UNIVERSITÀ DEGLI STUDI DI NAPOLI FEDERICO II

PH.D. THESIS IN

INFORMATION TECHNOLOGY AND ELECTRICAL ENGINEERING

A MULTI-STRATEGY METHODOLOGY FOR ONTOLOGY INTEGRATION AND REUSE

INTEGRATING LARGE AND HETEROGENEOUS KNOWLEDGE BASES IN THE RISE OF BIG DATA

ENRICO GIACINTO CALDAROLA

TUTOR: PROF. ANTONIO PICARIELLO, ANTONIO M. RINALDI

XXIX CICLO

SCUOLA POLITECNICA E DELLE SCIENZE DI BASE
DIPARTIMENTO DI INGEGNERIA ELETTRICA E TECNOLOGIE DELL'INFORMAZIONE

I dedicate my dissertation work to my family and many friends.
A special feeling of gratitude to my loving wife, whose words of
encouragement and push for tenacity ring in my ears.

The Eagle soars in the summit of Heaven,
The Hunter with his dogs pursues his circuit.
O perpetual revolution of configured stars,
O perpetual recurrence of determined seasons,
O world of spring and autumn, birth and dying!
The endless cycle of idea and action,
Endless invention, endless experiment,
Brings knowledge of motion, but not of stillness;
Knowledge of speech, but not of silence;
Knowledge of words, and ignorance of the Word.
All our knowledge brings us nearer to death,
But nearness to death no nearer to God.
Where is the Life we have lost in living?
Where is the wisdom we have lost in knowledge?
Where is the knowledge we have lost in information?
The cycles of heaven in twenty centuries
Brings us farther from God and nearer to the Dust.
The lot of man is ceaseless labor,
Or ceaseless idleness, which is still harder,
Or irregular labour, which is not pleasant.
I have trodden the winepress alone, and I know
That it is hard to be really useful, resigning
The things that men count for happiness, seeking
The good deeds that lead to obscurity, accepting
With equal face those that bring ignominy,
The applause of all or the love of none.
All men are ready to invest their money
But most expect dividends.
I say to you: Make perfect your will.
I say: take no thought of the harvest,
But only of proper sowing.
The world turns and the world changes,
But one thing does not change.
In all of my years, one thing does not change,
However you disguise it, this thing does not change:
The perpetual struggle of Good and Evil.

from ”The Rock” by Thomas S. Eliot

Acknowledgments

I would like to acknowledge my advisors Professors Antonio Picariello and Antonio M. Rinaldi for guiding and supporting me over the doctoral period. You have set an example of excellence as a researcher, mentor, instructor, and role model. I would also like to acknowledge all professors I've encountered along the way in several courses, modules and seminars attended at the University of Naples "Federico II" and in several doctoral schools I took part. All of them have contributed somehow to my personal and professional growth.

I would like to acknowledge Dr. Marco Sacco from the Italian National Council of Researches (Institute of Industrial Technologies and Automation) for supporting and fostering my choice to dedicate a sustained effort to this valuable and challenging training route.

Finally, I would like to acknowledge all my doctoral fellows for their support, advices and encouragement; without them would miss the most funny and lighthearted part of this route.

Contents

List of Figures

List of Tables

Introduction

How topical is the poem by Thomas Eliot in the current epoch! Technological advances and changes all around the world have brought us to an unprecedented abyss of knowledge. We are confident to understand the Universe, from the infinite to the infinitesimal. We gauge everything, collect information about all events, facts and aspects of the whole known reality. Since the early years of the information technology advent, we have been collecting a plethora of data, but now we are more and more greedier in doing so. Scarcely a day goes without an info-graphic pops out showing how the amount of data produced in the last year has doubled the one produced from the dawn of mankind until the last year. It can be imagined as an application of the Moore's law to data growth: the exponential growth of hardware processing capabilities has led to a correlative explosion in data. It is not just computers generating such humongous quantity of data, it's all sorts of hardware, instruments, or sensors. For example, a single Boeing jet engine generates 10 terabytes of information for each 30 minutes it operates. That means a single trans-Atlantic flight of a conventional passenger jet creates 640 terabytes of data. Not to mention Facebook or Twitter, which are extremely known cases. This huge cloud of data can be imagined as another spherical shell surrounding the Earth planet, a kind of *datasphere* (similar to the atmosphere). It is the set of technologies used to store and communicate data over the Internet, data shared among people or computers and analytics performed over them. This sphere grows over time as new data are delivered by different sources, new complex algorithms are applied for crawling data over the Internet and analytics tools are used to process and generate new knowledge. In this scenario, it is natural the paradigm shift from *datasphere* to the *knowledgesphere*, i.e., the sphere of human knowledge, encompassing knowledge about facts, things, people, which represents the substratum of the collective intelli-

gence of the Semantic Web. This new revolutionary web has augmented the previous one by promoting common data formats and exchange protocols in order to provide a common framework that allows data to be shared and reused across application, enterprise, and community boundaries. This revolution, together with the increasing digitization of the world, has led to a high availability of data and schema models, formalized through different languages, which span throughout a wide range of knowledge domains and applications. As more and more outbreaks of this new revolution light up, a major challenge came soon into sight: addressing the main objectives of the Semantic Web, the sharing and reuse of data, demands effective and efficient methodologies to mediate, integrate and reuse existing heterogeneous knowledge models.

Taking into account the above considerations and acknowledging that ontologies are the de facto standard in representing and sharing knowledge models over the web, the work presented in this dissertation proposes a multi-strategy methodology to ontology integration and reuse, based on different matching techniques. The methodology is based on a software framework that contemplates different stages and includes different components. It starts with the harvesting of *source knowledge models* from the Internet or from specific repositories, evaluates them based on a set of qualitative criteria for the first selection and applies some linguistic and semantic similarity measures in order to individuate the models that best fit the *target model*. After that, the framework provides an alignment for the input and the target models and uses this to integrate the ones in the other. Starting from a literature review of the main existing approaches and methodologies, it will be demonstrated how the adoption of such approach with the help of an *ad hoc* software framework can improve and simplify the creation of new ontology models, automatizing as much as possible the ontology creation task and promoting the knowledge reuse. Although the proposed approach tries to reduce the human intervention in all ontology integration phases, does not neglect it, neither considers it as en element of weakness; on the contrary, user involvement is considered an essential *tuner* for the whole strategy as it incorporates precious user knowledge in the framework making it more effective. The proposed approach will be applied to the *Food* domain, specifically the *Food Production* domain, by collecting and subsequently analysing some of the most spread knowledge models available in the literature. Nevertheless, the approach does not loose generality and can be applied to other knowledge domains.

This dissertation is structured as follows:

Chapter 1, after a brief historical background about solutions at the dawn of the Semantic Web, introduces some ground concepts and definitions in the *ontology matching and integration* landscape and reviews the main ontology integration and reuse methodologies existing in the literature. A subsection is also dedicated to information visualization techniques, specifically concerning the graph visualization, being this an important aspect of the work.

Chapter 2 describes the proposed multi-strategy methodology for ontology integration and reuse. The methodology is based on a software framework that contemplates different stages and includes different components. It starts with the harvesting of *source knowledge models* (also referred as *input models* or *reference models*) from the Internet or from specific repositories, evaluates them based on a set of qualitative criteria for the first selection and applies some linguistic and semantic similarity measures in order to individuate the models that best fit the *target model*. The chapter is structured as follows: the first section is an high-level outline of the framework with a description of the functionalities provided by each component (each in its own subsection), while the successive sections focus on the matching, aligning and integrating methodologies respectively. The chapter ends with a separate section describing how the target model has been created and the reasons why it has been introduced in the framework.

Chapter 3 provides the implementation details concerning the software framework supporting the proposed ontology integration methodology. Starting from an overall view of the framework, the technological solutions along with the third-party libraries and tools eventually used to overcome the encountered issues and to implement the framework functionalities will be detailed.

Chapter 4 applies the methodology proposed in this work to the *food* domain, specifically to the *production of food*, encompassing concepts like *food product* and *food product categories*. The chapter is structured as follows: the first section introduces the domain under study by further characterizing it, while, from the second section to the final one, all the methodology's phases described in chapter 2 are applied to the case-study along with the considerations that have eventually arisen.

Chapter 5 describes how the proposed methodology has been evaluated and how the experimental results have been obtained from the case-study described in the previous chapter. The first section introduces the evaluation architecture, detailing the components involved in this task and recalling the matching strategy described in chapter 2. Later on, a section is dedicated to the definition of the Recall and Precision measures (both relaxed and semantically-grounded) used to evaluate the performance of the Matcher and the Aligner components. Another section is dedicated to motivate the involvement of users in the evaluation strategy, in order to obtain a *reference alignment* (here considered as a *ground truth*) for training the classifier function. In this section, a GUI helping the user in making a ground alignment is also described. Finally, the last section shows the performance of the methodology considered in its entirety, i.e., how effective and efficient it is for a knowledge reuse perspective.

Chapter 6 discusses the outcomes of the methodology highlighting strengths and weaknesses in terms of efficiency and effectiveness of the entire approach.

Chapter 7 draws the conclusion summarizing the major findings.

Chapter 1

Ontology integration, reuse and visualization. A state-of-the-art

In this chapter, after an historical background about disciplines and concepts from which the Semantic Web and *ontologies* have emerged in computer science, an overview of the main existing ontology matching and integration techniques is provided. A literature review of the main approaches in knowledge reuse is also provided, while the last section ends with an outlook of information visualization principles and tools in representing ontologies conceived as large graphs, being this an important part of the proposed framework for the ontology integration.

1.1 A brief historical background of the Semantic Web

Throughout the last decade, a more revolutionary web has emerged just when the main ideas and concepts behind the Web 2.0 were starting to enter into the main stream. The new web has augmented the previous one by promoting common data formats and exchange protocols in order to provide a common framework that allows data to be shared and reused across application, enterprise, and community boundaries [2]. Fostering a common knowledge model in order to formally represents knowledge or data to be reused and exchanged is the subject of a specific area of *Artificial Intelligence* (AI), viz. the *Knowledge Representation*. This concerns with how knowledge can be represented symbolically and manipulated in an automated way by reasoning programs [3]. The study of knowledge and its various models and implementations is such an interdisciplinary topic

integrating logic, philosophy, linguistics and computer science [4]. While the earliest works in computerized knowledge representation were focused on general problem solvers by Allen Newell and Herbert A. Simon in 1959, later on, in the seventies, with the advent of expert systems [5], a plethora of knowledge representation languages and techniques have emerged: semantic networks [6], frames-based languages [7], rules based languages [8], description logics languages [9] and, eventually, *ontologies*. The latter, in particular, and the tools developed to support them, have rapidly become *de facto* standards in the Semantic Web landscape and they are increasingly used, not only in research labs, but in large scale IT projects [10]. Thus, the term *ontology*, originally introduced by Aristotle, has become today a buzzword among the computer scientists, while ontologies are considered the *silver bullet* for the realization of the Semantic Web vision. According to Gruber [11], an ontology is a *an explicit representation of a conceptualization*, i.e., a formal definition and representation of the concepts and their relations belonging to a certain domain of interest. This definition has been the base for other variants like that proposed by Borst in [12], where an ontology is defined as *a formal specification of a shared conceptualization*, and by Studer in [13] who merges the previous definitions in a longer and probably better version: *An ontology is a formal, explicit specification of a shared conceptualisation. A "conceptualisation" refers to an abstract model of some phenomenon in the world by having identified the relevant concepts of that phenomenon. "Explicit" means that the type of concepts used, and the constraints on their use are explicitly defined. "Formal" refers to the fact that the ontology should be machine readable, which excludes natural language. "Shared" reflects the notion that an ontology captures consensual knowledge, that is, it is not private to some individual, but accepted by a group*. Finally, another definition worth to be quoted here, since it is close to the sense of ontology subtended in this work, is provided by Hendler in [14]: *A set of knowledge terms, including the vocabulary, the semantic interconnections and some simple rules of inference and logic for some particular topic*. This corresponds to a lightweight ontology that includes concepts, concept taxonomies, relationships between concepts, and properties which describe concepts (axioms and constraints are left aside in this case). Once a knowledge domain or some aspects of it are formally represented using a common and shared language, they become understandable not only by humans but also by automated computer agents [15]. As a result, for example, web services or search engines can improve their performances in terms of

exchange of information or accuracy in searching results, exploiting the semantically enriched representation of the information they share. For these reasons, ontologies are increasingly considered also as a key factor for enabling interoperability across heterogeneous systems [16], improve precision in retrieval process [17] and enhance efficiency and effectiveness of document management and representation [18, 19]. This revolution has led companies of all sizes and research groups to produce a plethora of data or conceptual models for many applications such as: e-commerce, government intelligence, medicine, manufacturing, etc. [20]. This, together with the increasing digitization of the world, has made available a huge amount of disparate information, raising the problem of managing heterogeneity among various information sources [21]. As more and more outbreaks of this revolution light up, a major challenge came soon into sight: addressing the main objectives of the Semantic Web, i.e., the sharing and reuse of data, demands effective and efficient methodologies to mediate between heterogeneous knowledge models. Specifically, one of the most challenge consists in integrating heterogeneous models in a unified (homogeneous) conceptualization of a specific knowledge or application domain, which allows the reuse of existing knowledge models. This is not an easy task to face due to ambiguities, inconsistencies and heterogeneities, at different levels, that could stand in the way. The ability to effectively and efficiently perform knowledge reuse is a crucial factor in knowledge management systems, and it also represents a potential solution to the problem of standardization of information and a viaticum towards the realization of the Semantic Web. In the context of ontology engineering, reuse of existing knowledge models is recommended as a key factor to develop cost effective and high quality ontologies, since it reduces the cost and the time required for creating *ex novo* domain conceptualisations, increasing the quality of newly implemented ontologies by reusing components that have already been validated [22, 23, 24]. It also avoids the confusion and the inconsistencies that may be generated from multiple representations of the same domain; thus, strengthening the orchestration and harmonization of knowledge [25]. Nowadays, ontology reuse is becoming increasingly challenging since available ontologies in the literature are becoming increasingly large in terms of number of concepts and relations, insofar that technical solutions belonging to the Big Data landscape can be adopted in order to make scalable ontology operations like storage, visualization and matching [26, 27, 28]. Ontologies are often the result of collaborative and distributed efforts that require effective methodologies to guarantee their mainte-

nance and evolution. In the attempt to mitigate the increasing heterogeneity and complexity of modern ontologies, several related research fields have emerged in the last years. *Ontology evolution* and *ontology versioning* aim at managing the inevitable changes to which ontologies are subject over time; whilst, *ontology matching*, *mapping*, *alignment*, and *ontology integration* and *merging* are the most spread research areas which aim to overcome the heterogeneity issue.

1.2 Ontology matching ground definitions: an introduction

Welcoming the suggestion for a clarification of the terminology contained in [29], some definitions about the key concepts used in this dissertation are provided in order to establish a solid background for the successive sections. According to some related works in the literature [30, 31], *ontology matching* is defined as the process of finding relationships or correspondences between entities of different ontologies; *ontology alignment*, as a set of correspondences between two or more ontologies; *ontology mapping*, as the oriented, or directed, version of an alignment, i.e., it maps the entities of one ontology to at most one entity of another ontology. More formally, the ontology mapping can be defined according to [32] as the task of relating the vocabulary of two ontologies that share the same domain of discourse in such a way that the ontological signatures and their intended interpretations, as specified by the ontological axioms, are respected. *Ontology integration* and *merging* are defined as the construction of a new ontology based on the information found in two or more source ontologies; and finally, *ontology reuse* as the process in which available ontologies are used as input to generate new ontologies. Less used but with a broader meaning is the term *ontology change* [29], which refers to any type of modification that it is needed over an ontology in response to particular needs. Its sense includes changes due to heterogeneity issues, ontology engineering updates, ontology maintenance, etc. In the following subsections, an outline of the main researches and solutions proposed for the different ontology integration disciplines (each in its own section) is provided.

1.2.1 Related works in ontology integration, matching and mapping

It is a common practice in the literature to consider heterogeneity resolution and related ontology matching or mapping strategies to be an internal part of ontology merging or integration [33]. Several works have been addressed in the last decade to ameliorate the ontology mapping and matching strategies for an effective and efficient data integration. According to Choi [16], ontology mapping can be classified into three categories: 1) mapping between an integrated global ontology and local ontologies, 2) mapping between local ontologies and 3) mapping on ontology merging and alignment. The first category of ontology mapping supports ontology integration by investigating the relationship between an integrated global ontology and local ontologies. The second category enables interoperability by providing a mediation layer to collate local ontologies distributed between different nodes. The third category is used as a part of ontology merging or alignment in an ontology reuse process. Some of the most spread tools belonging to this category will be describe as follows. SMART [34] is an algorithm that provides a semi-automatic approach to ontology merging and alignment assisting the ontology developer by performing certain tasks. It looks for linguistically similar class names through class-name matches, creates a list of initial linguistic similarity (synonym, shared substring, common suffix, and common prefix) based on class-name similarity, studies the structures of relation in merged concepts, and matches slot names and slot value types. SMART also determines possible inconsistencies in the state of the ontology that may result from the user's actions, and suggests ways to remedy these inconsistencies. Another semi-automatic ontology merging and alignment tool is PROMPT [35]. This performs some tasks automatically and guides the user in performing other tasks for which his intervention is required. It is based on an general knowledge model and therefore can be applied across various platforms. Anchor-PROMPT [36] takes a set of anchors (pairs of related terms) from the source ontologies and traverses the paths between the anchors in the source ontologies. It compares the terms along these paths to identify similar terms and generates a set of new pairs of semantically similar terms. OntoMorph [37] provides a rule language for specifying mappings, and facilitates ontology merging and the generation of knowledge-base translators. It combines two powerful mechanisms for knowledge-base transformations: syntactic rewriting and semantic rewriting. The first one is done through pattern-directed rewrite rules for sentence-level transformation based on pattern matching, while the latter is done through se-

mantic models and logical inference; FCA-Merge [38] is a method for ontology merging based on Ganter and Wille's formal concept analysis, lattice exploration, and instances of ontologies to be merged; and finally, CHIMAERA [39] that is an interactive merging tool based on Ontolingual ontology editor. It makes users affect merging process at any point during merge process, analyzes ontologies to be merged, and if linguistic matches are found, the merge is processed automatically, otherwise, further actions can be made by the user. It uses subclass and super class relationship. A survey of the matching systems is also provided by Shvaiko and Euzenat in [21], where, in addition to an analytical comparison of the recent tools and techniques, the authors argue on the opportunity to pursue further researches in ontology matching and propose a list of promising directions for the future. Noteworthy are some recent trends and future challenges suggested by the authors: large-scale knowledge bases matching and integration and ontology matching using knowledge background [40]. The first challenge is also subject of interesting studies conducted by Wiederhold [41], who has defined the services (or functions) a domain-specific mediator module must guarantee in order to collect and mediate information coming from increasing large-scale information systems. While dealing with large ontology integration, it is needed to apply *divide et conquer* strategies in order to improve the global performances of matching algorithms. In that regard, a common approach is dividing each ontology to several sub-ontologies, performing the matching operations and combining the results in a global view of the integrated ontology [42]. One of the tricky aspect of this approach is how ontologies are partitioned in order to avoid collating dissimilar sub-ontologies. In fact, if two ontologies are partitioned in n and m partitions respectively, it is needed to apply the matching operations to $n \times m$ sub-ontologies pairs. But, if a strategy is used to *filter out* all dissimilar sub-ontologies pairs, the effort for performing the whole matching task will decrease. Some of the partitioned methods existing in the literature are: modularization, decomposition, summarization, clustering, and blocking. Briefly, the modularization [43] allows to construct an ontology by putting together different component-ontologies conceived as actual *building block*, i.e., modules with a minimum set of axioms which maintain their entire entities and relations, and are linked together through importing instructions. Modularization is a crucial task to allow ontology reuse and exploitation on the Semantic Web. Blocking (or decomposition) [44] uses graph partitioning algorithms or other logic-based methods to *disintegrate* an ontology in several parts. Summariza-

tion [45] extracts a summary of the ontology, the most suitable for the matching operations to carry out. Finally, the clustering technique [46] consists in creating clusters of nodes (classes) with similar characteristics (equivalent classes, classes which share the same individuals, etc.), in order to carry out the matching operations cluster-wise rather than node-wise. In addition to these approaches, another relevant strategy, adopted in this work in order to reduce the complexity in terms of number of matching operations, is the *holistic schema and ontologies integration* described in [47]. Here different approaches for integrating multiple schema are described. One is *incremental binary integration strategy* that uses one ontology (or schema) as the initial integrated schema and incrementally match and merge the next source with the intermediate result until all resource schemas are integrated. The second strategy consists in reuse existing mappings between schema collected in specialized portal (e.g., Bio-Portal). For example, if it needs to match schema S_1 with schema S_2 and there exist the mappings (S_1, S_i) and (S_i, S_2), then the mapping (S_1, S_2) can be automatically obtained by combining the previous mappings. The third approach, the one used in this work, uses a *target* (or *hub*) ontology in order to integrate multiple input schema. The idea is to use *hub* concepts, coming from the hub schema and perform matching between the input ontologies and the hub ontology. This way the pairwise mappings can be automatically obtained. Some of the existing systems for large ontology integration are: AgreementMaker [48] which supports a wide variety of methods or matchers, and provides a GUI (Graphical User Interface) for showing the alignment between the source and the target ontology, and a control panel that allows users to run and manage matching methods and their results; LogMap [49], a scalable ontology matching system with *built-in* reasoning and diagnosis capabilities able to deal with ontologies containing tens (and even hundreds) of thousands of classes; GOMMA [50], which provides a scalable infrastructure to manage large life science ontologies and analyze their evolution. In fact, some of the key functions include a generic storage of ontology versions and mappings, support for ontology matching and determining ontology changes; Yam++ [51], a system able to discover mappings between entities of given two ontologies by using machine learning approach based on combination methods such as Decision Tree, SVM, NaiveBayes; COMA++ [52], which extends a previous project (COMA) by the same authors. It provides a graphical interface enabling a variety of user interactions; uses ontology matching strategies based on shared taxonomies and reuses previously determined match results and a *fragment-based*

approach to ontology matching which decomposes a large match problem into smaller problems.

The second challenge mentioned by Shvaiko and Euzenat in [21], the background-based matching, is also worth to looking into. Contrary to the direct matching that involves only the knowledge contained in the input ontologies entities, the new methodology performs the matching by discovering a common context or background knowledge for ontologies and uses it to extract relations between ontologies entities. Adding context can help to increase the recall but at the same time may also generate incorrect matches decreasing the precision, thus, a right tradeoff must be found. As background knowledge, on the one hand, it is common to use generic knowledge sources and tools, such as WordNet [53], Linked Open Data (LOD) like DBpedia [54], or the web itself; on the other hand, they can be used domain specific ontologies, upper level ontologies, or the ontologies available on the Web. The semantic matching framework S-Match [55], for example, uses WordNet as a linguistic oracle, while the work in [56] discusses the use of UMLS (Unified Medical Language System) [57], instead of WordNet, as a knowledge background in medical applications. Recently, many actions have been undertaken and numerous researches have been conducted in the field of Ontology Matching. It deserves a mention the Ontology Alignment Evaluation Initiative (OAEI)[1], which forges a consensus for evaluation of the increasing number of methods available for schema and ontology matching. The goals of OAEI are: assessing strengths and weaknesses of alignment-matching systems, comparing performance of techniques; improve evaluation techniques and helping improving the work on ontology alignment-matching, through the controlled experimental evaluation of the techniques performances. A yearly evaluation event is organized in order to publish tests and results of the event for further analysis. Many of the criteria provided in OAEI's white paper [58] are used in this work. The scope of the mentioned paper is presenting what kind of evaluation can be carried out on alignment algorithms, i.e., it presents an evaluation methodology composed of a benchmarking iteration that is continuously repeated and is composed of three phases: *Plan*, *Experiment*, and *Improve* and ends with a *Recalibration* task. In this process, a strategic relevance assumes the user. In fact, after many editions of OAEI, it is becoming clear to the community that there are limits to the performance (in terms of precision and recall

[1]Ontology Alignment Evaluation Initiative website. Available at http://oaei.ontologymatching.org/

of the alignments) of automated systems, as adopting more advanced alignment techniques has brought diminishing returns [59]. Thus, automatic generation of mappings should be viewed only as a first step towards a final alignment, with validation by one or more users being essential to ensure alignment quality [60]. As mentioned in the introduction, this principle has been fully accepted in this work, as all the approach leverage the expertise of the users (incorporating their knowledge) from the preliminary phases of the methodology to the evaluation ones.

