instance annotator that annotates instances with classes defined in the manufacturing domain ontology based on contexts of these instances and the subsumption relationship defined in MDO. The experimental result shows that our approach is effective in accurately assign class labels to instances, so long as the concepts in the context of these instances can be mapped to classes defined in MDO. However, due to terminological heterogeneity, our approach has difficulty in mapping all the concepts in contexts of instances to classes defined in MDO, causing misannotations for some extracted instances. Besides, both the instance extractor and the instance annotator provide no mechanism that is able to correct misannotated instances. We propose a semantic resolution knowledge base (SR-KB) to address these drawbacks. This SR-KB contains a manufacturing concepts mapping repository (MCMR), a Naïve Bayes-based annotation corrector (NBAC) and an instance validation platform (IVP). The MCMR stores a collection of automatically established mappings between manufacturing concepts and classes defined in MDO aiming to address terminological heterogeneity in the instance annotation process. The IVP facilitates the participation of domain experts to validate annotated instances and correct misannotated ones. The goal of the NBAC is to automatically correct misannotated instances based on corrections made by domain experts with the assistance of the IVP. As demonstrated by the experimental results presented in Chapter 7, we conclude that SR-KB is able to improve both relaxed precision and relaxed recall of annotated instances while at the same time reduce human involvement. To further improve the performance of the semantic resolution framework, we have identified several directions for future work.

Parameter selection and sensitivity analysis. Before running the semantic resolution framework, we need to select a set of parameters such as the similarity thresholds used in clustering relation and annotating instance. We plan to develop methodical approaches that can help us identify a satisfactory set of parameters, if optimal set is not impractical. This work is to be accompanied by sensitivity analysis of selected parameters on the system performance and robostness.

Improvements on instance extractor. Certain improvements to instance extraction process have the potential to improve the SRF's overall relaxed precision and relaxed recall. In the current state of the instance extractor, we only consider short phrases and sentence as instances of interest. In the next version of the instance extractor, we may integrate a Web-scale named-entity recognizer such as Lex [90] to help identify entities of interest in raw unstructured texts. Moreover, the instance extractor currently can only extract instances from row header tables (§5.2). This can be extended by adopting other table parse solutions [45, 46] for extracting instances from tables with different structures.

Improvements on the semantic resolution knowledge base. The semantic resolution knowledge base (SR-KB) adopts Naïve Bayes classifier to correct possible misannotated instances. The training features used to train the Naïve Bayes classifier are assumed to be independent with each other. In fact, however, some conceptual mapping features (§7.2) of an instance apparently are dependent of each other. To better capture these possibilities, we may leverage more sophisticated classifiers such as Bayesian Network [91] and Support Vector Machine [92] to correct misannotated instances. The instance validation platform currently only allows users to correct class labels of instances that have been annotated by instance annotator. In the next version of the instance validation platform, we will add new features that allow users to create and add new instances and their contexts. This is particularly useful for the website builders who also want to semantically annotate the MSC information. Also, we will integrate other information and knowledge sources such as domain-specific corpus and other statistical data, WordNet, ontologies, linked data, and other forms of semantic information into the SR-KB in addition to the manufacturing concepts mapping repository such that more valuable data published on websites can be better identified, annotated and integrated by our semantic resolution framework.

Practical application on semantic resolution framework. Linked open data (LOD) has achieved success in a wide variety of fields and is supported by a comparably large and active user community. However, to the best of our knowledge, no single data set related to manufacturing domain is available in the LOD cloud. Therefore, publishing linked data of manufacturing domain to the LOD cloud would be a great contribution to both manufacturing and Linked Data community. The manufacturing knowledge base, the output of our semantic resolution framework, could serve as the basis for that purpose. This manufacturing knowledge base, when published as linked data, can be leveraged by real world applications to help accurately match customers and suppliers, or agilely form manufacturing supply chains.