1.2.2 Related works in knowledge and ontology reuse

As mentioned in the introductory section, ontology integration is mainly applied when the main concern is the *reuse* of ontologies. In this regard, it is noteworthy that several knowledge management methodologies consider the reuse of knowledge as an important phase of the entire knowledge management process. CommonKADS methodology [61], for instance, makes use of a collection of ready-made model elements (a kind of building blocks) which prevent the knowledge engineer to *reinventing the wheel* when modeling a knowledge domain. Moreover, the European research project NeOn [62] proposed a novel methodology for building ontology, which emphasizes the role of existing ontological and non-ontological resources for the knowledge reuse. Reuse is also a key requirement of OBO Foundry ontology [63], a collaborative effort to establish a set of principles for ontology development with the eventual goal of creating a set of interoperable reference ontologies in the domain of biomedicine [64]. The goal is to ensure that ontology developers reuse term definitions that others have already created rather than create their own definitions, thereby making the ontologies orthogonal, which means that each term is defined in only one ontology. Some recent works in the literature, mainly in the life sciences domain, still consider reuse as an important aspect of ontology construction or generation. OntoFox [65] is a web-based system that provides a timely publicly available service with different options for users to collect terms from external ontologies, making them available for reuse by import into client OWL ontologies. In [66] a semi-automatic ontology development methodology is proposed to ease the reusing phase in the development process, while [67] proposes a guiding framework for ontology reuse in the biomedical domain and [68] shows an approach to extract relevant ontology concepts and their relationships from a knowledge base of heterogeneous text documents. MIREOT (The Minimum Information to

Reference an External Ontology Term) [69] is a set of guidelines created to aid the development of the Ontology of Biomedical Investigations (OBI) based on Basic Formal Ontology (BFO) as upper-level ontology and is part of the Open Biomedical that proposes a set of guidelines for importing required terms from an external resource into a target ontology. In the context of Linked Open Data (LOD), [70] analyses 18589 terms appearing within 196 ontologies included in the Linked Open Vocabularies (LOV) registry with the aim of understanding the current state of ontology reuse in the LOD context, finding that the appearance of reused elements in the analyzed vocabularies is high (i.e., 40.53 per cent). Less recent works have investigated the ontology reuse from different points of view: [71] shows a new approach to reuse based on ontology modularization, [72] defines a three layered ontology design promoting maximal reuse of domain ontologies, [73] looks at the ontology reuse by exploiting the search engines have also started to appear, to facilitate search and retrieval of online ontologies and, finally, [74] presents CORE, a collaborative framework for Ontology Reuse and Evaluation. From a methodological point of view, Pinto and Martins [31] have analyzed the process of knowledge reuse by introducing an approach that comprises several phases and activities. In particular they identify three meanings of ontology integration: when building a new ontology by reusing (assembling, extending, specialising or adapting) other ontologies already available; when building an ontology by merging several ontologies into a single one that unifies all of them; when building an application using one or more ontologies [32]. However, some open issues remain, especially concerning the difficulty of dealing with the extreme formalisms heterogeneity of the increasing number of models available in the literature [22]. The absence of an automatic framework for the rigorous evaluation of the knowledge sources is also a severe limitation to overcome. The research introduced in this work tries to overcome the above difficulties, by adopting a framework for knowledge reuse based on the combination of existing ontology matching and integration methodologies, which exploits the available tools in order to automatize the repetitive tasks of ontology matching, but considering the human intervention strategic in some topical phases. It applies *divide et impera* strategies for dealing with large knowledge bases and applies the OAEI criteria for evaluating the whole approach.

1.2.3 Knowledge Visualization

The methodology proposed in this dissertation leverage the features and capabilities of some emerging tools to create, visualize and analyse NoSQL databases. Specifically, one of the methodology used in the matching strategy (the *extended linguistic matching*, see chapter 2) uses Neo4J [75] and Cytoscape [76] to construct a labelled property-based graph for implementing Semantic Networks (SNs). This model seems to fit well most of the common scenario from the real life and in particular from this work, since it conveniently allows to convert WordNet entities in a labelled graph whose nodes represent WordNet synset and words and edges represents semantic or linguistic relations. Since the study conducted in this work consists in the visual representation of WordNet excerpts in semantic networks implemented within Neo4j, this section provides a brief overview of the graph drawing algorithms existing in the literature.

Drawing algorithms

While *Graphs* are traditional and powerful tools that visually represent sets of data and the relations among them, *Graph visualization* usually refers to the representation of interconnected nodes arranged in space and navigation through a visual representation to help users understand the global or local original data structures [77]. Generally, graphs are represented by drawing a dot or circle for every vertex and an arc between two vertices if they are connected by an edge. If the graph is directed, the direction is indicated by drawing an arrow. A graph drawing should not be confused with the graph itself (the abstract, non-visual structure) as there are several ways to structure the graph drawing. All that matters is which vertices are connected to which others by how many edges and not the exact layout. In practice it is often difficult to decide if two drawings represent the same graph. Depending on the problem domain some layouts may be better suited and easier to understand than others. The pioneering work of W. T. Tutte [78] was very influential in the subject of graph drawing, in particular he introduced the use of linear algebraic methods to obtain graph drawings. The basic graph layout problem is very simple: given a set of nodes with a set of edges, it only needs to calculate the positions of the nodes and draw each edge as curve. Despite the simplicity of the problem, to make graphical layouts understandable and useful is very hard and there are generally accepted aesthetic rules [79, 80], which include: distribute nodes and edges evenly, avoid

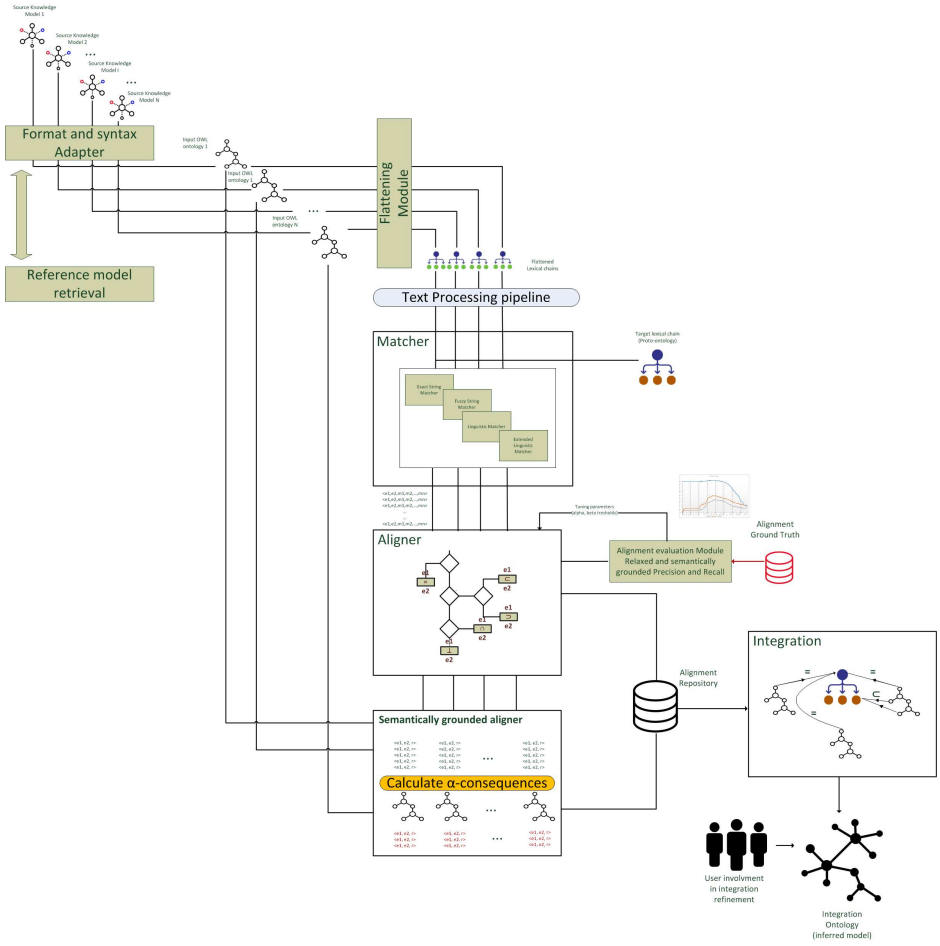
edge crossing, display isomorphic substructures in the same manner, minimize the bends along the edges. However, since it is quite impossible to meet all rules at the same time, some of them conflict with each other or they are very computationally expensive, practical graphical layouts are usually the results of compromise among the aesthetics. In this work, the Spring layout [81], also known as *Force-Directed* layout, will be used. By using it, graphs are modelled as physical systems of rings or springs. The attractive idea about spring layout is that the physical analogy can be very naturally extended to include additional aesthetic information by adjusting the forces between nodes. As one of the first few practical algorithms for drawing general graphs, spring layout is proposed by Eades in 1984 [82]. Since then, his method is revisited and improved in different ways [81, 83]. Mathematically, Spring layout is based on a cost (energy) function, which maps different layouts of the same graph to different non-negative numbers. Through approaching the minimum energy, the layout results reaches better and better aesthetically pleasing results. The main differences between different spring approaches are in the choice of energy functions and the methods for their minimization. Specifically regarding the visualization of WordNet, there are not many works in the literature. In [84], the authors makes an attempt to visualize the WordNet structure from the vantage point of a particular word in the database, this in order to overcome the down-side of the large coverage of WordNet, i.e., the difficulty to get a good overview of particular parts of the lexical database. An attempt to apply design paradigms to generate visualizations which maximize the usability and utility of WordNet is made in [85], whereas, in [86] a radial, space-filling layout of hyponymy (IS-A relation) is presented with interactive techniques of zoom, filter, and details-on-demand for the task of document visualization, exploiting the WordNet lexical database.

Chapter 2

A multi-strategy methodology for knowledge integration and reuse

This chapter describes the proposed multi-strategy methodology for ontology integration and reuse. The methodology is based on a software framework that contemplates different stages and includes different components. It starts with the harvesting of *source knowledge models* (also referred as *input models* or *reference models*) from the Internet or from specific repositories, evaluates them based on a set of qualitative criteria for the first selection and applies some linguistic and semantic similarity measures in order to individuate the models that best fit the *target model*. After that, the framework provides an alignment for the input and the target models and uses this to integrate the ones in the other. By doing so, it accomplishes the ultimate scope of this work, i.e., the reuse of existing knowledge models in the literature. The reminder of the chapter is structured as follows: the first section is an high-level outline of the framework with a description of the functionalities provided by each component (each in its own subsection), while the successive sections focus on the matching, aligning and integrating methodologies respectively. The chapter ends with a separate section describing how the target model has been created and the reasons why it has been introduced in the framework.

Figure 2.1: High-level view of the proposed framework



2.1 The proposed framework for ontology integration

As shown in Figure 2.1, the framework for knowledge integration and reuse proposed in this work (hereafter referred as *the framework*) presents several functional components (also referred as *blocks* or *modules*), the main ones being: the *Adapter*, the *Matcher*, the *Aligner* and the *Integrator*. In addition to these, there

is the *Reference Models Retrieval* component. Taken together, these blocks are able to obtain a comprehensive and integrated representation of the domain under study by an effective and efficient reuse of existing reference models in the literature. Stages (and blocks involved in each stage) are listed as follows:

- *sources knowledge models retrieval*, which consist in searching and retrieve from the Internet or from other repositories, the knowledge sources semantically close to the target model;
- *sources knowledge models adaptation and normalization*, which consists in adapting and homogenizing the retrieved source models at a syntactical and representational level. This operation is accomplished by the Adapter, the *Flattening Module* and the *Text-processing pipeline*;
- *input models matching*, which consists in calculating several lexicographic or linguistic similarity measures between the input and the target model. This stage involves the Matcher block;
- *input model aligning*, which consists in obtaining a mapping between the input and the target ontologies entities, i.e., an oriented, or directed, version of an alignment that maps the entities of one input model to at most one entity of the target models. This stage involves two sub-components: the Aligner and the *Semantically-grounded Aligner* (also referred as *Semantic Aligner*);
- *integrating the input models with the target*, which consists in applying an integration strategy between the input and the target models based on the alignment previously obtained, that means deciding how entities can be merged or linked in the global and integrated view of input and target models. The component involved in this stage is the Integrator.

The components listed above are not the only ones involved in the framework. In particular, as shown in figure 2.1, there is another component, namely the *Alignment Evaluation Module*, which is used to tune some parameters used by the Aligner classification algorithm in order to optimize its performance (in terms of *Precision* and *Recall*, see chapter 5). Figure 2.1 also shows an icon representing the users. Although this framework tries to automatize as much as possible all the stages of knowledge integration and reuse methodology, the user involvement is essential in some phases. As detailed in the next sections, user

involvement is needed in the first stages of the framework, when it is necessary to characterize the domain under study and to obtain the target model, and at the end (but also during the alignment process) in order to refine the integrated model obtained by the framework, validating alignments or detecting and correcting erroneous mappings. This enables system settings adjustments, selection of suitable alignment algorithms and the incorporation of user knowledge into the framework [87].

2.1.1 Reference Models Retrieval Module

The Reference Models Retrieval block is responsible for searching and retrieving the source models corresponding to the domain under study from the Internet or other specific repositories. In order to search for proper source models, it is necessary to identify the knowledge domain and the related sub-domains covering the specific topic under study. The contribution of users (with domain expertise) is essential in this phase in order to clarify the meaning of some poorly defined concepts and to help user with technical expertise to move among the existing *knowledge repositories* over the Internet or other legacy archives. Some of the available resources for domain identification are:

- Wordnet [53], a freely and publicly available large lexical database of English words;
- General purpose or content-specific encyclopaedia, e.g., Wikipedia and The Oxford Encyclopedia of Food and Drink in America [88];
- Web directories, e.g., DMOZ (from directory.mozilla.org) and Yahoo! Directory;
- Standard classifications, e.g., the International Classification for Standards (ICS) compiled by ISO (International Standardization Organization);
- Other electronic and hard-copy knowledge sources, including technical manuals, reports and any other documentation that the domain experts may consider useful to identify the knowledge domains.

Once the domain of interest has been properly defined, a first harvesting of source knowledge models can be done by accessing *knowledge repositories or providers* such as:

- Specialized portals and websites within public or private organizations;
- Search engines (e.g., Google, Bing, etc.), directory-based engines, e.g., Yahoo!, BOTW (Best of the Web Directory), DMOZ, etc., specialized semantic-based engines, e.g., Yummly (specialized on food), True Knowledge, etc.;
- Ontology repositories including: BioPortal [89], datahub.io[1], Knoodl repository[2], etc., and search engines for semantic web ontologies, e.g., Swoogle[3] and the Watson Semantic search engine[4];
- Available standards and non-standard reference models that provide requirements, specifications, guidelines and characteristics of a service or a product (ISO standards, the IFC Industry Foundation Classes [90], Ansi/ISA-95 [91], STEP [92], and Core Product Model [93]).

As soon as the source models are harvested from the repositories and collected into the *corpus*, it becomes indispensable to perform a first screening over them in order to select the most suitable for the purpose. The selection is accomplished by using some qualitative criteria reported in Table 2.1 and detailed as follows: the *Language formality* level (C1), which describes the formality of the conceptual model representation that can range from plain text with no formalism used to formal languages like the first-order logic-based languages; the *Domain specificity* (C2), which evaluates the model type from the viewpoint of its generality (upper-level model or application-specific model); the *Model structuring* (C3), which evaluates the model type from the viewpoint of its structure (simple classifications or taxonomies versus representation language based model like UML [94] and EXPRESS [92]); the *Model language* (C4), which describes the language used to represent the conceptual model, including RDF (Resource Description Framework) or OWL (Ontology Web Language), graphic-based languages and plain text; the *Model provenance* (C5), which evaluates the model from the viewpoint of its origin, thus giving higher rates to standards or conceptual models authored by influential scientific groups. Finally, the *Model availability* (C6), which evaluates the availability of the conceptual model (open

[1]Datahub dataset website, https://datahub.io/
[2]Knoodl webpage, http://www.knoodl.com/ui/home.html
[3]http://swoogle.umbc.edu/
[4]http://watson.kmi.open.ac.uk/WatsonWUI/

data-models versus proprietary and licensed models). A higher rate will be given to formal models because the aim of the framework is to reuse existing models for a formal conceptualization of the target model.

Table 2.1: Initial selection criteria

No.	Criterion	Range (examples)
C1	Language formality	No formalism used, a semi-formal language, formal language
C2	Domain specificity	Upper-level or abstract domain model, application specific domain
C3	Model structuring	Taxonomies, representation language based models (UML, EXPRESS), plain text
C4	Model language	XML-based language, Logics-based languages (OWL-DL ontologies), plain text
C5	Model provenance	Private studies (technical reports), model authored by influential scientific groups, de facto standard, standard.
C6	Model availability	Open-data models, proprietary and licensed

2.1.2 Reference Model Adaptation and Normalization Module

The second component of the framework is the Adaptation and Normalization block (Figure 2.2). It is responsible for adapting and homogenizing the retrieved source models at a syntactical and representational level. This step is mandatory due to the heterogeneity of languages used to represent and formalize knowledge models in the literature. In fact, these can be represented using plain text, semi-structured texts (like XML, HTML, EXPRESS, etc.), graphical-based languages (like UML, (E-R) Entity-Relationship, etc.), ontology languages with different syntax (e.g., SKOS: Simple Knowledge Organization System) or levels of expressiveness (RDF, OWL-Lite, OWL-DL, OWL-Full). The idea behind this component is to use an adapter for each reference model retrieved from the Internet that understand the nature of the model, reads it in a suitable way and transforms it in a *lite* ontology, which preserves the *is-a* hierarchy between the classes and all available linguistic annotations (labels and comments). In the understanding of the presented work a *lite* ontology consists of concepts and

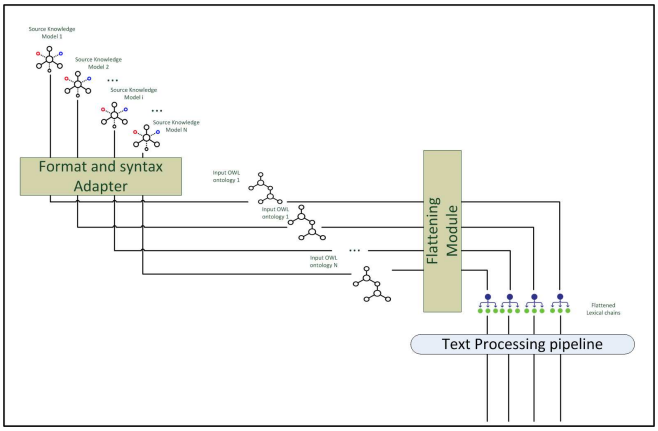


Figure 2.2: Reference Model Adaptation and Normalization component

subsumption relations between concepts without considering instances or individuals, and can be formally defined as follows:

$$O := (C, H_C, M_C, A) \tag{2.1}$$

An ontology O consists of the concepts C of the schema, which are arranged in a subsumption hierarchy H_C. All the annotation attached to the concepts in C (labels and comments) are included in M_C. Additionally, A represents the axioms which can be used to infer knowledge from already existing one, as explained soon. Technically, each adapter is a kind of *plugin* added to the software framework. As detailed further in chapter 3, the adapter is a Java object that encapsulates a memory-based OWL-Lite model using the Jena APIs [95]. The library supports a transitive reasoner so that it is possible to automatically add the *transitive closure* to one input model. In other words, if A is $subclassOf$ B and B is $subclassOf$ C, the axiom A $subclassOf$ C is automatically inferred and added to the model by the reasoner. Once the source models have been converted into OWL ontologies, they become *adapted input ontologies* and are ready to be read by the successive components of the framework. At this stage, they go through two different paths: the first one crosses the flattening module and the text-processing pipeline, described later, and enters the Matcher and the Aligner blocks for the linguistic matching; the second one goes directly to the Semantic Aligner for the semantic matching operations. The Flattening module produces

a list of linguistic labels, i.e., a *lexical chain* (hereafter referred as *input lexical chain*) gathered from the English labels attached to each class in the adapted input ontologies. Noteworthy, this version of the framework applies the matching and integration methodologies only at a concept-class level, so ontological individuals eventually belonging to the input ontologies are discarded and do not contribute to enrich the lexical chain. All input lexical chains go through the linguistic Matcher and the Aligner after being processed into the text-processing pipeline.

Text processing pipeline

The normalization of textual representation of entities (labels or comments) at a morphological and syntactic level is performed by the text-processing pipeline. Figure 2.3 shows the main phases of the pipeline:

- *Sentence Segmentation* is responsible for breaking up documents (entity description, comments or abstract) into sentences (or sentence-like) objects which can be processed and annotated by "downstream" components;

- *Tokenization* breaks sentences into sets of word-like objects which represent the smallest unit of linguistic meaning considered by a natural language processing system;

- *Lemmatisation* is the algorithmic process of determining the lemma for a given word. This phase substantially groups together the different inflected forms of a word so they can be analysed as a single item;

- *Stopwords elimination* phase filters out stop words from analysed text. Stop words usually refer to the most common words in a language, e.g. *the*, *is*, *at*, *which*, and so forth in English;

- *POS (Part-Of-Speech)-tagging* attaches a tag denoting the part-of-speech to each word in a sentence, e.g., *Noun*, *Verb*, *Adverb*, etc.;

- *Named Entity Recognition* phase categorizes phrases (referred to as entities) found in text with respect to a potentially large number of semantic categories, such as person, organization, or geopolitical location;

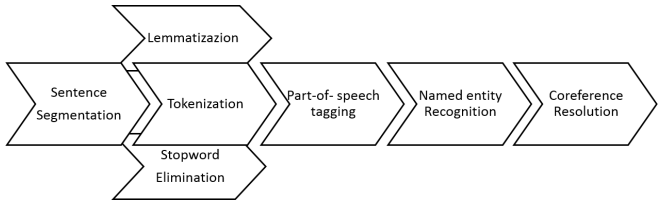


Figure 2.3: Text processing pipeline in the normalization phase

- *Coreference Resolution* phase identifies the linguistic expressions which make reference to the same entity or individual within a single document – or across a collection of documents.

The linguistic normalization also involves resolving multi-languages mismatching automatically or manually by translating metadata description from whatever languages into English.