We will also examine the applicability of the semantic resolution framework and its methods outside the manufacturing domain. The relatively good performance of this framework relies on, among other things, its ability to utilize the structure of the manufacturing websites. This leads us to believe that our framework would be applicable to other service-oriented web sites, such as travailing, restaurant, and healthcare where the services they offer are typically described in detailed and structured ways similar to manufacturing websites. It would of great interest to see how this framework and its methods apply to sample websites of a selected domain, such as the healthcare service domain or some of its subdomains. The data collected from the experiments may provide valuable hints on improving the generality and accuracy of the framework as a whole or of its individual methods.

Bibliography

[1] T. Sturgeon, "Modular Production Networks: A New American Model of Industrial Organization," in *Industrial and Corporate Change*, Baltimore, 2002.

[2] Li, C.S., Chang, Y.C., and Smith, J.R., "An E-marketplace Infrastructure for Information," in *Intelligent Multimedia, Video and Speech Processing*, 2001.

[3] Kim, J., Peng, Y., Ivezic, N., and Shin, J., "An Optimization Approach for Semantic-Based XML Schema Matching," in *International Journal of Trade, Economics, and Finance*, Finance, 2011.

[4] S. Grimes, "Semantic Web Business: Going Nowhere Slowly," 7 1 2014. [Online]. Available:http://www.informationweek.com/software/information-management/semantic-web-business-going-nowhere-slowly/d/d-id/1113323. [Accessed 16 11 2014].

[5] Janev, V. and Vranes, S., "Semantic Web Technologies: Ready for Adoption?," *IT Professional ,* vol. 11, no. 5, pp. 8-16, 2009.

[6] P. Gregorowicz, "The Semantic Web and Your Intranet," 19 9 2007. [Online]. Available: http://www.dataversity.net/the-semantic-web-and-your-intranet/. [Accessed 16 11 2014].

[7] Amit S., Cartic R. and Christopher T. , "Semantics for the Semantic Web: The Implicit, the Formal and the Powerful," *International Journal on Semantic Web & Information System,* vol. 1, pp. 1-18, 2005.

[8] Charnyote P. and Joachim H., "A Classification Scheme for Semantic and Schematic Heterogeneities in XML Data Sources," University of Florida, Gainesville, 2000.

[9] M. Garcia-Solaco, F. Saltor and M. Castellanos, Semantic heterogeneity in multidatabase systems, Printice-Hall, 1996.

[10] Halevy, A., "Why Your Data Won't Mix," *ACM Queue,* vol. 3, no. 8, 2005.

[11] D. C. Downey, Redundancy in Web-sacle Information Extraction: Probabilistic Model and Experimental Results, Seattle: university of washington, 2008.

[12] Boer V. de, Someren M. van and Wielinga B. J., "Extracting Instances of Relations from Web Documents Using Redundancy," *The Semantic Web:*

Research and Applications, vol. 4011, pp. 245-258, 2006.

[13] Banko M, Cafarella M. J., Soderland S., Broadhead M and Etzioni O., "Open information extraction from the web," *Communications of the ACM,* vol. 51, no. 12, pp. 68-74, 2008.

[14] Shvaiko P. and Euzenat J., Ontology Maching, Springer, 2010.

[15] Trung-Kien T. and Christophe D., "Towards Using OWL Integrity Constraints in Ontology Engineering," in *On the Move to Meaningful Internet System: OTM 2012 Workshops*, 2012.

[16] M. B., "The Open World Assumption: Elephant in the Room," 21 December 2009. [Online]. Available: http://www.mkbergman.com/852/the-open-world-assumption-elephant-in-the-room/.

[17] Tim B. Lee, James H. and Ora L., "The Semantic Web," *Scientific American,* pp. 29-37, 2001.

[18] Markus K., František S., and Ian H., A Description Logic Primer, arxiv.org, 2012.

[19] Nicola G, Daniel O. and Steffen S., "What Is an Ontology?," in *Hankbook on Ontologies International Handbooks on Information Systems*, Springer, 2009, pp. 1-17.

[20] Golbreich C., Zhang S. and Bodenreider O., "The Foundational Model of Anatomy in OWL: Experience and Perspective," *Web Semantics: Science, Services and Agents on the World Wide Web,* vol. 4, no. 3, pp. 181-195, 2006.

[21] Delaborda C. P. and Conrad S., "Relational OWL: a data and a schema representation format based on OWL," in *Proceedings of the 2nd Aisa-Pacific conference on conceptual modeling*, 2005.

[22] Ian H. and Peter F., "A proposal for an owl rules language," in *Proceedings of the 13th international conference on World Wide Web*, New York, 2004.

[23] T. Berners-Lee, "Linked Data," 27 7 2006. [Online]. Available: http://www.w3.org/DesignIssues/LinkedData.html.

[24] R. Klischewski, "Top down or bottom up? How to Extablish a Common Ground for Sematic Interoperability with e-Government Communities," *E-Government: Modelling Norms and Concepts as Key Issues, Proceedings of 1st International Workshop on E-Government at ICAIL,* pp. 17-26, 2003.

[25] Wache H. , Vögele T., Visser U., Stuckenschmidt H. , Schuster G., Neumann H. and Hübner S., "Ontology-Based Integration of Information - A Survey of Existing Approaches," in *Proceedings of Ontology-based Integration*, 2001.

[26] N. F. Noy, "Semantic Integration: A Survey Of Ontology-Based Approaches," *SIGMOD Record,* vol. 33, no. 4, pp. 65-70, 2004.

[27] Micheal U. and Micheal G. , "Ontologies and semantics for seamless connectivity," *ACM SIGMOD Record,* vol. 33, no. 4, pp. 58-64, 2004.

[28] Kaiser K. and Miksch S., Information Extraction: a Survey, Vienna, Austria: Institute of Software Technology & Interactive Systems, 2005.

[29] Alberto H. F. Laender, Berthier A. Ribeiro-Neto, Altigran S. da Silva and Juliana S. Teixeira, "A Brief Survey of Web Data Extraction Tools," *ACM SIGMOD Record,* vol. 31, no. 2, pp. 84-93, 2002.

[30] Chang C. H., Kayed M. and Girgis M. R., "A Survey of Web Information Extraction System," in *IEEE Transactions on Knowledge and Data Engineering*, 2006.

[31] Russell S. and Norvig P., Artificial Intelligence: A Modern Approach, Prentice-Hall, 2003.

[32] R. Grishman, Information Extraction: Capabilities and Challenges, Tarragona, Spain: Rovira i Virgili University, 2012.

[33] Prakash M Nadkami, Lucila Ohno-Machado and Wendy W. Chapman, "Natural Language Process: an Introduction," *J Am Med Inform Assoc,* vol. 18, no. 5, pp. 544-551, 2011.

[34] Petasis G., Karkaletsis V, Paliouras G, Krithara A. and Zavitsanos E, "Ontology Population and Enrichment: State of the Art," in *Knowledge-Driven Multimedia Information Extraction and Ontology Evolution*, 2011.

[35] Ciravegna F. and Wilks Y., "Designing Adaptive Information Extraction for the Semantic Web in Amilcare," in *Annotation for Semantic Web, Frontier in Artificial Intelligence and Applications*, 2003.

[36] Cunningham H., Maynard D., Tablan V., Ursu C. and Bontcheva K., "Developing Language Processing Components with GATE," in *Proceedings of the 5th Conference on Applied Natural Language Processing*, 1997.

[37] M. Moens, Information Extraction: Algorithms and Prospects in a Retrieval

Context, New York: Springer, 2003.

[38] Agichtein E. and Gravano L., "Snowball: extracting relations from large plain-text collections," in *Proceedings of the fifth ACM conference on digital libraries*, New York, 2000.

[39] Shinyama Y. and Sekine S., "Preemptive Information Extraction using Unrestricted Relation Discovery," in *Proceedings of the main conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*, Stroudsburg, 2006.