2.1.3 Reference Model Matcher

One of the main components of the framework is the *Matcher* (Figure 2.4), which is responsible for obtaining a set of similarity measures taking as inputs the terms coming from the source models lexical chains and the terms from the target lexical chain. As described further in chapter 3 and 4, the output of the Matcher is a vector as represented in table 2.2 for each pair of terms.

src	dst	str	lev	jac	fuz	syn	cos	wup	path	extAvgWup	extMinWup

Table 2.2: Vector of similarity measures between two compared terms

The vector is composed of three parts: the terms to be matched (*src* and *dst*), the block of lexicographic measures and the block of linguistic measures. This latter use an external linguistic resource, viz. WordNet, to obtain similarity measures which generally are based on the concept of path distance between two nodes in a hierarchical tree or in a graph.

Here a punctual description of each vector element follows:

- *src*, is the source string, one of the terms in the input lexical chain. It may be a single word (e.g., *food* or *meat*) or a multi-word (e.g., *solid food* or *orange juice*);

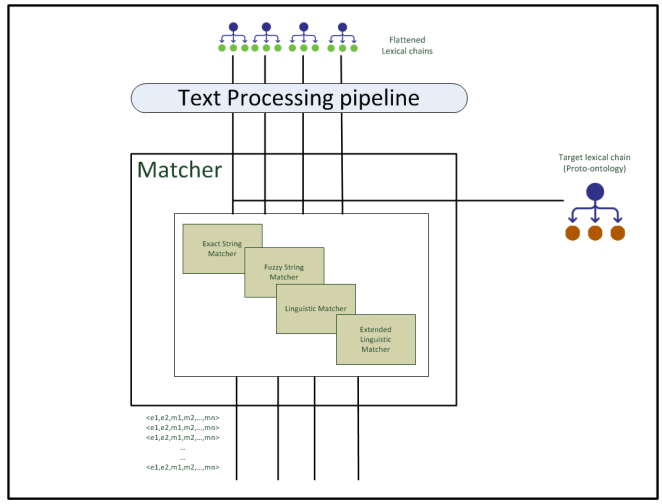


Figure 2.4: The matcher component

- *dst*, is the destination string coming from the target lexical chain. Like the source, it may be a single or a multi-word;

- *str*, is the value returned by a perfect matching function. It is an on-off value, which can be equal to 1 in case of exact matching or 0, otherwise;

- *lev*, is the *Levenshtein* [96] measure applied to *src* and *dst* as input strings. It is a decimal value ranging from 0.0 to 1.0;

- *jac*, is the Jaccard similarity applied to *src* and *dst* if one of them (or both) is (are) multi-word term(s);

- *fuz*, is a *fuzzy* partial string similarity measure between *src* and *dst*. It acts according the rules described in the matching methodology section;

- *syn*, is the *synonymy* grade between *src* and *dst*. It is a decimal values that is equal to 1 if there is at least one sense of *src* that is also a sense for *dst*. This is a linguistic measure that exploit an external linguistic resource like WordNet. Section 2.2.3 describes WordNet and all linguistic measures in detail;

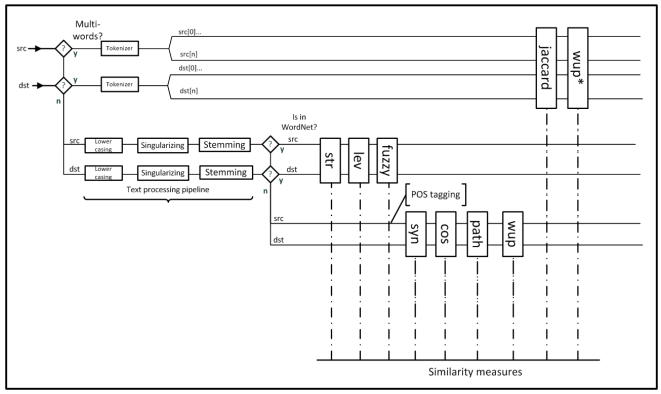


Figure 2.5: The matching strategy

- *cos*, is the *co-synonymy* grade between *src* and *dst*. It is a decimal values that is equal to 1.0 if all senses of *src* are just the senses for *dst*;

- *wup* is the average of the *Wu & Palmer* [97] similarity measures between all synsets of *src* and all synsets of *dst*. Its value is between 0.0 for low similarity and 1.0 for high similarity;

- *path*, is the average of *path distance* between all synsets of *src* and all synsets of *dst*;

- *extAvgWup* is the *Extended Averaged Wu & Palmer similarity* that applies when *src* and *dst* are multi-word terms. It is the average of wup similarities for all multi-words' tokens;

- *extMinWup* is the *Extended Minimized Wu & Palmer similarity* that applies when *src* and *dst* are multi-word terms. It is the minimum of wup similarities for all multi-words' tokens.

All the measures listed above are calculated according the scheme depicted in Figure 2.5, which shows the text-preprocessing operations performed over the inputs strings before applying each function (contained in the blocks).

2.1.4 Reference Model Aligner

The Aligner component (Figure 2.6) is responsible for obtaining a mapping, i.e., an oriented or directed version of the alignment that maps the entities of one input model to at most one entity of the target models (at this stage only lexicographic and linguistic based). The Aligner acts as a classifier, i.e., it takes two strings representing single words or multi-words as input and use the similarity measures provided by the matcher to predict the classification class where putting the pair according to a decision tree. The classification strategy and the decision tree will be described in detail in the aligning methodology section. Here follows a brief description of each classification class and its related meaning:

- *Equivalent* (in symbols: $\equiv$), the terms-pair is put in this class if they are supposed to be equivalent, i.e., they are supposed to have the same meaning;
- *Hypernym* (in symbols: $\sqsupset$), the terms-pair is put in this class if the first term is supposed to be a broader concept w.r.t. the second one, i.e., it is an hypernym, or a superclass that subsumes the second;
- *Hyponym* (in symbols: $\sqsubset$), the terms-pair is put in this class if the first term is supposed to be a narrower concept w.r.t. the second one, i.e., it is an hyponym, or a subclass of the second;
- *Related* (in symbols: $\sqcap$), the terms-pair is put in this class if the first term is supposed to be semantically related to the second one, or, thinking them set-theoretically, they have an intersection not null;
- *Disjointed* (in symbols: $\perp$), the terms-pair is put in this class if the first term is not related at all to the second one, or, thinking them set-theoretically, they have a null intersection;
- *Unknown*, this is a virtual class. The aligner classifier put the terms-pair here when it is not able to classify them.

Semantically-grounded Aligner

The Semantically-grounded Aligner, represented as a separate module in figure 2.6, adds all α-consequences to the alignment provided by the Aligner. The

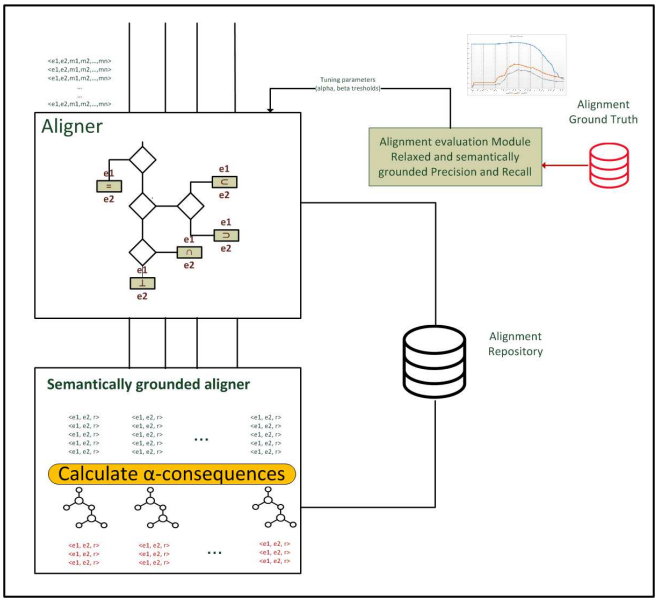


Figure 2.6: The aligner component

notion of α-*consequences* will be detailed in the aligning methodology section and is equivalent to what defined in [1]. It is important to note here that, while in the previous blocks only a flattened version of input ontologies have been matched together, now the structure of ontologies is recovered in order to exploit the inherent semantics in the *class-subclassOf* hierarchy. This way it is possible to generate new maps between terms (semantically-grounded) that otherwise will be missed.

2.1.5 Reference Models Integrator

The output of the Integrator module (Figure 2.7) is a global, richer and more integrated ontology (hereafter referred as the *integration ontology* or the *output ontology*), abstracting the local conceptualizations of the input ontologies and, as much as possible, complete, homogeneous, correct and coherent. The output ontology must not contain misleading or duplicate concepts, highly abstract or specific concepts w.r.t. the target ontology concepts, and must have a well-balanced hierarchy. In order to achieve this, it is necessary to further select the

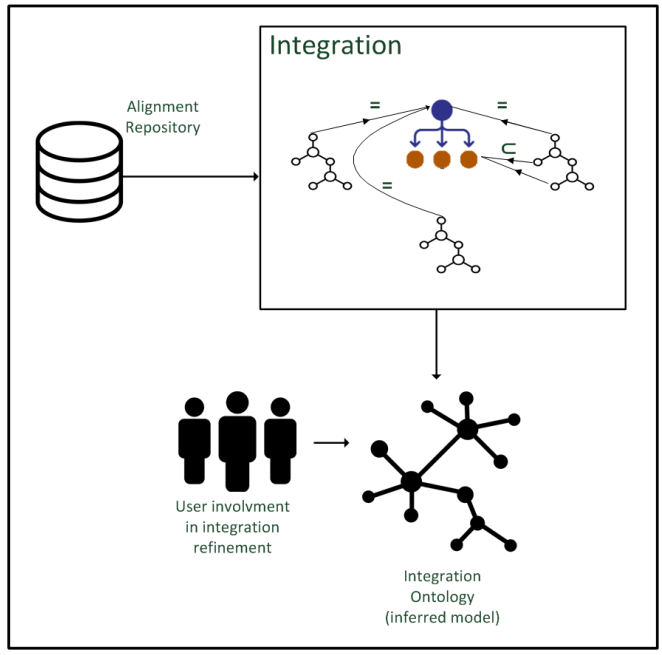


Figure 2.7: The integrator component

input ontologies using a ranking function that assign a vote to each ontology. Only those whose vote is higher than a certain value (hereafter referred as *selected ontologies*) enter the integrating phase. Different solutions can be adopted for creating a ranking function between the input ontologies, for example, the measures obtained from the matcher block can be aggregated for all ontologies in order to obtain a global rank value. In this work, a system grading that is able to assign a vote to each ontology based on their syntactic and semantic content is used and will be described in section 2.2.3.

The details of the integrating strategy will be described in the successive sections, while here a not exhaustive list (the most important ones) of the rules that the integrator must respect for obtaining an high quality output ontology is provided:

- *Rule 1: importing all selected ontologies as separate ontology modules.* This means constructing a modularized ontology by using *namespaces* with associated *uris* so that it is clear where the imported ontologies enti-

ties come from;

- *Rule 2: preserving linguistic annotations in the output ontology*. The Integrator must preserve all annotation available from the selected input ontologies (labels and comments) in order to enrich the entities merged or integrated in the output ontology. For example, if two equivalent classes have got a comment, one comment can be concatenated with the other and used as more complete description of the merged concept;

- *Rule 3: using human-readable labels for class names*. If available, each class must be annotated with a readable label coming from the source ontology. If not, a proper label will be created from the local class name and attached to the class;

- *Rule 4: avoiding unnecessary axioms*. This means keeping the integrated ontology as clear as possible and, at the same time, as concise as possible. Thus, all unnecessary axioms will be neglected. For example, if two classes are predicted as *Disjointed* by the Aligner, it is preferable to avoid to explicitly declare them as *disjointWith* in the output ontology;

- *Rule 5: using reasoning capabilities*. This means, for example, using transitive reasoner for automatically inferring a $subClassOf$ property rather than explicitly asserting it;

- *Rule 6: keeping ontology's hierarchy well-balanced*. This means adopting proper strategies, so that each path descends to the same depth. It would require introducing middle-level classes between narrower and broader concepts.

2.2 The matching methodology

As described in section 2.1.3, the Matcher module provides a vector of similarity measures for each pair of terms coming from the input and the target lexical chains. More formally, a measurement between two terms is a tuple in the form:

$$c = (t_1, t_2, m_1, v_1, m_2, v_2, ..., m_i, v_i, ..., m_n, v_n)$$

where t_1 is the source term, which come from the first ontology, t_2 is the destination term, from the second ontology, m_i is a lexicographic or linguistic

measure and v_i its value. All tuples are collected in a CSV (Comma Separated Values) file as detailed in the next chapter. All similarity measures can have a value in the interval [0, 1]. The matching methodology involves three types of matching: the *string-based matching*, the *linguistic matching* and the *extended linguistic matching* (used for the ranking function already described in section 2.1.5). In this version of the framework, only the previous methodologies are used, because they apply to the flattened *view* of the ontologies. Thus, actual ontology matching techniques, which operate at a structural level even handling with individuals, property axioms, restrictions and so forth, are not contemplated in this dissertation. In the following sub-sections each of the matching methodologies will be further detailed.

2.2.1 The string-based matching

The string-based matching operation is performed between the labels attached to the ontology entities (mainly classes, and also property labels, if meaningful) coming from the input and the target ontologies. Additionally, metadata like comments, abstracts or descriptions can be also taken into consideration, in this phase, if necessary.

Three different string matching techniques are used here: the *exact* string matching, the *partial* string matching (using the Levenshtein or Edit distance) and the *fuzzy* string matching.

The exact string matching

The exact string matching measure is an on-off value. It gives 1 if there is a perfect matching between two strings (or terms), 0 otherwise. Examples are very trivial but it is noteworthy that the algorithm performing the exact string is case-sensitive thus is necessary a lower(upper)-casing conversion before using it.

The partial string matching

The similarity measure used for partial string matching in this work is the Edit Distance (or Levenshstein similarity). This is a way of quantifying how dissimilar two strings (e.g., words) are to one another by counting the minimum number of operations required to transform one string into the other. Given two strings a

and b on an alphabet (e.g., the set of ASCII characters, the set of bytes [0..255], etc.), the Edit Distance $d(a, b)$ is the minimum-weight series of edit operations that transforms a into b. One of the simplest sets of edit operations is that defined by Levenshtein in [98]: insertion of a single symbol, deletion of a single symbol, substitution of a single symbol x for a symbol y. The Levenshtein distance between "kitten" and "sitting", for example, is 3. A minimal edit script that transforms the former into the latter is:

```
kitten vs sitten (substitution of "s" for "k")
sitten vs sittin (substitution of "i" for "e")
sittin vs sitting (insertion of "g" at the end).
```

The fuzzy string matching

Concerning the string relatedness measures it is note worthy that the standard measurement, like the edit or Levenshtein distance works fine for very short strings (such as a single word) and very long strings (such as a full book), but not so much for 3-10 word labels. The naive approach is far too sensitive to minor differences in word order, missing or extra words, and other such issues, so it is necessary to use *fuzzy* approaches and heuristics in order to relax the standard measures and avoid the risk of getting bad matchings. The fuzzy approach to string matching, implemented in a suitable Python library available on line [5] and used in this work, includes the following:

- *Partial String Similarity*, which uses the "best partial" heuristic when two strings are of noticeably different lengths. If the shorter string is length m, and the longer string is length n, this similarity basically scores the best matching length-m sub-string. So, for example, the string "Yankees" and "New York Yankees" are a perfect partial match;
- *Out of Order* is another issue encountered with string matching. In this case two strings are similar if they differ only on the order of the terms within them. Two different approaches can be used here:
  - *The token sort* approach involves tokenizing the string in question, sorting the tokens alphabetically, and then joining them back into a string;

[5] Fuzzy String Matching in Python. Available online, http://chairnerd.seatgeek.com/fuzzywuzzy-fuzzy-string-matching-in-python/

– *The token set* approach is similar, but a little bit more flexible. Here, both strings are tokenized, but instead of immediately sorting and comparing, the tokens will be split into two groups: intersection and remainder. These sets will be used to build up a comparison string.

Specifically, the fuzzy string similarity methods correct the standard measures by reducing their sensitiveness to minor differences in word order, missing or extra words, and other such issues. In this work, the Levenshtein distance (or Edit Distance) is used to quantify how dissimilar two single-word labels are dissimilar to one another, and the *fuzzy* string matching methods to compare multi-word labels or abstract and comments attached to the ontology' entities.

Jaccard similarity for multi-words

The Jaccard similarity measure will be applied when at least one of the terms being matched is a multi-words.The Jaccard similarity is defined as follows:

$$jaccard(A, B) = \frac{|A \cap B|}{|A \cup B|} \tag{2.2}$$

where A and B are the set of tokens in which the matched words can be split. Thus, the Jaccard measure is the ration between the number of the shared and the total number of tokens. For example, starting from the words *orange juice* or *lemon juice*, the Jaccard measure is $\frac{1}{3}$, while the Jaccard measure for *fresh food* and *food* is $\frac{1}{2}$.

2.2.2 The linguistic matching

The linguistic matching is responsible for a comprehensive analysis of the terms used in the input models at a semantic level, using an external linguistic database like WordNet or other domain specific knowledge sources as background knowledge. In WordNet, nouns, verbs, adjectives and adverbs are grouped into sets of cognitive synonyms (synsets), each expressing a different concept [53] (Figure 2.8). The synsets are interlinked by conceptual-semantic and lexical relations, thus realizing a graph-based structure where synsets are nodes and lexical-relations are edges. Exploiting the WordNet graph-based representation, it is possible to relate concepts at a semantic level, for example, by calculating the Wu-Palmer similarity or the *path* distance between two synsets [97], which

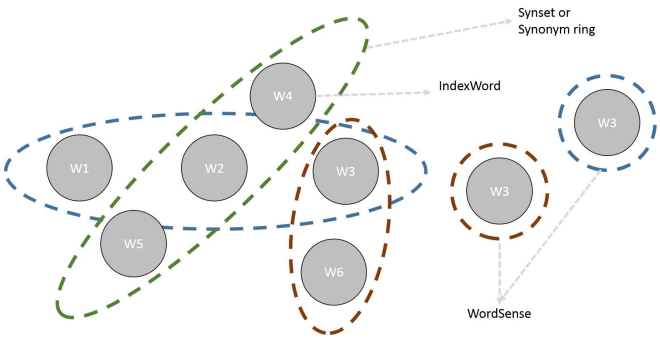
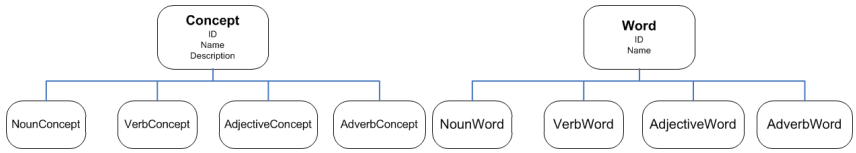


Figure 2.8: WordNet words, synsets and word senses

counts the number of edges between two concepts by taking into account their proximity to the root concept of the hierarchy. According to [99], Wu-Palmer similarity has the advantage of being simple to calculate, in addition to its performances while remaining as expressive as the others.

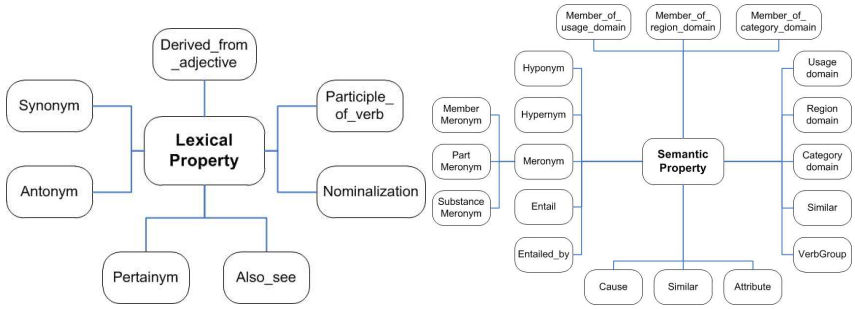
2.2.3 Extended linguistic matching

The extended linguistic matcher component defines and implements a meta-model for ontology matching using a conceptualization as much as possible close to the way in which the concepts are organized and expressed in human language [100]. The extended matcher exploits this meta-model for improving the accuracy in selecting candidate reference models. The meta-model is defined as a triple $\langle S; P; C \rangle$ where: S is a set of objects; P is the set of properties used to link the objects in S; C is a set of constraints on P. In this context, concepts and words as considered as objects, properties are linguistic relations and constraints are validity rules applied to linguistic properties w.r.t. the considered term category (i.e., noun, verb, adjective, adverb). In this approach, the target knowledge is represented by the target ontology; a concept is a set of words which represent an abstract idea; every node, both concept and word, is a labelled-graph node and, finally, the connecting edges represent relations (also referred as properties) between nodes and are implemented as *labelled arcs*. These relations have some constraints that depend on the syntactic category or on the kind of properties (semantic or lexical). For example, the hyponymy relation can relate only nouns to nouns or verbs to verbs; on the other hand a semantic relation links con-



(a) concept (b) word

Figure 2.9: Concept and Word



(a) lexical (b) semantic

Figure 2.10: Lexical and semantic relations

cepts to concepts and a syntactic one relates word forms to word forms. Concept and word attributes are implemented as *node properties*, which relate individuals with a predefined data type. Each word is related to the represented concept by the meta-relation *hasConcept* while a concept is related to words that represent it using the meta-relation *hasWord*. These are the only properties able to relate words with concepts and vice versa; all the other properties relate words to words and concepts to concepts. Concepts, words and properties are arranged in a hierarchy, resulting from the syntactic category for concepts and words and from the semantic or lexical type for the properties.

Figures 2.9a and 2.9b show that the two main classes are: Concept, in which all the objects have defined as individuals and Word which represents all the terms in the ontology.

The subclasses have been derived from the related categories. There are some union classes useful to define properties domain and co-domain; moreover, some attributes for Concept and Word are defined as follows: *hasName* that represents

the concept name; *Description* that gives a short description of concept. On the other hand, Word has *Name* as attribute (i.e., the Word name) and all elements (concepts and words) have an ID coming from the WordNet offset number or defined by the user. The semantic and lexical properties are arranged in a hierarchy (see Figure 2.10a and 2.10b). In table 2.3 some of the considered properties and their domain and range of definition are shown.

Table 2.3: Properties

Property	**Domain**	**Range**
hasWord	Concept	Word
hasConcept	Word	Concept
hypernym	NounsAnd	NounsAnd
	VerbsConcept	VerbsConcept
holonym	NounConcept	NounConcept
entailment	VerbWord	VerbWord
similar	AdjectiveConcept	AdjectiveConcept

The use of domain and co-domain reduces the property range application. For example, the hyponymy property is defined on the sets of nouns and verbs; if it is applied on the set of nouns, it has the set of nouns as range, otherwise, if it is applied to the set of verbs, it has the set of verbs as range. In table 2.4, there are some of the defined constraints along with the specification of which classes they have been applied to, by taking into account the considered properties; the table shows the matching range too.

Table 2.4: Model constraints

Costraint	**Class**	**Property**	**Constraint range**
AllValuesFrom	NounConcept	hyponym	NounConcept
AllValuesFrom	AdjectiveConcept	attribute	NounConcept
AllValuesFrom	NounWord	synonym	NounWord
AllValuesFrom	AdverbWord	synonym	AdverbWord
AllValuesFrom	VerbWord	also_see	VerbWord

Sometimes the existence of a property between two or more individuals entails the existence of other properties. For example, being the concept *dog* a hyponym of *animal*, it can be asserted that *animal* is a *hypernymy* of *dog*. This

can be represented in OWL by means of property features shown in table 2.5.

Table 2.5: Property features

Property	**Features**
hasWord	*inverse* of hasConcept
hasConcept	*inverse* of hasWord
hyponym	*inverse* of hypernym; *transitivity*
hypernym	*inverse* of hyponym; *transitivity*
cause	*transitivity*
verbGroup	*symmetry* and *transitivity*

Having defined the meta-model previously described, a Semantic Network (i.e., SN) is dynamically built using a dictionary based on WordNet or other domain specific resources. Hereafter, the Semantic Network is defined as a graph consisting of nodes, which represent concepts, and edges, which represent semantic relations between concepts. The role of domain experts is precious in this phase because they interact with the system by providing a list of domain keywords and concept definition feeding the *target-ontology* (see section 2.5).

The SN is built starting from such first version of the target ontology, i.e, the domain keywords and the concept definition words sets. Afterwards, a hierarchy of synsets based on the hyponymy relation is constructed. The last level of this hierarchy corresponds to the last level of WordNet's one. After this first step,the hierarchy has been enriched considering all the other kinds of relationships in WordNet (e.g., meronymy). In this approach, as soon as the SN is built, it is compared with the selected input models lexical chains. The intersection between SN and the input models leads to a lexical chain with the relevant terms related to the target ontology. All terms are linked by properties from the SN. Therefore, the SN leads to a conceptual frame useful to discriminate the pertinent reference models from the other ones. In order to evaluate the relevancy of the selected reference model, it is necessary to define a system grading that is able to assign a vote to the model based on their syntactic and semantic content. For this reason, the approach described in [100] is adopted to calculate a *Global Grade* (GG) for each semantic network related to each selected reference model. The GG is given by the sum of the *Syntactic-Semantic Grade* (SSG) and the *Semantic Grade* (SG). The first contribution returns information about the analyzed model by taking into account the polysemy of the term, i.e., the measure of ambiguity in

the use of a word, thus, with an accurate definition of the role of the considered term in the model. This measure is referred as the *centrality* of the term i and it is defined as: $\varpi(i) = \frac{1}{poly(i)}$ where *poly(i)* is the polysemy (number of senses) of i.

Additionally, the relevance of the reference model can be defined as the sum of its relevant word *weights* (terms centralities):

$$SSG(\nu) = \sum_{i=1}^{n} \varpi(i) \tag{2.3}$$

where *n* is the number of terms in the model ν.

The other contribution (SG) is based on a combination of the path length (l) between pairs of terms and the depth (d) of their *subsumer* (i.e., the first common ancestor), expressed as number of hops. Moreover, to each linguistic property, represented by arcs between the nodes of the SN, a weight is assigned in order to express the strength of each relation. In fact, not all the properties have the same strength when they link concepts or words (this difference is related to the nature of the considered linguistic property). The weights are real numbers in the [0, 1] interval and their values are set by experiments and they are validated, from a strength comparison point of view, by experts.

Taking into account the above and by extending the metric proposed in [101], the *Semantic Grade* (SG) measure can be defined as follow:

$$SG(\nu) = \sum_{(w_i,w_j)} e^{-\alpha \cdot l(w_i,w_j)} \frac{e^{\beta \cdot d(w_i,w_j)} - e^{-\beta \cdot d(w_i,w_j)}}{e^{\beta \cdot d(w_i,w_j)} + e^{-\beta \cdot d(w_i,w_j)}} \tag{2.4}$$

where (w_i, w_j) are pairs of word in the intersection between ν input model's SN and the target one, while $\alpha \geq 0$ and $\beta > 0$ are two scaling parameters whose values have been defined by experiments.

The final grade is the sum of the Syntactic-Semantic Grade and the Semantic Grade.

Once obtained the Global Grade for each semantic network, they are compared with a threshold value that act as a filter for the input reference models, thus returning the most relevant input models at a linguistic level.

2.3 The aligning methodology

As already stated in the previous sections, the Aligner outputs a mapping, i.e., an oriented or directed alignment between the entities of one input model to at most one entity of the target models. The Aligner acts as a classifier using the similarity measures produced by the Matcher and the classifier decision tree pictured in figure 2.11. The classifier uses two thresholds (th_1 and th_2) whose values have been calculated by experiments (see chapter 5). Formally, it is a mapping function between the measurement tuples as defined in section 2.1.3 and the classification classes (defined in section 2.1.4):

$$f(t_1, t_2, m_1, v_1, m_2, v_2, ..., m_i, v_i, ..., m_n, v_n) \rightarrow (t_1, t_2, <>) \tag{2.5}$$

where $<>$ is one of ($\equiv$, $\sqsupset$, $\sqsubset$, $\sqcap$ and $\perp$).

In this version, the Aligner classification algorithm does not use all the measures provided by the Matcher. In fact, the only measures involved in the decision tree are: the exact string measure (str), the co-synonym (cos), the Wu & Palmer similarity (wup) and the multi-word version of the Wu & Palmer similarity (both the averaged and the minimized, wup^*). Moreover, it uses other information about the terms being aligned: the *depth* of each term in the WordNet hierarchy, i.e., the maximum distance of its synsets, in terms of hops, from the root concept of WordNet, and a boolean function *substr()* that returns *true* if the first terms (src) is contained (i.e., is a *sub-string of*) the other. This additional parameters allows the classifier to use the following heuristics in order to align the terms against the hypernym-hyponym relation: if the first term contains the second one, it is likely to be a narrower concept w.r.t. the second (e.g., *apple juice* and *juice*); the same occurs if the first term has a depth grater then the second in the WordNet hierarchy (e.g., *juice*, whose related synsets have an average depth of 8.75 and *beverage*, whose related synsets have an average depth equal to 7.0). Note that the ability of disambiguating from hyponym and hypernym classes allows the Aligner to obtain a mapping (a direction in the alignment), and thus, to make possible for the Integrator module to automatically construct a class hierarchy in the integration ontology.

Selecting only the needed measures and adding the additional information mentioned above, the classification decision tree pictured in figure 2.11 can be formalized as follows:

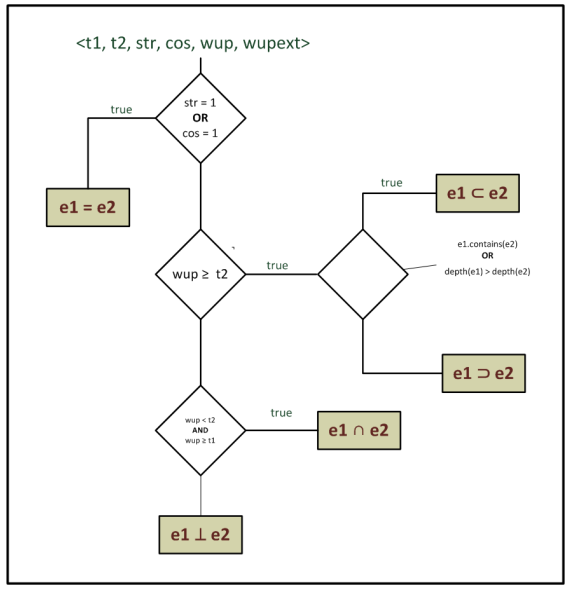


Figure 2.11: The Aligner classifier decision tree

$$f(t_1, t_2, str, cos, wup, wup^*) = \begin{cases} (t_1, t_2, \equiv) & cos = 1 \\ \begin{cases} (t_1, t_2, \sqsubset) & (*) \\ (th_1, t_2, \sqsupset) & (**) \end{cases} & wup \geq th_2 \\ (t_1, t_2, \sqcap) & th_1 \geq wup < th_2 \\ (t_1, t_2, \perp) & wup < th_1 \end{cases} \tag{2.6}$$

where:
$(*) = (substr(t_1, t_2) \vee (depth(t_1) > depth(t_2))$ and
$(**) = \neg(substr(t_1, t_2) \vee (depthv(t_1) > depth(t_2))$.

Thus, if the co-synonym grade between the terms is equal to 1, they are supposed to be equivalent, if the Wu & Palmer measures between them (or the extended Wu & Palmer similarity in case of multi-words) is grater or equal to a certain threshold (th_2) then the terms are in a hypernym-hyponym relation. In this case the disambiguation is based on the depth() and the substr() function

as already described. A clarification deserves the use of the averaged or the minimized extended Wu & Palmer similarity. Premising that such distinction holds only in case of multi-words, the first version is used to distinguish the Hyponym-Hypernym case against the Related one, while the second version is used to distinguish the Related pairs against the Disjointed ones. The motivations behind such distinction are argued in chapter 5 by comparing both versions in the ROC curves.

2.3.1 Semantically-grounded alignment

The semantically-grounded aligner extends the set of alignment obtained with the previous methodology (A) by taking into account the α-consequences of aligned ontologies. By adopting the definition given in [1], α-consequences can be defined as all correspondences in the form $(t_1, t_2, <>)$ that satisfy the models of interpretation of the aligned ontologies, given a domain of interpretation, and an equalising function λ whose goal is to compare elements of different domains of interpretation. In few words, aligning two ontologies with logic-based axioms linking ontological classes together, generates other correspondences which are valid for all models of interpretation and so worth to be added to the alignment set. Table 2.6 shows all the α-consequences which can be generated from the alignment in the rows by using the equivalence and transitive inferences, where $c(t_i)$ refers to the class, from the input ontology, whose linguistic label is t_i. The question marks mean that in correspondence of that alignment (on the row) and that class relationship (in the column) there is not a satisfiable α-consequence, i.e., there is not a correspondence that is valid for all models of interpretation of the ontology.

classes $\rightarrow$	$c(t_2) \equiv c(t_i)$	$c(t_2) \sqsubset c(t_i)$	$c(t_2) \sqsupset c(t_i)$
$(t_1, t_2, \equiv)$	$(t_1, t_i, \equiv)$	$(t_1, t_i, \sqsubset)$	$(t_1, t_i, \sqsupset)$
$(t_1, t_2, \sqsubset)$	$(t_1, t_i, \sqsubset)$	$(t_1, t_i, \sqsubset)$	$(t_1, t_i, \sqcap)$
$(t_1, t_2, \sqsupset)$	$(t_1, t_i, \sqsupset)$	$(t_1, t_i, \sqcap)$	$(t_1, t_i, \sqsupset)$
$(t_1, t_2, \sqcap)$	$(t_1, t_i, \sqcap)$	$(t_1, t_i, \sqcap)$	?
$(t_1, t_2, \perp)$	$(t_1, t_i, \perp)$	?	$(t_1, t_i, \perp)$

Table 2.6: α-consequences obtained through the transitive inference

The α-consequences shown in Table 2.6 can be demonstrated set-

theoretically, by associating one set to each term's concept: t_1, t_2 and t_i.

2.4 Integration methodology

The mapping produced by the Aligner (included the α-consequences) is used to integrate the selected input ontologies to the output ontology along with the target concepts. The concepts inside the target ontology can be envisioned as *hub* concepts, i.e., as a glue that holds together all the aligned concepts from the input ontologies. This approach is similar to that described in [47]. Figure 2.12 tries to explain it: each box represents a cluster of concepts equivalent to the hub concept and so equivalent to each other in turn. The red concept is the cluster representative and is used to create links with other clusters representative, this way creating the class-superclass hierarchy in the integration ontology. Introducing clustering is convenient to reduce computational load of the integrating algorithm because it avoid confronting each concept with each other concept of aligned ontologies, thus limiting the matching operations to the cluster representative. In addition to clustering, another strategy that can be used to make scalable the integration algorithm in presence of very large ontologies is the *blocking* or *framing* technique. This consists in creating frames of clusters each relating to a knowledge domain or sub-domain, or to a specific aspect of the ontology. Thus, only the clusters in the same frame are confronted considerably reducing the effort for the integration phase. A similar approach can be also used for the matching and alignment stages of the framework.

The rules for handling the alignments in order to create the classes of the integration ontology are listed as follows:

- *equivalence*, if one or more input concepts are classified as equivalent with respect to a target hub concept, an ontology class is created for each input concept and another one for the hub concept. Adopting the criteria defined in section 2.1.5, whenever a class is created in the integration ontology, it will be named with the linguistic label available in the input ontology and annotated with the name space prefix of the ontology it come from. Furthermore, each metadata available in the input ontology will be used to enrich the description of the hub concept through the `rdfs:comments` property.

- *hyponymy*, if one or more concepts are classified as hyponyms w.r.t. a

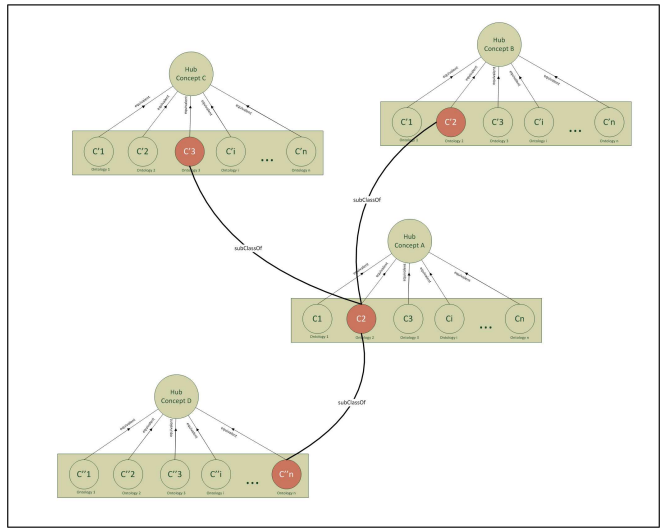


Figure 2.12: The ontology integration strategy

target hub concept, a corresponding class will be created in the integration ontology for each input concept and it will be declared as *subClassOf* the hub class.

Note that, according the criteria defined in section 2.1.5, only direct hyponym relations will be taken into consideration to be converted in class-subClassOf relation in the output ontology. This means that the integrator adopts a specific strategy in order to establishing if two terms classified as hyponym are actually direct hyponyms. The following cases are contemplated:

- the terms classified as hyponym are present in WordNet. In this case, the Integrator uses the JAWS APIs to establishing if one is the direct hyponym w.r.t. the other;
- at least one of the terms is a multi-word (not included in WordNet as multi-word). In this case the Integrator uses the following heuristics, after a POS-tagging operation:
  1. If the source term (src) is in the form `ADJECTIVE-NOUN` and the destination term (dst) in the form `NOUN`, then src is consider a direct hyponym of dst; e.g., *orange juice* vs. *juice*;

2. If the source term (src) is in the form `NOUN1-NOUN2` and the destination term (dst) in the form `NOUN` and `NOUN` and `NOUN2` are the same term, then src is consider a direct hyponym of dst; e.g., *apple cake* vs. *cake*.

2.5 The target ontology

In this work, the target ontology can be envisioned as the final goal of the integration approach but at the early phases of the proposed methodology also as a kind of *proto-ontology*, i.e, a raw set of domain keywords, terms, definitions or in general concepts related to the knowledge domain under study without a formal structure. It is not an ontology in the strict sense of the term but it just represents the *nucleus* of knowledge from where to start aggregating concepts from the top ranked ontologies matched throughout the framework components. The contribution of users with domain expertise is important in producing the target ontology because the choice of terms or concepts involved in this phase will affect heavily the choice of the knowledge repositories and the input ontologies to be collected in the *reference models corpus*. The target ontology will provide the terms against which to match all the terms in the input lexical chains. As already stated in the previous section, all concepts in the target ontology represent a kind of glue that put together the equivalent concepts coming from the other ontologies. Additionally, the target ontology, once converted in a lexical chain, is important in the extended linguistic analysis because it represents the target against which to evaluate the semantic coverage of the input ontologies by using the formula for the semantic grade defined in section 2.2.3.

The target ontology creation task is accomplished in the early phases of the methodology simultaneously with the knowledge domain identification and before the knowledge sources selection. The domain experts fix a number of meetings, actually brainstorming sessions, in which they agree on the nature of the target ontology, the topics of interest, and also they try to identify the questions the target ontology needs to answer. The methodology provided in this work welcomes the best practices described in [102] and includes the following tasks:

- Determine the domain and scope of the ontology. The matter experts must have in mind the following questions: What is the domain that the ontology will cover? For what we are going to use the ontology? For what types of questions the information in the ontology should provide answers?

- Individuate *competency questions*, i.e., a list of questions that a knowledge base based on the ontology should be able to answer. These questions will serve as the *litmus test* later: Does the ontology contain enough information to answer these types of questions? Do the answers require a particular level of detail or representation of a particular area? These competency questions are just a sketch and do not need to be exhaustive.

- Enumerate important terms in the ontology and start to think at a taxonomical hierarchy between the first chunk of terms.

- Selection of knowledge sources for the reuse of existing ontologies in the literature. With this phase, the target ontology creation task ends and it enters in the integration methodology described and proposed in this dissertation.

Chapter 3

Implementation of the ontology integration methodology

In this chapter, the software framework supporting the proposed ontology integration methodology will be described to an implementation level. This means that, starting from an overall view of the framework, technological solutions along with the third-party libraries and tools eventually used to overcome the encountered issues and to implement the framework functionalities will be detailed.

3.1 The framework implementation

The proposed framework has been implemented in Java language creating a stand-alone application (with some components equipped with GUI) using the Eclipse IDE[1] with the Maven [2] plugin installed for managing third-party libraries attached to the project. Figure 3.1 shows an high-level picture of the whole implementing architecture. Before describing each object in detail, a brief mention to all third-party libraries and their usage in the project will be provided as follows:

- *Apache Jena*[3], is a free and open source Java framework for building Se-

[1]Eclipse IDE website, https://eclipse.org/

[2]M2Eclipse plugin website, http://www.eclipse.org/m2e/

[3]Apache Jena website, Available online, https://jena.apache.org/

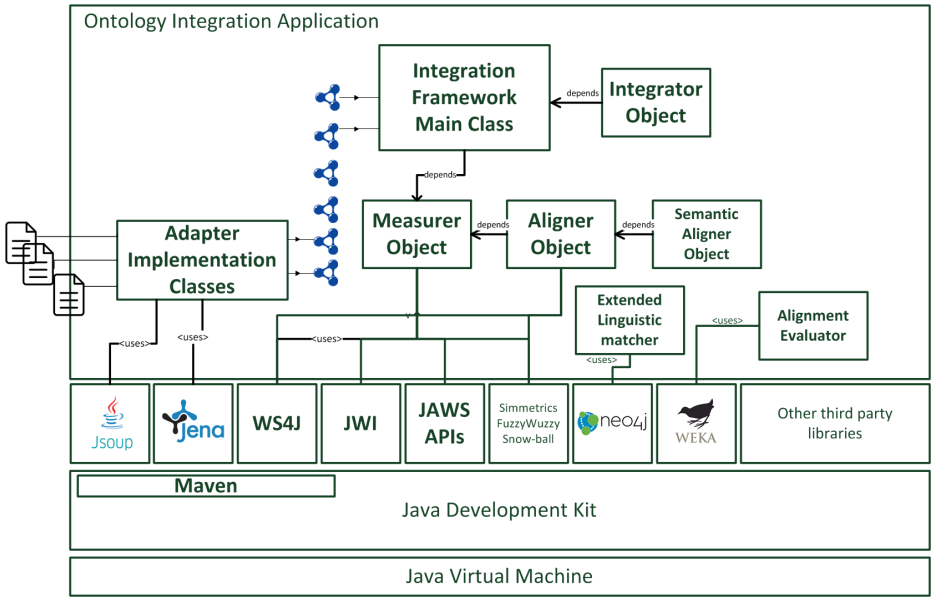


Figure 3.1: Implementation of the Integration Framework in Java with third party libraries

mantic Web and Linked Data applications. The framework consists of different APIs interacting together to process RDF data. In this context, Jena APIs have been used for transforming heterogeneous source models in homogeneous OWL-Lite ontology models using the Jena specification that means creating an in memory model with the transitive reasoner able to infer the transitive closure over the asserted axioms.

- *WS4J: WordNet Similarity for Java*[4], provides a pure Java API for several published Semantic Relatedness/Similarity algorithms such as: Wu and Palmer similarity (used in this work), LCH (Leacock and Chodorow, 1998), etc. This library is used to calculate the depth of a synonym in the WordNet hierarchy.

- *JWI: the MIT Java Wordnet Interface*[5], is a Java library for interfacing with

[4]WS4J Google code website. Available online, https://code.google.com/archive/p/ws4j/
[5]JWI website. Available online, http://projects.csail.mit.edu/jwi/

Wordnet. JWI supports access to Wordnet versions 1.6 through 3.0, among other related Wordnet extensions. In this work JWI APIs have been used to access WordNet and get all the synonyms rings associated to each term coming from the input lexical chains, if available. JWI APIs also allow to explore the hierarchy of hypernyms-hyponyms in order to extends the linguistic matching between two terms.

- *JAWS: Java API for WordNet Searching*, is an API that provides Java applications with the ability to retrieve data from the WordNet database. It is a simple and fast API that is compatible with both the 2.1 and 3.0 versions of the WordNet database files and can be used with Java 1.4 and later. In this work, it has been used in conjunction with the JWI APIs as it is more versatile and easy to use.

- *Simmetrics and FuzzyWuzzy APIs*, provides java-based methods which implement the most spread string matching algorithms, such as Levenshtein, Jaro, Jaro-Winkler, SimonWhite, Jaccard Similarity, BlockDistance, etc... The FuzzyWuzzy library provides practical methods for partial string matching that handle sentences like set of words (or ordered set of words) and relax the string matching algorithms according to the set theory.

- *JSoup*[6] is a Java library for working with real-world HTML. It provides a very convenient APIs for extracting and manipulating data, using the best of DOM (Document Object Model), CSS (Cascading Style Sheets) and jquery-like methods.

- *Neo4J*[7], is a highly scalable native graph database. It is used in this work for visualizing the Semantic Networks (SNs) created within the extended linguistic analysis (see chapter 2). It provides a Java APIs which allow user to programmatically access existing Neo4J graphs in order to read, modify or traverse the graph; it is also possible to create new graph with an effective and efficient way.

Over the third-party APIs or tools mentioned above, the whole software framework has been designed and implemented *ad hoc*, representing an integral part of the multi-strategy methodology proposed in this work. Particularly,

[6] Jsoup website. Available online, https://jsoup.org/

[7] Neo4J website. Available online, https://neo4j.com/

the reference module adapters (described in the following sections) have been conceived *from scratch* to make it practicable performing the alignment algorithm between heterogeneous and large knowledge models. Moreover, the main component of the framework, the Aligner, uses a decision tree-based classification function that has been designed and experimented specifically for this work. Finally, the integration strategy (together with the rules for an high-quality ontology integration process) have been implemented within the aims of this work. The same applies to the whole implementation of Semantic Networks, which exploits the WordNet linguistic database and Neo4J APIs to dynamically create the semantic expansion of each term in the input and target lexical chains.

3.2 Adapter implementation

Different Java classes have been implemented in order to adapt the input reference models. In most cases, a Jena OntModel object has been created passing it the source model file through the read method, as follows:

```
OntModel model =  ModelFactory.createOntologyModel(OntModelSpec.
    OWL_LITE_MEM_TRANS_INF);
model.read(sourceFilePath);
```

The `OWL_LITE_MEM_TRANS_INF` is a specification for OWL-Lite models that are stored in memory and use the transitive inferencer for additional entailments. Afterwards, a new OntModel has been created with the adapted level of syntax and then serialized in a uniform way using the `RDF/XML-ABBREV` language specification:

```
OntModel write_model = ModelFactory.createOntologyModel(
    OntModelSpec.OWL_LITE_MEM_TRANS_INF);
write_model.setNsPrefix("bbc", uri);
write_model.write(fileWriter, "RDF/XML-ABBREV");
```

A different implementation has been adopted for the NCIT, AGROVC and Eurocode 2 reference models adaptation. These models are further described in chapter 4 and in the Appendix, while, in the following subsections, all the implementation issues for adapting the above will be further described.