[40] Meyers A., Grishman R., Kosaka M. and Zhao S., "Covering Treebanks with GLARF," in *Proceedings of the ACL 2001 Workshop on Sharing Tools and Resources*, Stroudsburg, 2001.

[41] Handschuh S., Staab S. and Ciravegna F., "S-CREAM - Semi-automatic CREAtion of Metadata," in *Proceedings of the 13th International Conference on Knowledge Engineering and Knowledge Management. Ontologies and the Semantic Web*, London, 2002.

[42] Chang C. H. and Lui S. C., "IEPAD: Information Extraction based on Pattern Discovery.," in *Proceedings of the Tenth International Conference on World Wide Web*, Hong Kong, 2001.

[43] Crescenzi V., Mecca G. and Merialdo P., "RoadRunner: Towards Automatic Data Extraction from Large Web Sites," in *Proceedings of the 26th International Conference on Very Large Database Systems*, Rome, 2001.

[44] Yoshida M. Torisawa K. and Tsujii J., "Extracting Ontologies from World Wide Web via HTML Tables," in *In Proceedings of the Pacific Association for Computational Linguistics*, 2001.

[45] Mulwad V, Finin T. and Joshi A., "Automatically Generating Government Linked Data from Tables," in *In working notes of the AAAI Fall Symposium on Open Government Knowledge: AI Opportunities and Challenges*, 2011.

[46] Mulwad V., Finin T. and Joshi A., "Interpreting Medical Tables as Linked Data to Generate Meta-Analysis Reports," in *15th IEEE International Conference on Information Reuse and Integration*, Las Vegas, 2014.

[47] Daya, W. and Dou, D., "Ontology-based information extraction: An introduction and a survey of current approaches," *Journal of Information Science,* vol. 36(3), pp. 306-323, 2010.

[48] H. Saggion, A. Funk, D. Maynard, and K. Bontcheva 2007, "Ontology-based

information extraction for business intelligence," in *In proceedings of the 6th International and 2nd Asian Semantic Web Conference*, Berlin, 2007.

[49] Vargas-Vera M., Motta E., Domingue J., Shum S. B. and Lanzoni M., "Knowledge Extraction by Using an Ontology-based Annotation Tool," in *Proceedings of the Workshop on Knowledge Markup and Semantic Annotation*, New York, 2001.

[50] Navigli R and Velardi P, "Ontology Enrichment Through Automatic Semantic Annotation of On-Line Glossaries," in *Managing Knowledge in a World of Networks*, 2006.

[51] F. Wu and D.S. Weld, "Autonomously semantifying Wikipedia," in *In Proceedings of the Sixteenth ACM Conference on Information and Knowledge Management*, New York, 2007.

[52] P. Cimiano, S. Handschuh, and S. Staab, "Towards the self-annotating web," in *In Proceedings of the 18th ACM Conference on Information and Knowledge Management*, New York, 2009.

[53] Kietz J., Maedche A. and Volz R., "A Method for Semi-automatic ontology acquisition from a corporate intranet," in *Proceedings of the EKAW'00 Workshop on Ontologies and Text*, Berlin, 2000.

[54] Maynard D., Peters W. and Li Y., "Metrics for Evaluation of Ontologies-based Information Extraction," in *Proceedings of the WWW 2006 Workshop on Evaluation of Ontologies for the Web*, New York, 2006.

[55] Wimalasuriya D.C. and Dou D., "Using Multiple Ontologies in Information Extraction," in *Proceedings of the 18th ACM Conference on Information and Knowledge Management*, New York, 2009.

[56] Popov B., Kiryakov A., Kirilov A., Manov D., Ognyanoff D. and Goranov M., "KIM - Semantic Annotation Platform," in *Proceedings of the 2nd International Semantic Web Conference*, Berlin, 2003.