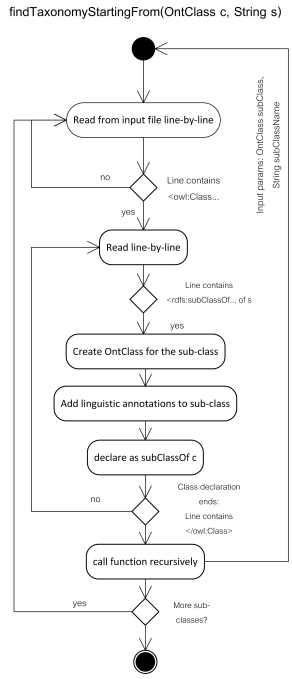


Figure 3.2: Activity diagram for the NCIT Adapter main function

3.2.1 National Cancer Institute Thesaurus adaptation

The main concern, while handling the NCIT source model, is its dimension. The owl source file is approximately 316 Mbyte and contains 118941 classes not all related to the Food domain or to the Food production domain. This means that adapting this model for the scopes of this work implied a kind of topic filtering on the NCIT entities (with summarization and slicing techniques). From the thousands of NCIT classes, only those belonging to the top-class named *Food* have been taken into account. In order to perform *filtering*, a kind of *scraper* has been implemented by leveraging the flexibility of JSoup APIs. Rather than import the owl file within a Jena model, task almost impossible to accomplish due to the source dimension, the file has been read line by line searching for ontology fragments related to the Food class. Figure 3.2 shows the activity diagram for the main recursive routine used in NCIT adapter, i.e., `findTaxonomyStartingFrom()`.

The routine reads line by line the source file and search for a line containing the class declaration whose name is equal to that passed through the String parameter. The first call to the function search for the *Food* top-class. Once found, the routine reads all `subClassOf` declarations and creates an OntClass object for each sub-class. Afterwards, each sub-class is declared as *subClassOf* the top-class and passed recursively to the same routine in order to construct the complete Food top-concept hierarchy. Once the Food fragment has been scraped from the NCIT source file, a new OntModel has been created according the procedure described in the previous section.

3.2.2 AGROVC Model Adaptation

A dimension issue has been encountered also for this model, in fact, the AGROVC core rdf file is approximately 900 Mbyte and contains 32000 concepts annotated in 27 different languages. In this case, it has been necessary a smarter crawler that was able to filter only the desirable fragments and to retrieve the English annotations (concept labels) attached to the filtered entities. Furthermore, the source file was written in SKOS language and so the transformation rules described in section 4.3 has been used. The adapter has focused only on `<skos:narrower>` and `<skos:broader>` properties which are equivalent to `<rdfs:subClassOf>` and its inverse.

As in the previous case, the source file has been read line by line and Java string functions has been used to handle the line. Particularly, the adapter uses three routines to scrape the needed entities: `getNarrower(String resource, OntClass c)`, which search for narrower concepts of resource, `getPreferredLabelUri(String resource, String lang)`, which retrieve the Uri of the label annotation in the specified language (lang), and `getPreferredLabel(String label_uri)`, which retrieves the actual string used for labelling the concept in the specified language. Figure 3.3 shows the `getNarrower()` routine.

3.2.3 Eurocode 2 Model Adaptation

The Eurocode 2 Fooding System taxonomy is available online as HTML pages: each top category with its sub-categories is listed in a html table. Each sub-category has a linked url referring to the sub-category description page (which contains its sub-category links in turn). Thus, in order to construct the classifi-

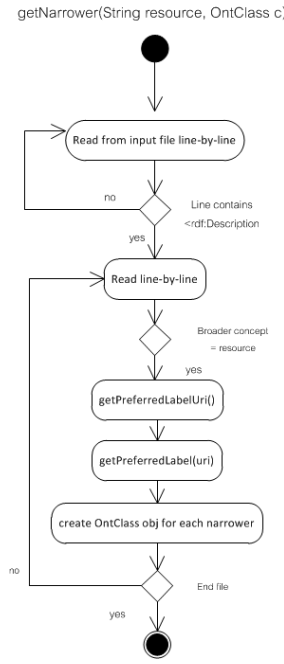


Figure 3.3: Activity diagram for the AGROVC Adapter main function

cation, it is necessary to load the html page and scrape category info from the table. Using JSoup java libraries it is possible to read the html document within a JSoup Document object and then get the table row elements by tag name. By iterating over the row elements, it is possible to retrieve the top-level category names which are embedded in a `<strong>` tag and then the second level categories info, by iterating over the cell elements `<td>`. These ones contain also a comment children that has been ported as `rdf:comment` in the adapted ontology.

3.3 Matching and aligning methodology implementation

The matching methodology has been implemented by means of three Java objects, namely: `TermsSetMeasuresCalculator`, `Aligner` and

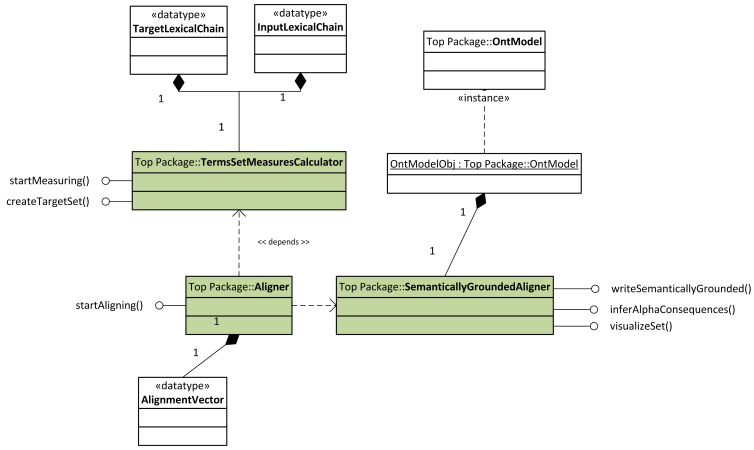


Figure 3.4: Class diagram excerpt for matching and alignment phase objects

`SemanticallyGroundedAligner`. The first takes the target and the input lexical chains (serialized in CSV: Comma Separated Values) and outputs a list of all lexicographic and linguistic measures for each terms pairs again in a CSV file (with extension .measures). The aligner takes the .measures file and uses it two obtain another CSV file (this time with extension .alignment) containing the same information of the measures file plus the predicted alignment between the terms pairs. Finally, the SemanticallyGroundedAligner object takes the ontological model and the .alignment file to obtain the α-consequences starting from the linguistic alignments and the asserted and inferred axioms in the input ontology model. The TermsSetMeasuresCalculator object uses WS4J, JAWS, simmetrics and fuzzywuzzy libraries in order to calculate the Path distance and the Wu & Palmer similarity, the synonymy and co-synonymy grade, the Levenshtein and Jaccard similarities, and, finally, the fuzzy string similarity, respectively. The Aligner uses a thresholds-based strategy and a decision tree to classify the alignment as *equivalent*, *hyponym-hypernym*, *related*, and *disjointed* as described in chapter 2. Figure 3.4 shows the class diagram of the mentioned above classes. In each calls are shown the main interfaces implemented as public methods and the main objects or data structures involved.

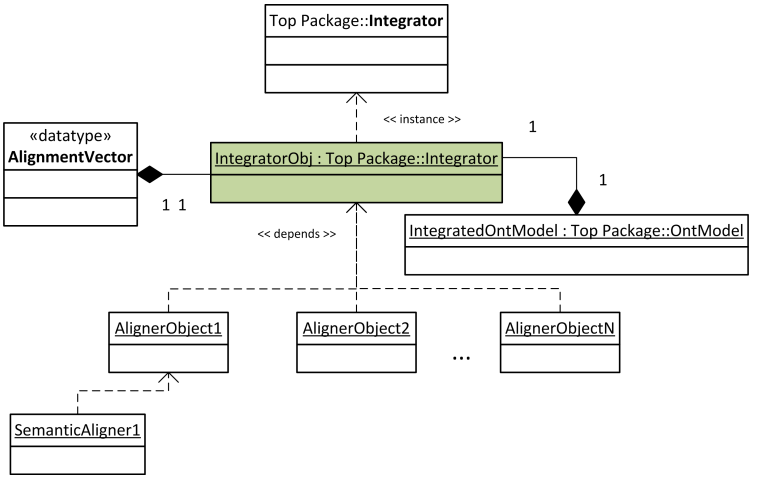


Figure 3.5: Class diagram excerpt for the integration phase objects

3.4 Integration methodology implementation

The actual integration of the selected input ontologies has been implemented in a Java class (`Integrator.class`) that creates an OntModel for the output ontology, adding the following namespace prefixes: *target* for the hub concepts coming from the target lexical chain and *ncicb*, *agrovc* and *eurocode2* for the selected input ontologies. The Integrator object reads the alignment files produced by the Aligner component plus the files created by the Semantic Aligner, then uses them according the integration strategy described in section 2.4 to obtain the output ontology. Figure 3.5 shows the class diagram of the Integrator.class and the other objects, data structure and classes mainly involved in the integration phase.

The output ontology contains 236 `owl:equivalentClass` and 943 `rdfs:subClassOf` asserted axioms, while the entailed `owl:equivalentClass` and `rdfs:subClassOf` are 1938 and 8531, respectively. These additional entailments have been obtained using the micro OWL rules inference engine when creating the Jena OntModel object, as follows:

```
OntModel model = ModelFactory.createOntologyModel(OntModelSpec.
    OWL_MEM_MICRO_RULE_INF);
```

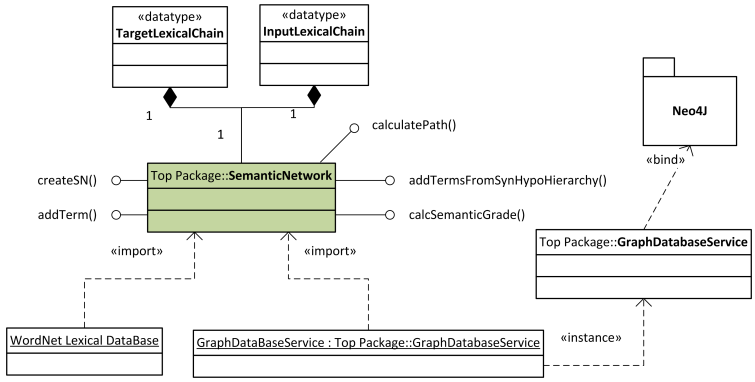


Figure 3.6: Class diagram excerpt for the SemanticNetwork class

3.5 Reference models grading implementation and Semantic Networks visualization

The extended linguistic analysis described in section 2.2.3 helps users to further select the input models from the corpus by applying a grading system (semantically-grounded). This analysis involves the creation of a Semantic Network for each input ontology. Technically, the Semantic Networks (SNs) have been implemented as properties-based labelled graphs through Neo4J GraphDB and are created starting from the meta-model described in section 2.2.3. This allows to import words and synsets from the WordNet lexical database and link them together through linguistic, semantic and semantic-linguistic relations. The SN for one input model is constructed starting from the terms in the corresponding lexical chain and expanding them with the terms linked to synsets belonging to an hyponyms or meronym chain that starts with the synsets linked to the initial terms and ends with any other synsets in the chain. A target SN is also constructed from the terms in the target lexical chain and added to the input SN. This way it is possible to evaluate how much two SNs are close to each other, at a semantic level, by calculating the Semantic Grade (SG) defined in section 2.2.3. In order to obtain the SNs as property-based and label graphs inside Neo4J, the procedure described in the following section has been applied. A set of rules for large graph visualization are also examined in the section.

3.5.1 Importing WordNet into Neo4J

The import of WordNet entities (synsets and words) inside Neo4J graphDB [103] has been implemented using the JWI WordNet APIs and Neo4J Java APIs according to a methodology described in [104, 105, 106, 107, 108]. The JWI APIs are used to access the WordNet linguistic database, while the Neo4J APIs to create an instance of Neo4J graphdb and populate it with the WordNet nodes. A Java class has been created, namely SemanticNetwork.class (Figure 3.6), in order to read the target and the input lexical chains and convert them in a graph. The workflow of the SemanticNetwork object is described as follows: initially, the target and the input lexical chains are imported in two String vectors, then they are used to search for the *semantic expansion* of the term, i.e., all the terms (retrieved from WordNet) that belong to the hyponym and meronym chain of the initial terms. Only hyponyms and meronyms are taken into account at this stage since these semantic relations are able to specialize the initial terms by semantically enrich the lexical chains without loss the specificity of the domain of discourse.

According to the meta-model described in section 2.2.3, the information which are needed in order to import the WordNet concern the synsets, the semantic relations among synsets, the words, the lexical relations among words and finally the links between the semantic and the lexical world, i.e., how a word is related to its concepts (or its meaning) and *vice versa*.

To be more specific, the following information are retrieved through the JWI methods:

1. For each synset involved in the expanded lexical chains:
   (a) *Id*: the univoque indentifier for the synset;
   (b) *SID*: the Synset ID as reported in the WordNet database;
   (c) *POS*: the synset's part of speech;
   (d) *Gloss*: the synset's gloss which express its meaning; and
   (e) the *semantic relations* among the synsets.
2. For each word involved in the expanded lexical chains:
   (a) *Id*: the univoque identifier for the word;
   (b) *WID*: the Word ID as reported in the WordNet database;

(c) *POS*: the word's part of speech;

(d) *Lemma*: lexical representation of the word;

(e) *SID*: the synset Id whose the word is related to;

(f) *lexical relations* linking the source and the destination words.

3. The lexical-semantic relations containing the following fields:

   (a) *Word Id*: the word id of the word that is linked to the synset on the right via the *hasConcept* relation;

   (b) *Synset Id*: the synset id of the synset that is linked to the word on the left via the *hasWord* relation;;

According to the meta-model, each synset and word has been converted into a node of the graph with label respectively: *Concept* and *Word*. Each semantic relation has become an edge between two concept nodes with the *type* property expressing the specific semantic relation holding between the concepts. Each lexical relation has been converted into an edge between two word nodes with a type property expressing the specific lexical relation between the word nodes. Finally, the word nodes have been connected to their related concept nodes through the *hasConcept* relation.

3.5.2 Large-scale representation of Semantic Networks

For each input lexical chain, several thousands of nodes have been created within the corresponding graph. Each node has labels and different properties each with labels in turn. Thus, the large scale visualization of the Semantic Network in a way that is elegant and human friendly at the same time, with the details of every labels, is a *chimera*, due to the dimension of the graph and the performance issues of the visualization tools, in particular when sophisticated drawing algorithms are used, and to the strongly connected nature of information to be represented, which often results in a messy and dense structure of nodes and edges. Figure 3.7 shows two arborescences of one input Semantic Network, but it contains near 2,500 nodes and as many relations, obtained through *Cytoscape* visualization tool [76]. The Neo4j running instance has been accessed via the cyNeo4j plugin, that converts the query results into Cytoscape table format. Afterwards, starting from the query tables, a view has been created by defining a custom style and the default layout. This latter is the already mentioned *Force-directed graph drawing*

algorithm that draws graphs in an aesthetically pleasing way by positioning the nodes of a graph in two-dimensional or three-dimensional space, so that all the edges are of more or less equal length and there are as few crossing edges as possible [109]. The resulting figure is more considerable for global analysis than for information that you can retrieve from it. Nevertheless, thanks to the force-directed algorithm, it is possible to observe agglomerates of nodes and edges which correspond to specific semantic categories. The figure does not shows labels attached to node or edges, but only uses different colours for synset nodes and word nodes (blue and red respectively), while a strong red is used to represent word nodes shared between the target and lexical chains (the terms belonging to the common intersection between the two), so that it is possible to evaluate, with just a look, how wide and dense is the area of shared concepts. The more numerous are the red dots, the more semantically close are the input model and the target one. A narrower red dots area, even if dots are numerous, means that the input model is specialized on a particular aspect of the knowledge domain.

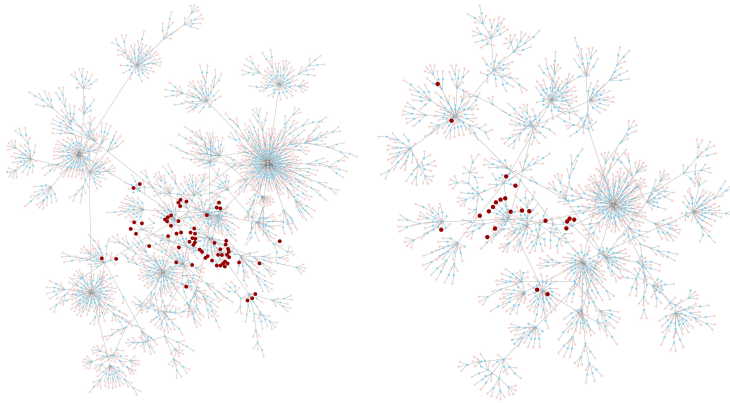
An attempt to visualize also the labels for the SNs nodes and edges has been made by establishing some aesthetic criteria as follows:

1. the efficiency of the visualization; i.e., avoid the information redundancy and the proliferation of useless signs and graphics as much as possible;

2. the effectiveness of the visualization; i.e., grant that the graphical representation of the network covers the whole informative content of the WordNet graph-based implementation;

3. the clearness of visualization, i.e., use light colours, such as gray, light blue, dark green, etc. with a proper level of brightness and with an appreciable contrast.

Thus, only *Words* labels have been visualized, avoiding to show again the lexical chain of words representing the corresponding concept into the synset nodes. These ones only show the synset ID as retrieved from the WordNet database inside the stretched blue oval for gthe sake of traceability w.r.t. the lexical databse. Furthermore, since for each *Hyponym* relation between synsets corresponds an *Hypernym* relation, only one of the two, i.e. the *Hyponym*, has been explicitly drawn in order to increase the clearness of the representation. The same approach has been adopted for the other pairs of antinomian relations



(a) (b)

Figure 3.7: The broader Semantic Network arborescences with the shared terms highlighted

like *Meronym-Holonym*. This strategy does not affect the effectiveness criteria, in fact, even if there is not an explicit representation of the *Hyponym* relations, these ones can be inferred from the corresponding *Hypernym* ones. Each synset is linked to the corresponding lemmas through the *hasConcept* relation which has been represented with a dashed line in light gray without an explicit label. This improves the efficiency of the visualization and the effectiveness, since no informative contents is sacrificed for clearness. Figure 3.8 shows an excerpt of the SemanticNetwork depicted previously using this visualization criteria.

3.6 Querying and traversing the Semantic Networks

The greatest value of importing WordNet database into a Neo4j graph, it is not only related to the graph visualization capabilities of the Neo4J web visualizer or other tools like Cytoscape, but, mainly, to the power of the *Cypher* language [110], a declarative query language that allows for expressive and efficient querying and updating of the graph store. Since very complicated database queries can easily be expressed through Cypher, this allows the user to focus on the data model domain instead of getting lost in database access. Most of the keywords like WHERE and ORDER BY are inspired by SQL, while pattern matching bor-

rows expression approaches from SPARQL [75]. In this section some of the queries used in the proposed framework are discussed:

3.6.1 Query 1: getting random lexical-semantic relations

The following Cypher query get all relations which link *Synset* nodes to *Word* nodes via the *hasWord* lexical-semantic relation (by limiting the results to 25 relations):

```
MATCH (s: Synset)-[r:hasWord]->() RETURN r LIMIT 25
```

It appears clear that a graph-based query language is most suitable in order to select sub-graphs, as in this case. In fact, it comes very natural to select a bunch of nodes and relations just by using patterns and pattern-matching, expressed in an intuitive and iconic syntax, to describe the shape of the data you are looking for [111].

3.6.2 Query 2: getting specific synsets and synonyms rings

Starting from the previous query this one search for all synset nodes linked to Word nodes with the Lemma property equal to *food*:

```
MATCH (s: Synset)-[r:hasWord]->(w:Word {lemma: 'food'}) RETURN r
```

This query is quite interesting because it returns three synset nodes each attached to one word node. In fact, three are the *food* senses contained in WordNet, i.e.:

```
1. (34) food, nutrient -- (any substance that can be metabolized
     by an organism to give energy and build tissue)
2. food, solid food -- (any solid substance (as opposed to
    liquid) that is used as a source of nourishment; "food and
    drink")
3. food, food for thought, intellectual nourishment -- (anything
     that provides mental stimulus for thinking)
```

More interestingly, starting from the previous query it is possible to retrieve the synsets rings as defined in chapter 2, i.e, the set of synonyms for the three senses of the *food* concept in WordNet by launching the following more complex query:

```
MATCH (s: Synset)-[r: hasWord]->(w:Word {lemma: 'food'}), (s)-[v
    : hasWord]-(t: Word) RETURN s,v,r,t
```

Although it may appear quite complex at a first glance, this query just adds another matching rule to the previous one, i.e., that for obtaining all other words related to the synset returned by the first matching, that is, the synsets linked to a Word node with the *food* label.

3.6.3 Query 3: getting all hyponyms/hypernyms of a specific synset

The flexibility and iconicity of Cypher language allows to easily move across the hyponymm-hypernym hierarchy. For example, starting from query 2, it is possible to ask Neo4J for all hyponyms of the synsets labelled as *Food* (linked via the semantic relation *hyponymy*), by adding a match clause as in the following query:

```
MATCH (s: Synset)-[r: hasWord]->(w:Word {lemma: 'food'}), (t:
    Synset)<-[v: Hyponym]-(s) RETURN s,v,t
```

Noteworthy, the iconicity of the arrow direction used in the clause `(t: Synset)<-[v: Hyponym]-(s)` allows to immediately get hyponyms or hypernyms even if the latter are not explicitly declared. For example, by just changing the arrow direction it is possible to get all the hypernyms of *Milk*-labelled synset:

```
MATCH (s: Synset)-[r: hasWord]->(w:Word {lemma: 'milk'}), (t:
    Synset)-[v: Hyponym]->(s) RETURN s,v,t
```

3.6.4 Query 4: getting specific arborescences

Cypher allows to retrieve synsets or words linked by an arbitrary number of hops through semantic or lexical relations. Thus, starting from specific synsets (e.g., the *Food*-labelled) it is possible to obtain the wider connected sub-graphs engulfing such synsets. These connected sub-graphs are also mentioned as arborescences. For example, the following query returns all the synsets linked through the hyponym relation to the three synset of *Food* discounting the number of hops which separate the first w.r.t. the last synset.

```
MATCH (s: Synset)-[r: hasWord]->(w:Word {lemma: 'food'}), (t:
    Synset)<-[v: Hyponym*]-(s) RETURN s,v,t
```

A more holistic version of the qyery is:

```
MATCH (s: Synset)-[r: hasWord]->(w:Word {lemma: 'food'}), (t:
    Synset)<-[v*]-(s) RETURN s,v,t
```

which returns the arborescences obtained linking all synsets discounting the number of hops which separate the nodes and regardless the specific semantic relation in hop. This way can be taken into consideration also meronyms. Even more holistic is the following:

```
MATCH (s: Synset)-[r: hasWord]->(w:Word {lemma: 'food'}), (s)-[v
    *]-(t) RETURN s,v,t limit 100
```

where the second clause `(s)-[v*]-(t)` returns all nodes and relations linked to *Food* synsets regardless the nature of the source and the destination node. Figures 3.7 are obtained from this last query.

3.6.5 Traversing the Semantic Network

Neo4J Java APIs and Cypher offers methods for traversing the graph through the identification of the shortest path between the source and the destination nodes. Several algorithms can be used to search for the shortest path, such as the Dijkstra's algorithm [112], the Bellman–Ford algorithm and the A* search algorithm [113]. The Cypher query to get all the paths between two nodes is as follows (in this example between the Word node *food* and *hamburger* respectively):

```
MATCH (src:Word { lemma:"food" }),(dst:Word { lemma:"hamburger"
    }), p = (src)-[*]-(dst) RETURN p
```

In order to search for the *shortestPath* a convenient function can be used in the query like in the examples below:

```
MATCH (src:Word { lemma:"food" }),(dst:Word { lemma:"hamburger"
     }), p = shortestPath((src)-[*]-(dst)) RETURN p
```

Finally, since the Semantic Network has weights associated to the semantic relations, it is possible to find the Weighted Shortest Path by means of the following query:

```
MATCH (src:Word { lemma:"food" }),(dst:Word { lemma:"hamburger"
    }), p = shortestPath((src)-[*]-(dst))
RETURN p AS shortestPath, reduce(w=0, r in relationships(p) |  w
    +r.w) AS totalDistance ORDER BY totalDistance ASC LIMIT 1;
```

More complex query are able, for example, to find the deepest hyponymm of *Food*:

```
MATCH (src: Synset)-[t: hasWord]->(w:Word {lemma: 'food'}),(dst
    : Synset) MATCH p=src-[r*1..]-dst
WITH length(p) AS l,src,dst  WHERE l>=8  RETURN src,dst,l
```

The query above is able to return the destination node that is distant a number of hops greater or equal to 8. It turns out that such node is *French_dressing_for_fruit_salad*, which has the following hierarchy of hypernyms:

```
French dressing for fruit salad;
 French dressing;
  Dressing, salad dressing, etc.
   sauce, etc.;
    condiment, etc.;
     flavorer, etc.;;
      ingredient, fixings, etc.;
       foodstuff, food product;
         food, nutrient.
```

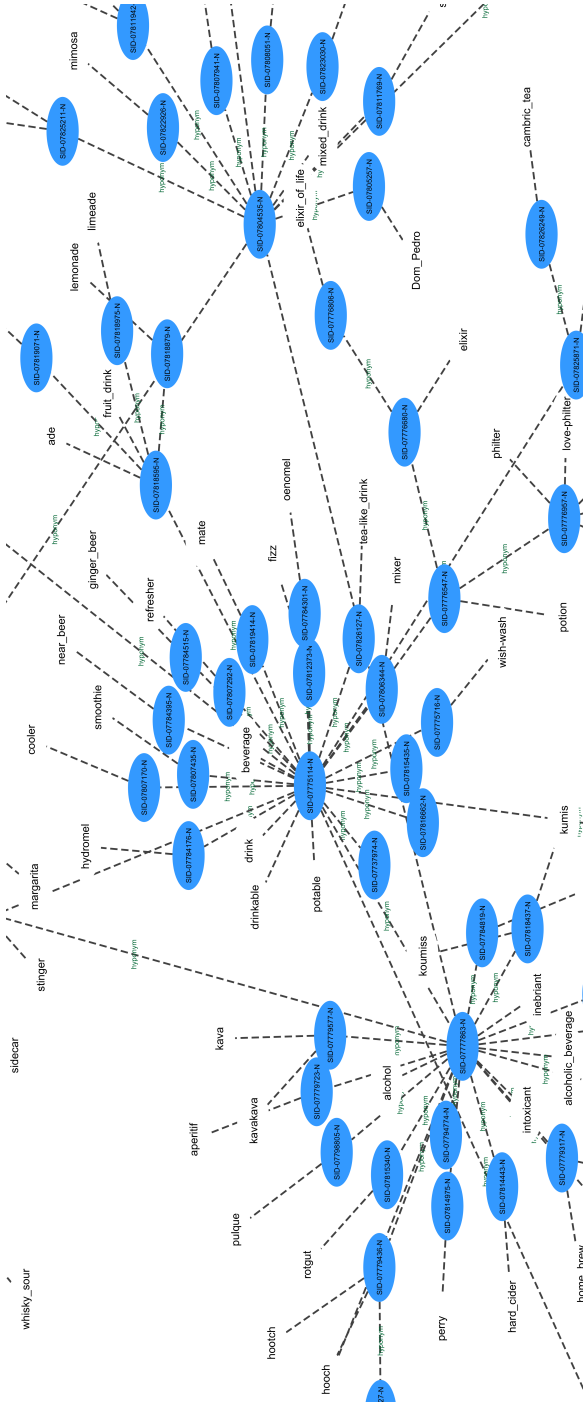


Figure 3.8: Implementation of the Integration Framework in Java with third party libraries

Chapter 4

Applying the methodology to a specific domain case-study

This chapter applies the methodology proposed in this work to the *food* domain, specifically to the *production of food*, encompassing concepts like *food product* and *food product categories*. The remainder of the chapter is structured as follows: the first section introduces the domain under study by further characterizing it, while, from the second section to the final one, all the methodology's phases described in chapter 2 are applied to the case-study along with the considerations that have eventually arisen.

4.1 A case study from the Food and Food production domain

Since the food is an *umbrella* topic involving concepts related to different disciplines and applications, it represents a valid benchmark for the proposed methodology. This pursues a twofold objective: on the one hand, it selects the retrieved reference models, only those whose main concern is on the *production of food* and *food product categories*, while, on the other hand, it integrates them in a coherent and homogeneous view of the final ontology. The term "food" can be found in different knowledge models such as: recipes for dishes served in a restaurant, biomedical thesaurus, commercial products catalogues and many others. Thus, the main result expected here is to select the reference models that

best fits the specific domain under study discarding all models which are too generic (upper level ontologies) or highly specialized w.r.t. to the target. In order to select the best reference models, as already described in the previous chapters, the retrieved reference models are subjected to two selections: a qualitative and a semantically-grounded selection. The first applies the criteria described in 2.1.1 for a preliminary skimming of the models corpus, while the second one involves all lexicographic, linguistic and semantic analysis described throughout the sections of chapter 2. The target ontology defined at the end of the chapter 2 is strategic for this selection because, as already stated, it represents the target against which to evaluate the semantic coverage of the input ontologies. Each block in Figure 2.1 will be applied as follows.

4.2 Knowledege sources selection

The collaboration with users with domain expertise is precious in this phase of the methodology in order to individuate knowledge sources from which to extract source models. Google search engine, Google Scholar, ISO International Classification of Standards and OAEI Iniziative Food test cases suit best in this case. Also specialized portal like BioPortal have been taken into consideration. The harvesting of reference models has been executed mostly manually even though some tools for automatizing search queries over Google Scholar, for example, have been successfully experimented [1]. As a result of applying this phase, many reference models, gathered in the reference *corpus*, have been collected. All these models have been subjected to the first selection according to the evaluation criteria discussed in section 2. In particular, a greater weight has been given to reference models constructed in OWL or RDF language, these ones being the final languages used in the integrated ontology, and a greater weight has also been given to availability (open model data have been preferred) and model provenance (a high rank has been given to standard data or data coming from influential research initiative or groups). Appendix section provides a description of each reference model and some information about their provenance. Table 4.1 shows the list of the reference models with the values fields filled according to the selection criteria.

[1] scholar.py, A parser for Google Scholar, written in Python. Available online: https://github.com/ckreibich/scholar.py

N.	Reference model	Formality	Generality	Structure	Lang.	Provenance	License
1)	National Cancer Institute Thesaurus	Formal	Domain	Ontology	OWL	Non-stand	Open
2)	AGROVOC	Semi-formal	Domain	Ontology	RDF	Non-stand.	Open
3)	BBC Food Ontology	Semi-formal	Domain	Ontology	RDF	Other	Open
4)	LIRMM	Semi-formal	Domain	Ontology	RDF	Other	Open
5)	The Product Types Ontology	Semi-formal	Application	Ontology	RDF	Non-stand.	Open
6)	Eurocode 2 Food Coding System	Informal	Domain	Classification	Text	Non-stand.	Open
7)	WAND Food and Beverage Taxonomy	Semi-formal	Domain	Taxonomy	Text	Private companies	Proprietary
8)	Food technology ISO Standard	Semi-formal	Domain	Taxonomy	Text	Stand. Organiz.	Proprietary
9)	Foodon food ontology	Formal	Upper-Domain	Ontology	OWL	Stand. Organiz.	Open

Table 4.1: Selected Reference Models Analysis

4.3 The adaptation and normalization phase

Going forward, the adaptation and normalization function block has obtained a set of normalized, language-agnostic and de-structured data models. As described in chapter 2, the adaptation module takes a list of source models which are heterogeneous, in syntax and level of details, and obtains a list of OWL-Lite ontologies serialized in owl files. The solutions adopted in order to transform the source model in a proper owl lite ontology are different and depend on the nature of the source model. In general, if an ontological model (RDF or OWL) is available for the source model, it can just been imported in a Java class and loaded inside a Jena ontology model. As described further in section 3, the Jena model has the `OWL_LITE_MEM_TRANS_INF` specifications and supports the OWL-Lite expressiveness and the transitive reasoner. Afterwards, the adapted module become an ontology to all effects and is serialized in a owl file using the syntax `"RDF/XML-ABBREV"`. Looking at table 4.1, models 3), 4), 5) and 9) have been subjected to the procedure just described. Conversely, models 1) and 2) have required more attention due to their dimension. In particular, the National Cancer Institute Thesaurus (NCIT) has 118941 classes and 46839 individuals as

Metrics:	no.
Number Of Classes	118941
Number Of Individuals	46839
Number Of Properties	173
Maximum Depth	16
Maximum Number Of Children	3235
Average Number Of Children	6
Classes With A Single Child	8509
Classes With More Than 25 Children	750
Classes With No Definition	36013

Table 4.2: National Cancer Institute Thesaurus metrics

reported in table 4.2 (adopted from the NICT website [2]), while the AGROVC Multi-lingual Thesaurus presents more than 32000 concepts having labels in up to 27 languages. In order to deal with these numbers, the adapter modules for the NCIT and AGROVC Thesaurus have acted as a kind of crawlers searching for ontology fragments specifically regarding the Food and Food product domain and, in the case of AGROVC, which is multi-lingual, by selecting the linguistic annotations (class labels) related to the English language. AGROVC has required further adaptations for a twofold reason: it is written in SKOS language and the classes are identified with an ID which need to be resolved in order to retrieve the linguistic annotations associated to the class (label in different languages or the preferred label). Figure 4.1 draws en excerpt of the AGROVC Thesaurus. The first issue has been tackled using meta-model transformation rules as shown in Table 4.3, while the second one by constructing a class-id *resolver* that scrapes the preferred label from a class given its id. Once the fragments have been scraped from the source model they have been transformed into an OWL Lite model and serialized in an OWL file like for the previously described cases. Table 4.9 show an excerpt for the adapted NCIT, AGROVC and Eurocode2 ontologies, respectively.

The Eurocode 2 Fooding System taxonomy (6) is available online as HTML pages: each category has its own web page with a description for the category and the list of its sub-category. Each sub-category has a linked url referring to the sub-category description page (which contains its sub-category links in

[2]National Cancer Institute Thesaurus (NCIT), https://bioportal.bioontology.org/ontologies/NCIT

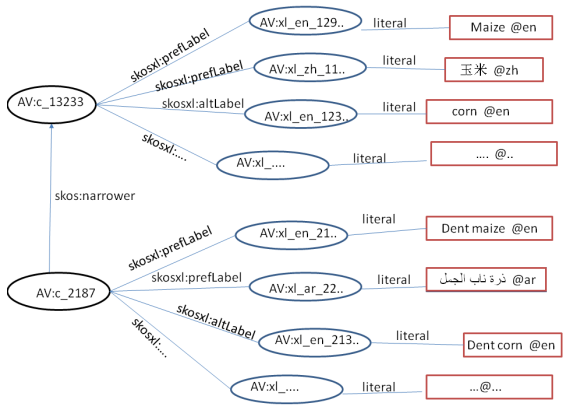


Figure 4.1: AGROVOC concepts linked by the property skos:broader (hierarchical relation). From http://aims.fao.org/

turn). Thus, in order to construct the classification, it is necessary to load the html pages and scrape the info about the categories. For doing this, the JSoup java libraries have been used. JSoup allows to extract and manipulate data from html markup by filtering tag elements on the base of the *tag name*, the name of a particular *attribute* of the tag or its value. Eurocode 2 taxonomy has been cut to the third level because categories at a level greater than two provide too specific concepts, which are out of the scope of the final ontology and so they have been discarded. Finally, since the source models 7) and 8) are not entirely available because licensed, they have been adapted from a textual representation of their available fragments in Internet. They have not been taken into account in the successive phases of the framework because one of the selection criteria for filtering the reference models is their availability as open source models.

Once the owl lite ontologies have been obtained from the adapter modules,

Meta-model entity	OWL/RDF entity	SKOS entity
Concept	owl:Class	skos:Concept
A hyponym B	A rdfs:subClassOf B	A skos:narrower B
A hypernym B	B rdfs:subClassOf B	A skos:broader B

Table 4.3: OWL/RDF respect to SKOS meta-model conversion rules

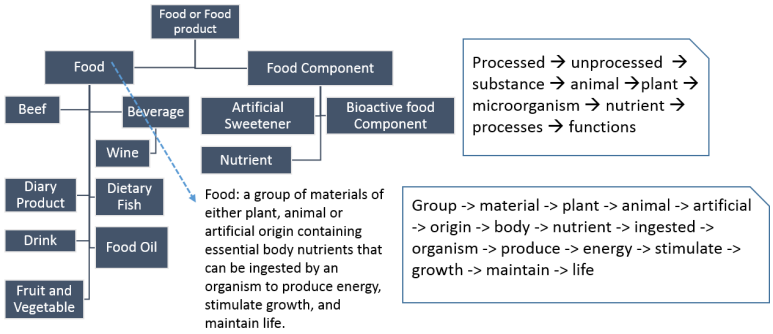


Figure 4.2: Ontology Excerpt and Lexical Chain

another operation needed before accessing the matching phase if the flattening. All linguistic labels attached to classes in the adapted modules have been extracted from the hierarchy to create a flatted lexical chain for each ontology. This because the Matcher component do not apply any structural analysis in this phase, but simply compare each term from one input lexical chain with each term from the target lexical chain obtaining a list of lexicographic and linguistic similarity measures. Figure 4.2 tries to picture this. The terms in the lexical chains need to be processed by the text-processing pipeline described in chapter 2. In particular, the operations that were necessary for the input lexical chains are: punctuations sign elimination, stop word elimination, plural form word singularization, part-of-speech tagging and lemmatization.

4.4 The matching phase

As already described in section 2.2, the matcher is responsible for obtaining a set of similarity measures taking as inputs the terms coming from the source models lexical chains and the terms from the target lexical chain. Table 4.4 shows an excerpt of the similarity table for the NCIT thesaurus and the target ontology. The table shows the terms pairs on the left (*src* and *dst*) and nine similarity measures grouped in to two categories: lexicographic and linguistic category. To the first one belong: the exact string matching (*str*), the partial string mathcing (Levenshtein similarity, *lev*), the Jaccard similarity (*jac*) applied only in case of multi-words and the fuzzy string matching (*fuz*). To the second

src	dst	str	lev	jac	fuz	syn	cos	wup	path	wup*
food	food	1.0	1.0	0.0	1.0	1.0	1.0	1.0	1.0	0.0
meat	meat	1.0	1.0	0.0	1.0	1.0	1.0	1.0	1.0	0.0
juice	juice	1.0	1.0	0.0	1.0	1.0	1.0	1.0	1.0	0.0
food	nutrient	0.0	0.0	0.0	0.0	1.0	0.33	1.0	1.0	0.0
food	solid_food	0.0	0.4	0.0	1.0	1.0	0.33	1.0	1.0	0.95
dairy product	dairy_product	0.0	0.92	1.0	0.0	1.0	1.0	0.0	0.0	0.72
chocolate	cocoa	0.0	0.56	0.0	0.0	1.0	0.25	1.0	1.0	0.0
swordfish	seafood	0.0	0.22	0.0	0.0	0.0	0.0	0.88	0.33	0.0
taro	sundowner	0.0	0.11	0.0	0.0	0.0	0.0	0.88	0.09	0.0
tuna	seafood	0.0	0.0	0.0	0.0	0.0	0.0	0.88	0.33	0.0
venison	meat	0.0	0.14	0.0	0.0	0.0	0.0	0.88	0.33	0.0
whey	food_product	0.0	0.0	0.0	0.0	0.0	0.0	0.88	0.33	0.67
whey	foodstuff	0.0	0.0	0.0	0.0	0.0	0.0	0.88	0.33	0.0
applesauce	fresh_food	0.0	0.0	0.0	0.0	0.0	0.0	0.5	0.11	0.8
applesauce	fresh_foods	0.0	0.0	0.0	0.0	0.0	0.0	0.5	0.11	0.8
buttermilk	fresh_food	0.0	0.0	0.0	0.0	0.0	0.0	0.5	0.1	0.8
buttermilk	fresh_foods	0.0	0.0	0.0	0.0	0.0	0.0	0.5	0.1	0.8
white wine	beverage	0.0	0.2	0.0	0.0	0.0	0.0	0.0	0.0	0.91
corn syrup	beverage	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.8
sage tea	beverage	0.0	0.12	0.0	0.0	0.0	0.0	0.0	0.0	0.8
egg white	beverage	0.0	0.11	0.0	0.0	0.0	0.0	0.0	0.0	0.78
corn oil	beverage	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.75
goat milk	beverage	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.73
organic food	beverage	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.73
cherry juice	beverage	0.0	0.17	0.0	0.0	0.0	0.0	0.0	0.0	0.67
alcoholic beverage	beverage	0.0	0.44	0.5	1.0	0.0	0.0	0.0	0.0	0.65
beef	meat	0.0	0.25	0.0	0.0	0.0	0.0	0.93	0.5	0.0
fowl	meat	0.0	0.0	0.0	0.0	0.0	0.0	0.93	0.5	0.0
lamb	meat	0.0	0.0	0.0	0.0	0.0	0.0	0.93	0.5	0.0
mutton	meat	0.0	0.33	0.0	0.0	0.0	0.0	0.93	0.5	0.0
sardine	meat	0.0	0.0	0.0	0.0	0.0	0.0	0.75	0.2	0.0
scallop	meat	0.0	0.14	0.0	0.0	0.0	0.0	0.75	0.2	0.0

Table 4.4: Excerpt of similarity measures between NCIT and target ontology terms

category belongs: the synonymy grade (*syn*), the co-synonymy grade (*cos*), the Wu & Palmer similarity (*wup*), the path distance (*path*) and, finally, the extended Wu & Palmer (*wup**) applied only in case of multi-words. The *str* measure, together with *syn* and *cos*, allows to retrieve equivalent concepts by taking into account two main issues of spoken languages: *synonymy* and *polysemy*. The first issue concerns the fact that one concept can be expressed by several terms (synonyms), while the second issue concerns the fact that different concepts can be expressed with the same term. The co-synonymy grade allows to relate two terms w.r.t. all the possible meanings they have in a spoken language.

From table 4.4 the following examples are analysed:

- *food* vs *food, meat* vs *meat, juice* vs *juice*, in this case the exact string measure, the synonym and cosynomym grade are equal to 1.0, so all the pairs are classified as equivalent classes in the aligning phase.

- *food* vs *nutrient*, in this case while the exact string measure and the syn-

onym grade are equal to 1.0, the co-synonym grade is equal to 0.33. This means that *food* and *nutrient* have one or more senses not shared by both and so there are some contexts in which it can be used one but not the other. For example, from WordNet, *food* has three senses (and synsets): 1. food, nutrient – (*any substance that can be metabolized by an organism to give energy and build tissue*); 2. food, solid food – (*any solid substance (as opposed to liquid) that is used as a source of nourishment; "food and drink"*); 3. food, food for thought, intellectual nourishment – (*anything that provides mental stimulus for thinking*); while nutrient only one, which is shared with food (the first sense on food). For this reason the co-synonymy grade is $\frac{1}{3}$, i.e., the number of shared senses divided by the sum of senses of the both terms. According to the procedure described in section 2.1.4, they are considered equivalent, based on a heuristic that considers terms coming from the input lexical chain and the target one sharing the same sense since they belong to the same domain of discourse. Anyway, in this case a *red flag* is raised in order to induce users to verify the alignment.

- *dairy product* vs *dairy_product*. In this case the terms are multi-words. They do not match exactly but refer to the same concept (from WordNet: dairy product – (*milk and butter and cheese)*). In fact, the synonym and co-synonym grade are equal to 1.0, so they are exactly the same concept.

- *chocolate* vs *cocoa*. They are synonyms in WordNet and have a synonymy grade equal to 1.0. Although they have a low co-synonym grade are classified as equivalent for the same reason described in the previous item;

- *swordfish* vs *seafood*. In this case all lexicographic similarities are equal to 0.0 but the Levenshtein that is very low. The synonymy and co-synonymy grades are 0.0, but Wu & Palmer similarity and the path distance give 0.88 and 0.33, respectively. The Aligner uses Wu & Palmer similarity in order to decide how to classify pairs based on a threshold as described in chapter 2 (in this case they are classified as hyponyms). Note that, in this case, Wu & Palmer similarity is a more accurate measure in relating terms than the path distance since it normalizes the distance between them w.r.t. the distance between the common ancestor (of the two) and the hierarchy root concept (see chapeter 2 for a complete description). Example similar to this are *beef* vs *meat*, *fowl* vs *meat* and so forth.

src	dst	rel	src	dst	rel
food	food	$\equiv$	meat	meat	$\equiv$
juice	juice	$\equiv$	food	nutrient	$\equiv$
food	solid_food	$\equiv$	dairy product	dairy_product	$\equiv$
swordfish	seafood	$\sqsubset$	taro	sundowner	$\sqsubset$
tuna	seafood	$\sqsubset$	venison	meat	$\sqsubset$
whey	food_product	$\sqsubset$	whey	foodstuff	$\sqsubset$
egg white	beverage	$\sqsubset$	corn oil	beverage	$\sqsubset$
goat milk	beverage	$\sqsubset$	organic food	beverage	$\sqsubset$
cherry juice	beverage	$\sqsubset$	alcoholic beverage	beverage	$\sqsubset$
beef	meat	$\sqsubset$	fowl	meat	$\sqsubset$
lamb	meat	$\sqsubset$	mutton	meat	$\sqsubset$
sardine	meat	$\sqsubset$	scallop	meat	$\sqsubset$
food	cheese	$\sqsupset$	food	chocolate	$\sqsupset$
beverage	juice	$\sqsupset$	beverage,	milk_shake	$\sqsupset$
beverage	pledge	$\sqcap$	beverage	potable	$\sqcap$
beverage	potation	$\perp$	food	shandy	$\perp$
fowl	Pallets Tertiary	$\perp$	fowl	Freezing	$\perp$

Table 4.5: Excerpt of alignment between NCIT and target ontology terms

- *cherry juice* vs *beverage*. In this case *src* is a multi-word and is not present in used version of WordNet, so it is tokenized and the Jaccard measure along with the extended Wu & Palmer (wup*) measure are calculated. This latter, in particular, uses the average of the Wu & Palmer similarities calculated over each pair of tokens coming from the compared multi-words (cherry-beverage, juice-beverage). In this case, the extended averaged Wu&Palmer measure is 0.67, and, accordingly to the choice of threshold t_1, the Aligner to classifies the terms as *hyponyms*.
- *Food* vs *brewing*. src and dst in this case are single word, moreover, both are present in WordNet and have a Wu&Palmer measure equal to 0.4. Thus, the classifier put them in the *related* class;

4.5 Aligning methodology

The Aligner module acts as a classifier, i.e., it classifies the terms pairs according to the measures calculated by the matcher and a decision tree discussed in

Hierarchy (fragment)	Alignment	α-consequences
ncicb:Food ncicb:Fruit_and_Vegetable ncicb:Brassica_Vegetable ncicb:Collard_Greens ncicb:Mustard_Greens ...	... Food $\equiv$ solid_food Food $\sqsupset$ food_product Food $\sqsubset$ product	... Mustard Greens $\sqsubset$ solid_food Mustard Greens $\sqsubset$ food_product Collard Greens $\sqsubset$ product Collard Greens $\sqsubset$ solid_food
ncicb:Food ncicb:Fruit_and_Vegetable ncicb:Peppers ...	... Food $\sqsupset$ food_grain Food $\sqsupset$ grain	... Peppers $\sqcap$ grain Peppers $\sqcap$ food_grain
ncicb:Food ncicb:Meat ncicb:Fish_Vertebrates ...	... Food $\sqsupset$ seafood Food $\sqsupset$ fresh_foods Food $\equiv$ food	... Fish_Vertebrates $\sqsubset$ seafood Fish_Vertebrates $\sqsubset$ food Fish_Vertebrates $\sqsubset$ fresh_foods

Table 4.6: Examples of α-consequences from the NCIT ontology

section 2.1.4. Table 4.5 shows a fragment of the alignment for the NCIT Ontology. Noteworthy, *cherry juice* and *beverage* have been classified as *hyponyms* because of the extended Wu & Palmer measure obtained from the Mathcer and the heuristic described in section 2.3. Furthermore, *beverage* and *pledge* are classified as *related* because of the Wu & Palmer measure and, in fact, they refer to concepts which have a weak semantic link. Finally, terms like *fowl* and *freezing* are classified as disjointed according to the fact that they refer to different semantic categories.