[57] Popov B., Kiryakov A., Manov D., Ognyanoff D. and Goranov M., "KIM-a Semantic Annotation Platform for Information Extraction and Retrieval," *Natural Language Engineering,* vol. 10, pp. 375-392, 2004.

[58] P. Buitelaar and M. Siegel, "Ontology-based Information Extraction with SOBA," in *In Proceedings of the Fifth International Conference on Language Resources and Evaluation* , Genoa, Italy, 2006.

[59] Dividino R., Romanelli M. and Sonntag D., Semantic-based Ontology

Evaluation Tool (S-OntoEval), Marrakech: European Language Resources Association, 2008.

[60] Paliouras G., Spyropoulos C. D. and Tsatsaronis G., "Bootstrapping Ontology Evolution with Multimedia Information Extraction," in *Knowledge-driven Multimedia Information Extraction and Ontology Evolution*, Berlin, 2011.

[61] Shvaiko P. and Euzenat J., "A Survey of Schema-based Matching Approaches," *Data Semantics,* vol. 4, pp. 146-171, 2005.

[62] Choi N., Song I. Y. and Hyoil H., "A Survey on Ontology Mapping," *SIGMOD Record,* vol. 35, no. 0163-5808, pp. 34-41, 2006.

[63] Li M., Chen X., Li X., Ma B. and Paul M. B. Vitanyi, "The Similarity Metric," *IEEE Transactions on Information Theory,* vol. 50, no. 12, pp. 3250-3264, 2004.

[64] Euzenat J. and Valtchev P., "Similarity-based Ontology Alignment in OWL-Lite," in *Proceedings of the 16th European Conference on Artificial Intelligence ECAI*, 2004.

[65] Cheatham M. and Hitzler P., "String Similarity Metrics for Ontology Alignment," *The Semantic Web ,* vol. 8219, pp. 294-309, 2013.

[66] West R. and Turner Lynn, Information Theory, McGraw-Hill, 2013.

[67] Wu Z. and Embley D., "Verb Semantics and Lexical Selection," in *Proceedings of 32th Annual Meeting of the Association for Computational Linguistics*, Las Cruces, 1994.

[68] Madche A. and Zacharias V., "Clustering Ontology-based metadata in the Semantic Web," in *Proceedings of 6th European Conference on Principles and Pratice of Knowledge Discovery in Databases*, Helsinki, 2002.

[69] Niwattanakul S., Singthongchai J., Naenudorn E. and Wanapu S., "Using of Jaccard Coefficient for Keywords Similarity," in *Proceedings of the International MultiConference of Engineers and Computer Scientists*, Hong Kong, 2013.

[70] Giunchiglia F. and Shvaiko P., "Semantic Matching," in *Proceedings of Workshop on Ontologies and Distributed Systems*, Acapulco, 2003.

[71] Hebeler J., Fisher M., Blace R. and Andrew P. L., Semantic Web Programming, John Wiley & Sons, 2011.

[72] Christiane F. and Randee T., "WordNet," 2005. [Online]. Available:

http://wordnet.princeton.edu/.

[73] L. Bergroth, H. Hakonen, and T. Raita, "A Survey of Longest Common Subsequence Algorithms," in *In Proceedings of the Seventh International Symposium on String Processing Information Retrieval*, Washington, DC, USA, 2000.

[74] Banerjee S. and Pedersen T., "The Design, Implementation and Use of the Ngram Statistics Package," in *Proceedings of the 4th International Conference on Computational Linguistics and Intelligent Text Processing*, Berlin, 2003.

[75] Mitra P. and Wiederhold G., "Resolving Terminological Heterogeneity in Ontologies," in *Workshop on Ontologies and Semantic Interoperability at the 15th European Conference on Artificial Intelligence*, 2002.

[76] Han L., Kashyap A. L., Finin T., Mayfield J. and Weese J., "UMBC_EBIQUITY-CORE: Semantic Textual Similarity Systems," in *Proceedings of the Second Joint Conference on Lexical and Computational*, 2013.