4.5.1 Semantic-grounded aligning methodology

The Semantically-grounded Aligner adds all α-consequences to the alignment provided by the Aligner, according to what described in section 2.3.1. Table 4.6 shows some of the α-consequences added to the alignment for the NCIT Ontology starting from the linguistic alignment and using the transitive or equivalence rules over the NCIT ontology.

4.6 Extended linguistic analysis

As described in section 2.2.3, the extended linguistic analysis is a complementary strategy added to the whole approach proposed in this work. It converts

the input and target lexical chains in Semantic Networks (SNs) containing an extended set of concepts w.r.t. the initial set from the lexical chain. This new set encompasses hyponyms, hypernyms, meronyms, which are retrieved from WordNet. Furthermore, the concepts of the semantic network are linked to each other with the linguistic and semantic relations provided in the meta-model described in chapter 3. The SNs provide a conceptual frame useful to discriminate the pertinent reference models from the other ones throughout a system grading that is able to assign a vote to the model based on their syntactic and semantic content. In this work, the approach described in section 2.2.3 is used to calculate a *Global Grade* (GG) for each semantic network related to each selected reference model. The GG is given by the sum of the *Syntactic-Semantic Grade* (SSG) and the *Semantic Grade* (SG) as described in detail in section.

Table 4.7 show the Global Grade, the semantic and Syntactic-Semantic Grade for the input ontologies, while figure 4.3 show the Semantic Network containing the expanded lexical chains of NCIT ontology and the Target ontology according the described procedure. The SNs have been implemented by using the Neo4J Java APIs, as detailed in chapter 3. Figure 4.3 is an image exported through the Neo4J Web Visualizer and shows an excerpt of the SN for the NCIT Ontology. It represents a local view of the entire SN (limited to few nodes) from which it is possible to focus on the paths (and the shortest paths) between two terms, for example, *food* and *pudding*. In particular, on of the shortest path in figure is:

```
food ->[holonym]->
    nutrient ->[hyponym]
       aliment - [hyponym]->
          course - [hyponym] -> pudding.
```

where in square brackets are the semantic relations between synsets and nouns refer to the synset representative. Figure 4.5 shows the broader Semantic Network arborescences for the NCIT Ontology. The figure has been obtained through the graph visualization tool Cytoscape, as described in chapter 3.

According to the extended linguistic analysis (see Figure 4.4), models with a high semantic coverage w.r.t. the target ontology are 1), 2) and 5), but the highest ratio between GG and the number of terms in the corresponding lexical chain is for model 1), 2) and 6) (models 3) and 4) are excluded because of their

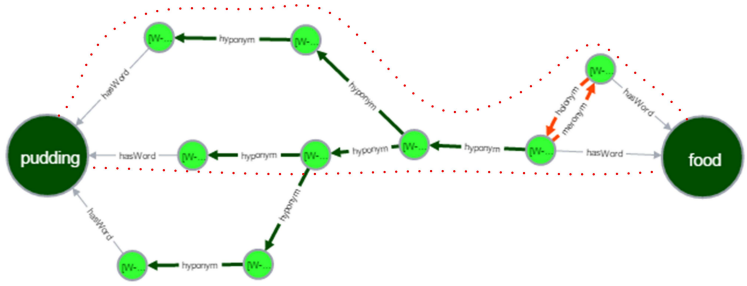


Figure 4.3: An excerpt of the Semantic Network created from the NCIT and Target Ontology lexical chains

low number of terms, while model 8) since it is a proprietary model and so not available), thus, these models will be passed to the Integrator.

4.7 The actual integration of local ontologies

The outcome of the Integrator component described in section 2.4 is an integrated ontology (also refereed as output ontology), which integrate in a global view the concepts coming from the selected input ontologies. Since the extended linguistic analysis has suggested three ontologies as the ones having the highest coverage w.r.t. the target ontology, this phase will focus only on the selected three models, discarding the others for the reasons explained in the previous section. According to the integration methodology described in 2.4, all input

Model	SG	SSG	GG	No.
NCIT	355.37	44.91	400.28	207
AGROVC	383.51	111.17	494.68	898
BBC	39.92	12.79	52.71	58
LIRMM	21.24	1.83	23.08	7
PRODUCT	275.68	216.83	492.51	1039
Eurocode2	52.51	12.53	65.04	167
WAND	8.57	6	14.57	38
ISO	80.93	3.92	84.85	58
Foodon	9.82	46.29	56.11	686

Table 4.7: Input models Semantic/ Syntactic-Semantic grade

concepts classified as *equivalent* w.r.t. a target concept are added as classes to the output ontology and are declared as equivalent w.r.t. the target class. This way they are inferred as equivalent to each other through the transitivity of the equivalence rule. Moreover, taking into account the criteria defined in section 2.1.5, in particular: *Avoiding unnecessary axioms* and *infer sub-class properties rather than explicitly assert it whenever possible*, not all alignment classified as *hyponyms* will be converted in `subclassOf` axioms, but only those which are strictly related, i.e., the direct hyponyms. In Table 4.8 are shown some of the asserted axioms, explicitly created by the Integrator (on the left) and some of the entailments obtained by the transitive reasoner applied to the output ontology.

To be more precise, the integration strategy consists in creating a class for each terms classified as equivalent or hyponym w.r.t. a term in the target ontology. Note that the target concepts act as hub concepts, i.e., they represent the shared objects in the equivalent statements created for the input concepts. This way, using transitive inference rules, it is possible to entail equivalent and subClassOf axioms for the input classes. Figure 4.6 tries to explain this more clearly. The solid arrows in the figure represent the asserted equivalent or subClassOf properties, while the dashed ones the inferred equivalent or subClassOf. Thus, asserting (as premise) that `ncicb:Food`, `agrovc:Food` and `eurocode2:Food` are equivalent to `target:Food`, it is inferred that they are also equivalent each with the others; moreover, asserting, for example, that `ncicb:Meat rdfs:subClassOf ncicb:Food`, `agrovc:Meat rdfs:subClassOf agrovc:Food` and `eurocode2:Meat rdfs:subClassOf eurocode2:Food`, it is possible to infer that `target:Meat` is subClassOf `target:Food`. Additionally,

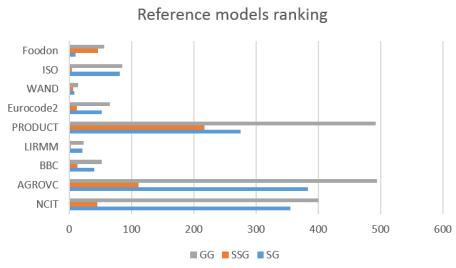


Figure 4.4: Reference models ranking

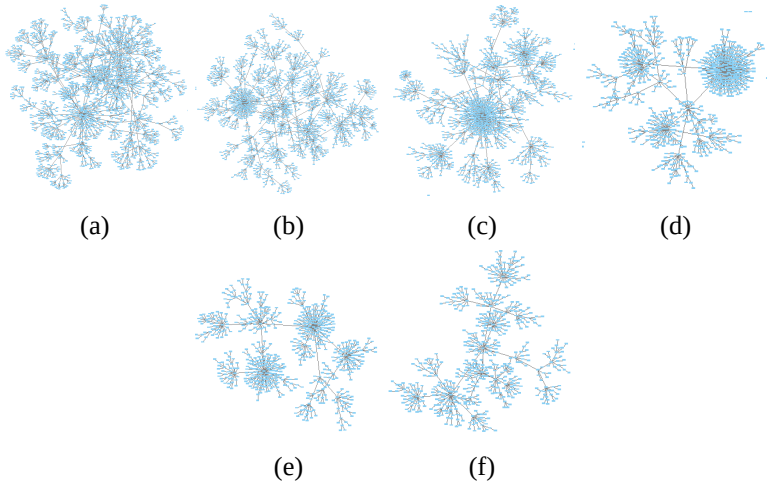


Figure 4.5: The broader Semantic Network arborescences for the NCIT Ontology

every asserted subclasses of one of the *Meat* concept become subclasses of the others plus the target. According to the procedure described in 2.4, only direct hyponym relations are converted in subClassOf properties, thus minimizing the asserted axioms in the output ontology and leveraging the inference capabilities of transitive reasoner in order to retrieve all other entailments. Figure 4.7 shows a large-scale representation of the resulting integration ontology obtained within Ontorion Fluent Editor [3] by Cognitum.

[3] http://www.cognitum.eu/semantics/FluentEditor/

Asserted	Inferred
ncicb:Food ≡ target:Food agrovc:Food ≡ target:Food eurocode2:Food ≡ target:Food	ncicb:Food ≡ ncicb:Food ncicb:Food ≡ agrovc:Food ncicb:Food ≡ eurocode2:Food agrovc:Food ≡ agrovc:Food agrovc:Food ≡ ncicb:Food agrovc:Food ≡ eurocode2:Food eurocode2:Food ≡ eurocode2:Food eurocode2:Food ≡ agrovc:Food eurocode2:Food ≡ ncicb:Food ...
agrovc:Meat ≡ target:Meat agrovc:Meat ⊏ target:Food eurocode2:Meat ≡ target:Meat eurocode2:Meat ⊏ target:Food ncicb:Meat ≡ target:Meat ncicb:Meat ⊏ target:Food ...	target:Meat ⊏ target:Food agrovc:Meat ≡ ncicb:Meat eurocode2:Meat ≡ agrovc:Meat ...
ncicb:Beef ⊏ target:Meat agrovc:Marrow ≡ target:Meat ...	ncicb:Beef ⊏ agrovc:Marrow ...

Table 4.8: Asserted and inferred axioms in the output ontology

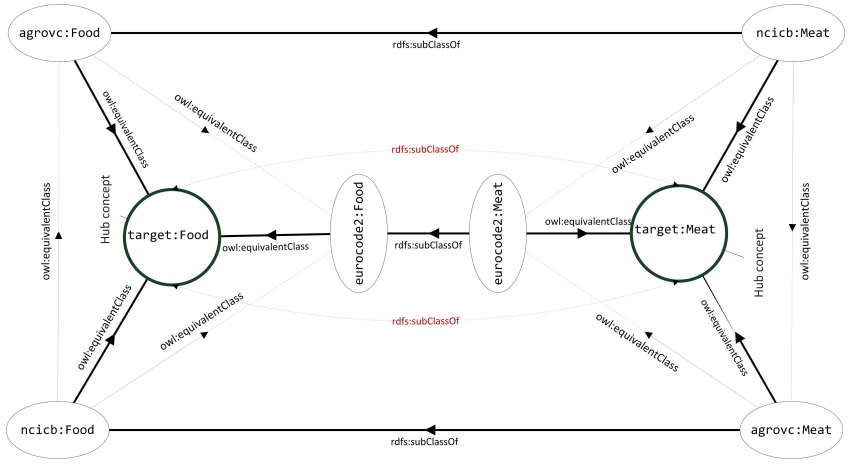


Figure 4.6: Class diagram excerpt for matching and alignment phase objects

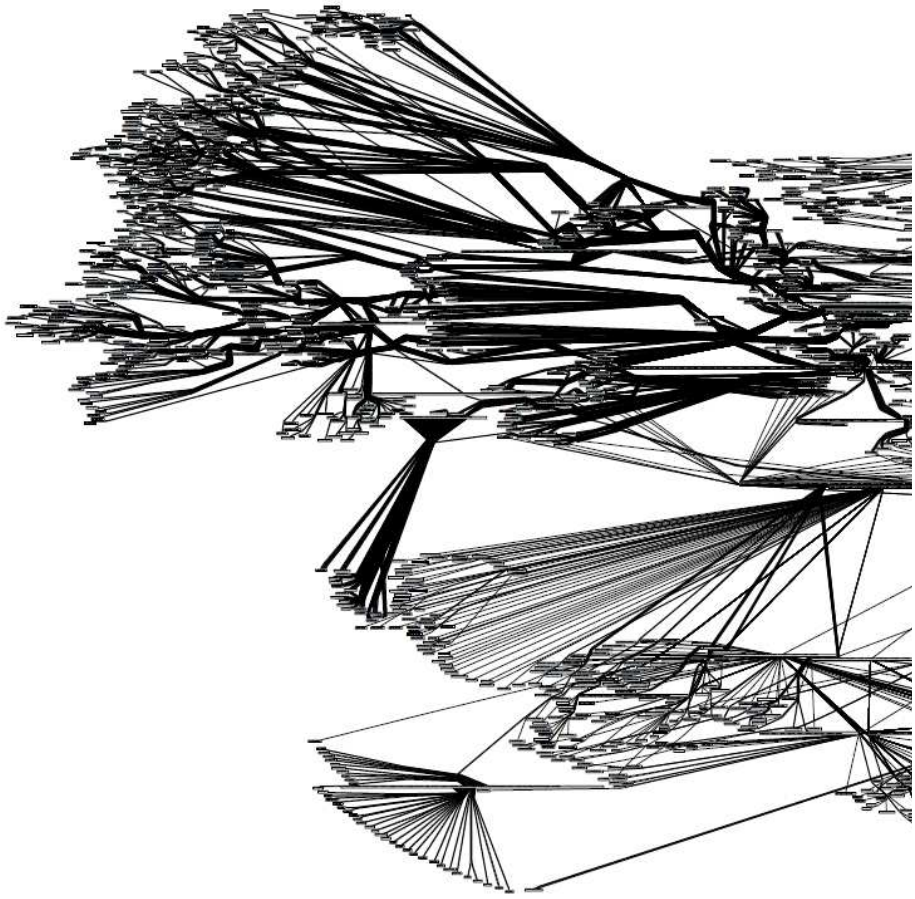


Figure 4.7: A large-scale view of the Integration ontology obtained as a result of the proposed framework

NCIT	ARGOVC	EUROCODE
ncicb:Food	agrovc:Foods	eurocode2:MilkAndMilk...
->ncicb:Cottonseed_Meal	->agrovc:IrradiatedFoods	->euro2:Cream
->ncicb:Dairy_Product	->agrovc:Seafoods	->euro2:OtherFermentedMilk...
->ncicb:Goat_Milk	-->agrovc:Squids	->euro2:Milk
->ncicb:Milk	-->agrovc:SeaCucumbers	->euro2:Cheese
->ncicb:Buttermilk	-->agrovc:Cuttlefish	->euro2:Whey
->ncicb:Butter	-->agrovc:Octopuses	->euro2:Kefir
->ncicb:Honey	->agrovc:PreparedFoods	->euro2:CheeseSubstitutes
->ncicb:Sunflower_Honey	-->agrovc:InfantFoods	->euro2:Yogurt
->ncicb:Rape_Honey	-->agrovc:InstantFoods	->euro2:IceCream
->ncicb:Dandelion_Honey	--->agrovc:InstantCoffee	euro2:MiscellaneousFoods
->ncicb:Manuka_Honey	-->agrovc:Soyfoods	->euro2:BakingGoodsAnd...
->ncicb:Food_Oil	--->agrovc:SoyMilk	->euro2:Sauces
->ncicb:Corn_Oil	--->agrovc:Tempeh	->euro2:Mayonnaise
->ncicb:Sesame_Oil	--->agrovc:SoySauce	->euro2:Soups
->ncicb:Pastry	-->agrovc:Icecream	->euro2:NonDairyCoffee...
->ncicb:Infant_Formula	->agrovc:DieteticFoods	->euro2:SpicesAndHerbs
->ncicb:Egg_Yolk	-->agrovc:LowFatFoods	->euro2:SeasoningAndExtracts
->ncicb:Sea_Salt	-->agrovc:LowCalorieFoods	->euro2:SavourySnacks
->ncicb:Chocolate	->agrovc:Beverages	euro2:FatsAndOils
->ncicb:Wheat_Gluten	-->agrovc:VegetableJuices	->euro2:CompoundFatsOils
->ncicb:Drink	-->agrovc:CoffeeSubstitutes	->euro2:MarineOils
->ncicb:Egg_White	-->agrovc:CocoaBeverages	->euro2:Margarine
->ncicb:Liver_Food	-->agrovc:HerbalTeas	->euro2:VegetableFatsOils
->ncicb:Maple_Syrup	-->agrovc:AlcoholicBeverages	->euro2:FatSpread
->ncicb:Salt_Substitute	--->agrovc:Ciders	->euro2:Butter
->ncicb:Fruit_and_Veg...	--->agrovc:Wines	->euro2:AnimalFats
->ncicb:Squash	--->agrovc:Beers	euro2:VegetablesAnd...
->ncicb:Pumpkin	--->agrovc:Perry	->euro2:BulbVegetables
->ncicb:Pimento	-->agrovc:Tea	->euro2:Brassicas
->ncicb:Nonstarchy_Veg...	--->agrovc:CarbonatedBeverages	->euro2:LeafVegetables
->ncicb:Alfalfa_Sprout	->agrovc:Salads	->euro2:Tubers
->ncicb:Green_Vegetable	->agrovc:ProcessedFoods	->euro2:VegetableMixtures
->ncicb:Green_Leafy_Veg...	-->agrovc:Desserts	->euro2:FruitingVegetables
->ncicb:Onion	-->agrovc:IntermediateMois...	euro2:FishAndFishProducts
->ncicb:Rhubarb	-->agrovc:ReconstitutedFoods	->euro2:PreservedFish
->ncicb:Table_Salt	--->agrovc:ReconstitutedMilk	->euro2:Crustaceans
->ncicb:Cumin	->agrovc:GeneticallyModifiedFoods	->euro2:Perciformes
->ncicb:Sugar	->agrovc:CookingOils	->euro2:CannedFish
->ncicb:Brown_Sugar	-->agrovc:Shortening	->euro2:SaltedAnd...
->ncicb:Raw_Sugar	->agrovc:StreetFoods	->euro2:Gadiformes
->ncicb:Maize_Invert_Sugar	->agrovc:FermentedFoods	->euro2:Frogs
->ncicb:Oryza	-->agrovc:Garri	->euro2:Clupeiformes
->ncicb:Raisin	-->agrovc:Gari	->euro2:Reptiles
->ncicb:Mycoprotein	->agrovc:FastFood	->euro2:MiscellaneousMarine...
->ncicb:Caviar	->agrovc:ValueAddedProduct	->euro2:Pleuronectiformes
->ncicb:Popcorn	-->agrovc:RiceValueAdded...	->euro2:RestructuredFish...
->ncicb:Corn_Syrup	--->agrovc:ColdRice	->euro2:FishProducts
->ncicb:Beverage	->agrovc:SnackFoods	->euro2:SmokedFish
->ncicb:Alcoholic_Beverage	->agrovc:BakeryProducts	euro2:FruitAndFruitProducts
->ncicb:Sake	-->agrovc:PuffPaste	->euro2:CitrusFruit
->ncicb:Wine_Cooler	-->agrovc:Biscuits	->euro2:MiscellaneousFruit
->ncicb:Beer	-->agrovc:Cakes	->euro2:FruitMixtures
->ncicb:Liquor	-->agrovc:Batters	->euro2:Berries
->ncicb:Herbal_Tea	-->agrovc:Doughs	->euro2:StoneFruit
->ncicb:Sage_Tea	-->agrovc:Bread	->euro2:MalaceousFruit
->ncicb:Juice	->agrovc:CookingFats	euro2:GrainsAndGrainProducts
->ncicb:Carrot_Juice	-->agrovc:Shortening	->euro2:SavouryProducts...
->ncicb:Celery_Extract	->agrovc:Soups	->euro2:MixedGrainProducts
->ncicb:Meat	->agrovc:FrozenFoods	->euro2:OatProducts
->ncicb:Lamb	->agrovc:Confectionery	->euro2:WheatFlours
->ncicb:Poultry	-->agrovc:ChewingGum	->euro2:MaizeProducts
->ncicb:Turkey_Poultry	-->agrovc:Cakes	->euro2:WheatBreads
->ncicb:Duck	-->agrovc:Chocolate	->euro2:MilletProducts

Table 4.9: NCIT (1), AGROVC (2) and Eurocode2 (6) adapted ontology fragments

Chapter 5

Experimental results

This chapter describes how the proposed methodology has been evaluated and how the experimental results have been obtained from the case-study described in the previous chapter. The first section introduces the evaluation architecture, detailing the components involved in this task and recalling the matching strategy described in chapter 2. Later on, a section is dedicated to the definition of the Recall and Precision measures (both relaxed and semantically-grounded) used to evaluate the performance of the Matcher and the Aligner components. Another section is dedicated to motivate the involvement of users in the evaluation strategy, in order to obtain a *reference alignment* (here considered as a *ground truth*) for training the classifier function. In this section, a GUI helping the user in making a ground alignment is also described. Finally, the last section shows the performance of the methodology considered in its entirety, i.e., how effective and efficient it is for a knowledge reuse perspective.