[77] Hitzler P., Krotzsch M. and Rudolph S., Foundations of Semantic Web Technologies, CRC Press, 2009.

[78] Poole D. and Mackworth A., Artificial Intelligence: Foundations of Computational Agents, Cambridge University Press, 2010.

[79] S. Grimm, "A Unifying Formal Ontology Model - A Simple Formal Model for Unifying the Presentation of Ontologies in Semantic Web Research," in *KEOD*, 2009.

[80] W3C HTML Working Group, "The Extensible HyperText Markup Language," [Online]. Available: http://www.w3.org/TR/xhtml1/.

[81] XML Activity Lead, "Extensible Markup Language," [Online]. Available: http://www.w3.org/XML/.

[82] James C. and Steve, D., "XML Path Language," [Online]. Available: http://www.w3.org/TR/xpath/#dt-document-order.

[83] Fader A., Soderland S. and Etzioni O., "Identifying relations for open information extraction," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, Stroudsbury, PA, USA, 2011.

[84] Jain A. K., Murty M. N. and Flynn P. J., "Data Clustring: A Review," in *ACM*

Computing Surveys, 1999.

[85] Christopher D. M., Prabhakar R. and Hinrich S., Introduction to Information Retrieval, Cambridge University Press, 2008.

[86] Ameri F. and Dutta, D., "An Upper Ontology for Manufacturing Service Description," in *ASME International Design Engineering Technical Conferences & Computers and Information in Engineering Conference*, Philadelphia, 2006.

[87] M. Ehrig and J. Euzenat, "Relaxed precision and recall for ontology matching," in *In Proceedings of K-CAP Workshop on Integrating Ontologies*, Banff, CA, 2005.

[88] S. R. Eliason, Maximum Likelidhood Estimation: Logic and Practice, Sage Publications, 1993.

[89] S. Aggarwal, "Naive Bayes Classifier with Various Smoothing Techniques for Text Documents," *Computer Trends and Technology,* vol. 4, no. 4, pp. 873-876, 2013.

[90] Downey D., Broadhead M. and Etzioni O., "Locating complex named entities in web text," in *Proceedings of IJCAI*, 2007.

[91] A. Darwiche, Modeling and reasoning with Bayesian networks, Cambridge, 2009.

[92] Cristianini N. and John S. T., An Introduction to Support Vector Machines and Other Kernel based Learning Method, Cambridge, 2000.

[93] R. Romano, L. Rokach, and O. Maimon, "Automatic discovery of regular expression patterns representing negated findings in medical narrative reports," in *In proceedings of the 6th International Workshop on Next Generation Information Technologies and Systems*, Berlin, 2006.

[94] Kang, Y., Kim, J., and Peng, Y, "Extensible Dynamic Form Approach for Supplier Discovery," in *In Proceedings of Information Reuse and Integration, 2011 IEEE International Conference*, Las Vegas, NV, 2011.

Appendices

Appendix A – List of Abbreviations

- CC – Concept-to-Class mapping features
- CWA – Closed World Assumption
- DOM – Document Object Model
- HTML – Hypertext Markup Language
- IDM – Instance Description Model
- IE – Information Extraction
- IVP – Instance Validation Platform
- LCS – Longest Common Sequence
- LOD – Linked Open Data
- MCMR – Manufacturing Concepts Mapping Repository
- MDO – Manufacturing Domain Ontology
- MLE – Maximum Likelihood Estimate
- MSC – Manufacturing Service Capability
- NBAC – Naïve Bayes-based Annotation Corrector
- OBIE – Ontology-based Information Extraction
- OMBM – Ordered Maximum-weighted Bipartite Matching algorithm
- OWA – Open World Assumption
- OWL – Web Ontology Language
- RDF – Resource Description Framework
- RP – Relation-to-Property mapping features
- SRF – Semantic Resolution Framework
- SR-KB – Semantic Resolution Knowledge Base
- XHTML – Extensible Hypertext Markup Language
- XML – Extensible Markup Language