5.1 The aligner training and evaluation

As stated in the previous chapters, the objective of the Aligner is to produce an effective alignment of the term pairs coming from the input reference models and the target (once converted in lexical chains), based on different measures provided by the Matcher module. It actually is a multi-class classifier that uses a decision tree to put each alignment in one of five different classes: *equivalent*, *hypernym*, *hyponym*, *related* and *disjoint*. Figure 5.10 shows the blocks involved in the alignment process, i.e., the already mentioned Matcher and Aligner and a

small block, namely, the *Evaluation Block*. This latter is in charge of applying a thresholding strategy in order to find the optimum (in terms of precision and recall) of the classifier function over the training set. As described in chapter 2, the classifier uses the Wu & Palmer similarity (the actual version and an extended version for multi-word terms), calculated by the matcher, in order to score an alignment and, based on a thresholoding strategy, to categorize the alignment according to three possible cases: hyponym-hypernym (considered as one classification class), related and disjointed. Hereafter, the equivalent class will not be included in the discussion since it is based on other similarity measures (synonym and co-synonym grades) which present good performance (discussed later) and do not need to be further investigated. Thus, the discussion will focus on the three classes mentioned above. The thresholding strategy is one of the common approaches used in automated text categorization, i.e., the SCut methodology [114]. This scores a validation set of documents for each category and tunes the threshold over the local pool of score until the optimal performance of the classifier is obtained for that category, then fix the per-category thresholds when applying the classifier to new document in the test set. In this case are used alignments rather than documents and the Wu & Palmer similarity measure (or its extended version) as score.

The approach uses two thresholds t_1 to discriminate between hyponym-hypernym class (abbreviated as hypo-hyper class) and related, and t_2 to dis-

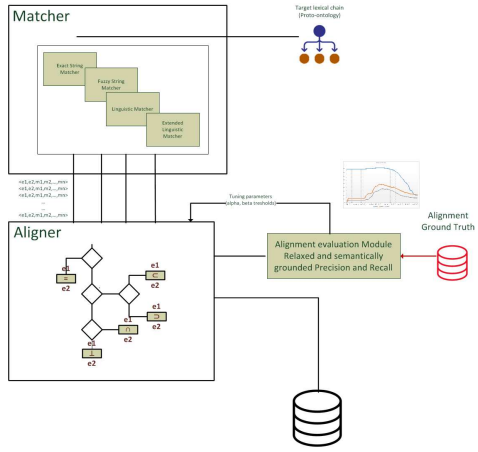


Figure 5.1: Aligner evaluation block

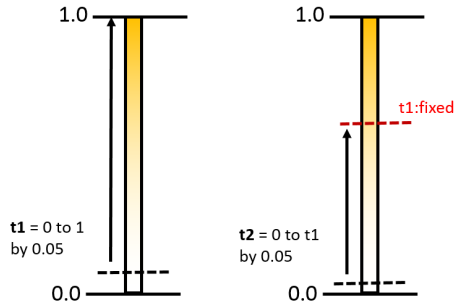


Figure 5.2: Determining thresholds values

criminate between related and disjointed class. So, it turns that, for the sake of this analysis, the aligner is a three-classes classifier that uses a decision tree method, based on an SCut thresholding strategy with two thresholds. Searching for the optimum value of t_1 and t_2 in terms of Precision and Recall for this polycotomous classifier means decomposing it into more binary classifiers and optimizing each of them according to the thresholding strategy described previously. There are different decomposition (also known as binarization) techniques in the literature, but the one used in this work is the *ordered class binarization* [115], which transforms a c-class problem into c-1 binary problems. These are constructed by using the examples of class i ($i = 1...c-1$) as the positive examples and the examples of classes j>i as the negative examples. In this case, starting from the three-class classifier, the ordered binarization leads to two binary classifiers: the first classifies hypo-hyper alignment (positive examples) versus not hypo-hyper alignment (negative examples, i.e., related and disjointed alignments), while the second classifies the related alignment (positive examples) versus disjointed alignment (negative). This technique is particularly suitable for the classifier described here, because it involves an ordering relation between the classes and is used when a decision tree-based function is adopted to discriminate between different categories based on a cut threshold. Actually, an implicit ordering exists between the classes since hypo-hyper, for example, has on average a similarity measure greater than that of related, which, in turn, has a similarity measure greater than that of disjointed. This binarization technique imposes an order on the induced classifiers too, which has to be adhered to at classification time. This means that the classifier learned for discriminating class

1 from classes $2...c$ has to be called first. If this classifier classifies the example as belonging to class 1, no other classifier is called; if not, the example is passed on to the next classifier. Taking into account all the above, the first binary classifier, hypo-hyper vs related and disjoint, has been evaluated by thresholding the t_1 parameter between 0.0 and 1.0 by 0.05. Figure 5.2 shows the thresholding strategy, while Figure 5.3 shows the classical Precision and Recall curves for the first binary classifier (in blue) and the second binary classifier (in brown), in correspondence of each value of t_1, together with the F1-score isometric lines (from 0.1 to 1.0). The F1-score is the harmonic mean of precision and recall. The plot in figure shows that the optimal choice for t_1 in order to maximize the F1-score is 0.6 (F1-score = 0.83). Taking this values as a maximum value for t_1 and evaluating the second binary classifier (related vs disjointed) by thresholding t_2 between 0.0 and 0.6, the Precision and Recall curves shown in the brown plot are obtained. This plot shows three suboptimal points near the 0.7 F1-isometric. The t_2 has been fixed in correspondence of the one that maximizes the precision (i.e., t_2 = 0 at F1 = 0.68 and P = 0.62). Figure 5.3 clearly shows that the first binary classifier outperforms the second one, presenting an F1-score 15 percentage points higher than the second. It is possible to measure the average of the two in terms of micro- or macro-average precision, recall and F1-score. When scores are micro-averaged, the binary decisions are collected in a joint pool and then the recall, precision and F1 values are computed from that pool. When the scores were macro-averaged, the recall, precision and F1 values for individual categories are computed first and then averaged over categories. This leads to the following formula for micro (and macro) version of precision, recall and F1-score:

$$P_{micro} = \frac{tp1 + tp2}{(tp1 + tp2) + (fp1 + fp2)} \tag{5.1}$$

$$P_{macro} = \frac{P_1 + P_2}{2} \tag{5.2}$$

$$R_{micro} = \frac{tp1 + tp2}{(tp1 + tp2) + (fn1 + fn2)} \tag{5.3}$$

$$R_{macro} = \frac{R_1 + R_2}{2} \tag{5.4}$$

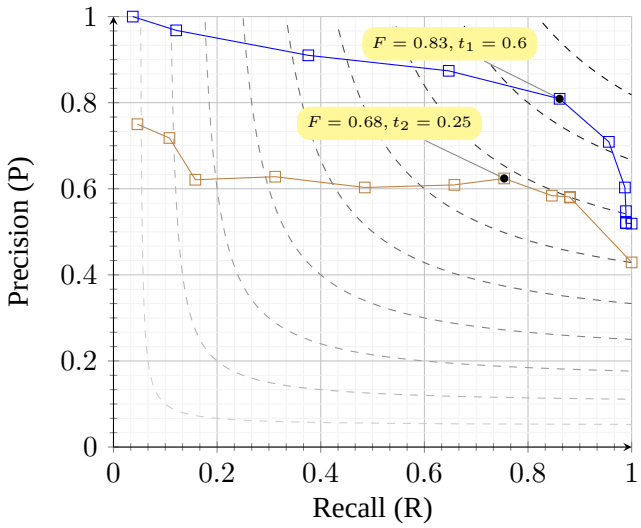


Figure 5.3: Precision and Recall curves of the two binary classifiers

$$F1_{micro} = \frac{2 * P_{micro} R_{micro}}{P_{micro} + R_{micro}} \tag{5.5}$$

$$F1_{macro} = \frac{F1_1 + F1_2}{2} \tag{5.6}$$

where tp_i, fp_i, and fn_i are the true positives, false positives and false negative for the i-th class, respectively. Table 5.1 shows the precision, recall and F1 measures, both micro and macro-averaged, for the multi-class classifier in correspondence of the suboptimal points retrieved from figure 5.3. Not surprisingly, the micro-averaged version of precision, recall and F1-score are greater than the macro-averaged ones. This is due to the first binary classifier performance that dominates the second one.

In addition to precision and recall curves, another effective graphical plot in order to evaluate the performance of any binary classifier if the ROC (Receiver Operating Characteristic) curve [116]. This curve is created by plotting the true positive rate (tpr) against the false positive rate (fpr) at various threshold settings. Tpr is defined as $\frac{tp}{tp+fn}$ while fpr as $\frac{fp}{fp+tn}$, where t_n are the true negatives. The more the curve is above the diagonal, the more accurate is the classifier (a curve

	Precision	Recall	F1-score
micro	0.756	0.833	0.793
macro	0.716	0.807	0.760

Table 5.1: Averaged precision, recall and F1-measures for the whole classifier

close to the diagonal represents a random classifier while a curve under the diagonal is a *perverse* or inverted classifier). The ROC curve is a valid method to compare classification algorithms. Figure 5.4 compares three versions of the first binary classification algorithm (hypo-hyper vs related-disjoint), in blue plot, and three versions of the second binary algorithm (related vs disjoint), in brown plot. Each version differs from the others for the choice of the particular extended Wu & Palmer measure adopted, namely, the *average* (extwup_avg), the *maximum* (extwup_max)) and the *minimum* version (extwup_min). The plot shows that for the first classifier the best choice can be the average or the maximum versions (both present good performance but the first tends to be more precise at the expense of the recall), while, the best choice for the second classifier is the minimum version (the only one that is always above the diagonal).

At this stage of the evaluation, the equivalent and hypernym class are recovered in order to evaluate the whole 5-classes classifier with the values for t_1 and t_2 found. A methodology to evaluate the performance of a polycotomous classifier without binarization strategy is by using the confusion matrix (shown in table 5.2). The diagonal elements represent the number of alignments for which the predicted class is equal to the true class, while off-diagonal elements are those that are mislabelled by the classifier. The higher the diagonal values the better are the performance of the classifier (it means many correct predictions). By dividing each element in the diagonal by the sum of the elements in the respective row and column it can be obtained the recall and the precision respectively for each class. The confusion matrix shows also a *virtual* class, namely, *unknown* that collects all alignments the classifier was unable to predict. Normalized values for precision and recall for all confusion matrix elements are used to visualize color maps (rgb or gray-scale based), which immediately picture the accuracy of the classification algorithm. Figure 5.5 (a) and (b) show the color maps for the proposed classification algorithm. The more the gray is dense in the diagonal elements, the more accurate is the classifier. A rapid look shows that while

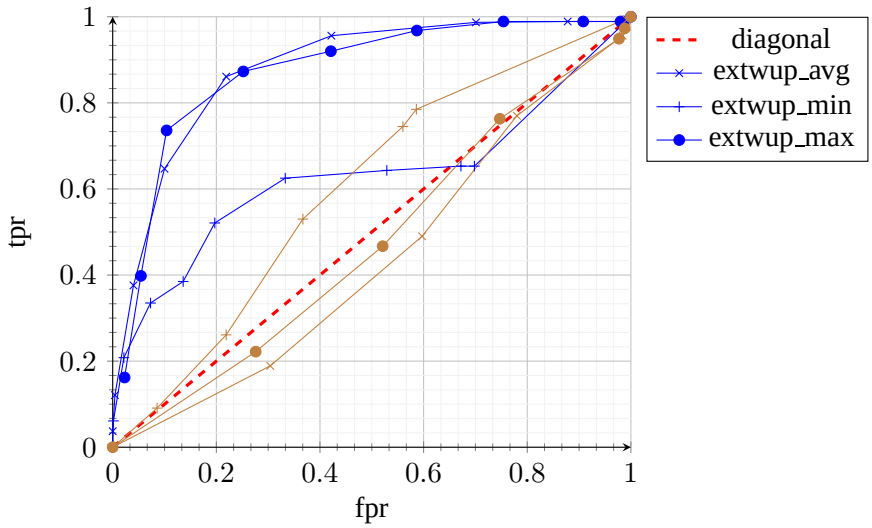
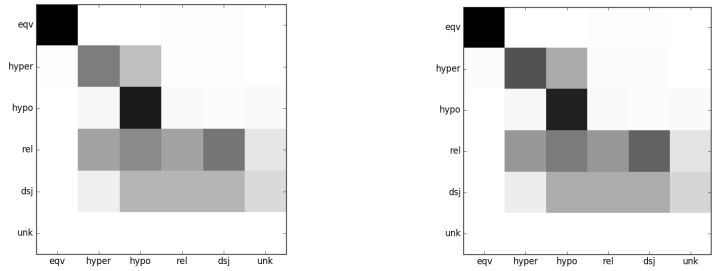


Figure 5.4: ROC curves of the two binary classifiers

equivalent, hypernym and hyponym are well balanced class, that is, they are more dense in the diagonal, meaning a great precision and recall, the other two class are not. In numbers, the precision (recall) measures retrieved from the confusion matrix are 0,991(0,981) 0,361(0,608) 0,826(0,777) 0,494(0,485) 0,609(0,508) for the equivalent, hypernym, hyponym, related, disjoint class respectively. The case for the hypernym, related and disjoint class can be ameliorated (or relaxed) using relaxed precision and recall strategy as described in the next section.

Table 5.2: Confusion Matrix

	eqv	hyper	hypo	rel	dsj	unk
eqv	105.0	0.0	0.0	1	1.0	0.0
hyper	1.0	48	22	3	5.0	0.0
hypo	0.0	23	537	51	64	16.0
rel	0.0	56	51	175	54	25
dsj	0.0	6	40	124	193	17
unk	0.0	0.0	0.0	0.0	0.0	0.0



(a) Precision Confusion matrix (b) Recall confusion matrix

Figure 5.5: Confusion matrix visualization

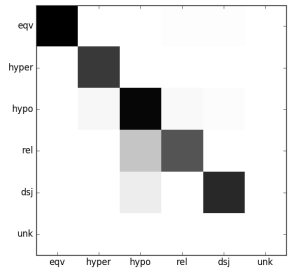
5.2 Relaxed Precision and Recall

Precision and recall together with the F-score are commonplace measures in information retrieval and they have also been adapted for ontology alignment evaluation [117], but they have the drawback that whatever correspondence has not been found is definitely not considered. As a result, they do not discriminate between a bad and a better alignment. For example, in the context of this work, an alignment predicted as *related* is better than an alignment predicted as *disjoint* if the true alignment is *hyponym* or *hypernym*. So, following the approach in [118], instead of comparing alignments set-theoretically, it can be measured the proximity of correspondence sets rather than the strict size of their overlap, in other words, instead of taking the cardinal of the intersection of the two sets ($|\ R \cap A|$), the generalizations of precision and recall measure their proximity ω(A,R), where ω is an overlap function between alignments based on a proximity function (σ) between two correspondences. Since in this work the users have to check and correct the final integration ontology that is based on the predicted alignments, the quality of alignment algorithms can be measured through the effort required for transforming the obtained ontology *fragment* (the one depending on the predicted alignment) into the corrected one. This effort can be measured as an edit distance [96], which defines a number of operations by which an object can be corrected (here the operations on correspondences authorized) and assigns a cost to each of these operations (here the effort required to identify and repair some mistake). The result can always be normalized in

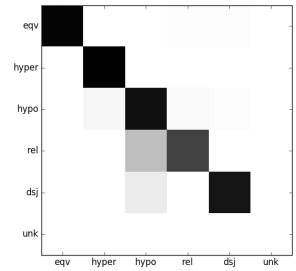
function of the size of the largest object. Such a distance can be turned into a proximity by taking its complement with regard to 1. In order to determining the edit distance here, it is necessary to make some premise: in this version of the framework only equivalent alignments and hyponym alignments will be used in the integrated ontology. The hypernym alignment will not be used because the hierarchy of classes is constructed through the hyponym relation rather than the hypernym one. This is in line with tha fact that usually classes are organized in a taxonomy in which they have less direct super than sub-classes. So in general, it is easier to correct a class to (one of) its superclass than to one of its sub-classes. Here no correction at all are made to a class against its superclass. The related alignment requires *a priori* the user involvement because this relation does not provide sufficient information to predict strong linguistic or semantic relation between terms. The disjoint alignments (the most part of alignment) do not translate in a explicit axiom in the integrated ontology. Given the above, the only effort required by the user for transforming the predicted ontology *fragment* in the correct one consists in adding all alignments predicted as related or disjoint (and the unknown) but in fact hyponym (false negative hyponyms), and in deleting all alignments predicted as hyponyms but in fact related or disjointed (false positive hyponyms). The cost function assigns a greater effort for the adding operations (1.0) and a lesser value for the deleting operation (0.25 for true related and 0.5 for true disjoint). These values are justified by the fact that deleting an axiom like *subClassOf* or *equivalentOf* only means clicking a button in many ontology editors (e.g., NeoN Toolkit, see Figure 5.7) while adding a subClassOf axiom means create a class (providing a name for the class) and assert it as subclass of another class (see Figure 5.7). Note that, in some case the costs will be added if more than one modification is needed to resolve the mismatch. Table 5.3 shows the edit distances for all class-to-class pairs while their proximity measures are the complement to 1. By relaxing the Precision and Recall according the edit-distance and the proximity measures obtained from table 5.3 leads to an improved *relaxed* confusion matrix whose accuracy and recall value are: 0,991(0,981) 0,827(0,709) 0,950(0,912) 0,853(0,789) 0,795(0,850) for the equivalent, hypernym, hyponym, related, disjoint class respectively. The new color maps are shown in Figure 5.6. It is clear that relaxing the precision and recall measures, by taking into account only the effort required to correct the integration ontology fragments in case of misleading, greatly improves the performance of the aligner classifier.

Table 5.3: Edit Distance costs for misleading alignments

	eqv	hyper	hypo	rel	dsj	unk
eqv	0.0	1.0	1.25	1.0	1.0	1.0
hyper	0.25	0.0	0.25	0.0	0.0	0.0
hypo	1.25	1.0	0.0	1.0	1.0	1.0
rel	0.25	0.0	0.25	0.0	0.0	0.0
dsj	0.25	0.0	0.5	0.0	0.0	0.0
unk	0.0	0.0	0.0	0.0	0.0	0.0



(a) Precision color map



(b) Recall color map

Figure 5.6: Relaxed Confusion matrix

5.3 Semantically grounded alignment

Since the Semantic Aligner adds logical consequences (α-consequences) to the set of alignment obtained at a linguistic and syntactic level by the Aligner component (see section 2.3.1), the evaluation of the whole aligner performance must take into account these newly added alignments, i .e., it must consider the nature of the objects being aligned, i.e., the terms which represent labels attached to entities (class, individuals or properties) contained in an ontology: a formal (first-order logics based) representation of concepts and relations among concepts. In other words, the alignment Precision and Recall measures is grounded to the semantics of the ontological models where entities involved in the alignment come from. In consequence, those correspondences that are consequences of the evaluated alignments have to be considered as recalled and those that are consequence of the reference alignments as correct. Following the guidelines

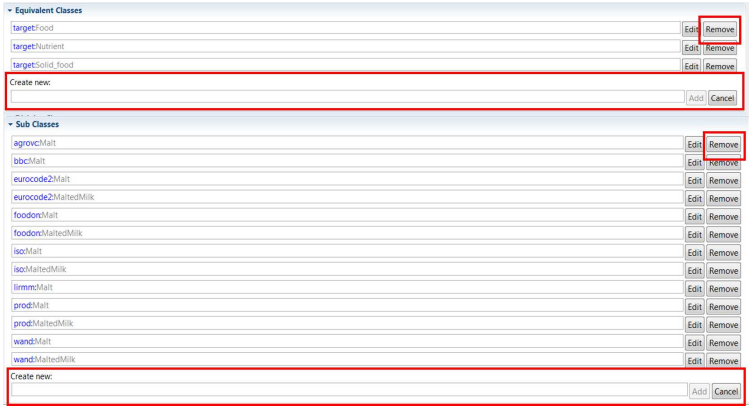
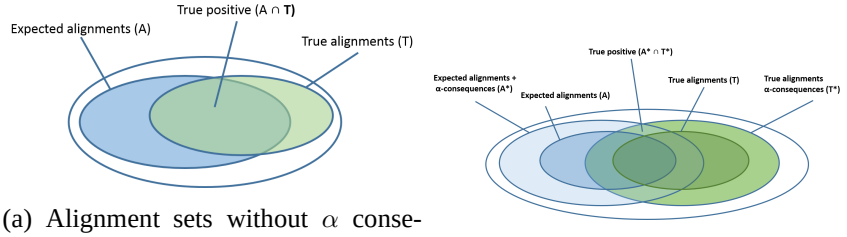


Figure 5.7: NeoN toolkit screenshots



(a) Alignment sets without α consequences

(b) Alignment set with α consequences

Figure 5.8: Alignment sets. Adopted from [1]

introduced in [118], the natural semantic extension of precision and recall measures consists of using the set of α-consequences (or deductive closure) instead of the intersection between the expected and true (reference) alignment. In this case, the true positive becomes the correspondences that are consequences of both alignments and the usual definitions of true and false positives and negatives are only extended to alignment consequences. Figure 5.8 tries to explain what meant before. While the classical notions of Precision and Recall are as follows:

$$P = \frac{|A| \cap |T|}{|A|} \tag{5.7}$$

$$R = \frac{|T| \cap |A|}{|T|} \tag{5.8}$$

the semantically grounded versions become:

$$P^* = \frac{|A^* \cap T^*|}{|A^*|} \tag{5.9}$$

$$R^* = \frac{|T^* \cap A^*|}{|T^*|} \tag{5.10}$$

where A^* is the deductive closure for the expected alignments and T^* is the deductive closure for the ground truth (or reference) alignments.

In this work only, just taxonomies are considered as input ontologies and it is used a transitive reasoner for additional entailments. This means that the deductive closure of the expected (A*) and true alignments (T*) become the *transitive closure* of the *rdfs:subClassOf* property and the closure of the *owl:equivalentClass* property over the set A and T. This extension of precision and recall has two drawbacks [1]: (1) both numerator and divisor could be infinite, yielding an undefined result, and (2) do not guarantee to provide better results than precision and recall in general. In order to deal with the problems raised by the infinite character of the set of α-consequences, a natural way would be to compare the deductive reductions instead of the deductive closures. Unfortunately, the deductive reduction is usually not unique. A solution can be using the deductive closure bounded by a finite set so that the result is finite. For example it is possible to use the set A to bound the ground truth alignment extension and the ground truth to bound the alignment extension, like in the following formula:

$$P^{sem} = \frac{|A \cap T^*|}{|A|} \tag{5.11}$$

$$R^{sem} = \frac{|A^* \cap T|}{|T|} \tag{5.12}$$

By using this revised formula and the α-consequences according to the procedure described above and in the previous sections, the measures of precision and recall are significantly improved mainly due to the hyponyms class recall increase.

5.4 User involvement in ground alignment creation

According to [87] user validation in ontology alignment it is not only essential in any automated system due to the complexity and intricacy of the ontology alignment process, for the detection and the removal of erroneous mappings, or the addition of alternative mappings; but, if user validation is done during the alignment process, it enables the adjustment of system settings, the selection of the most suitable alignment algorithms, and the incorporation of user knowledge in them [59]. Welcoming this suggestion, the proposed framework involves the user in three stages: the creation of the proto-ontology, the refinement of the integration ontology according to the proto-ontology previously created and the creation of the alignment ground truth. Focusing on the third stage, the strategy used in this work consists in creating a ground alignment by adopting a Graphical User Interface (GUI). Often, GUIs are indispensable part of every interactive system, as the visual system is humans' most powerful perception channel. Alignment validation is a cognitively demanding task that involves a high memory load – ontologies are complex knowledge-bases, and validating each mapping requires considering the structure and constraints of two ontologies while also keeping in mind other mappings and their logical consequences – and thus is all but impossible without visual support. Given the above, a useful GUI has been adopted in order to help users to create the ground truth.

Figure 5.9 shows a GUI consisting in two panels: the left and the right panel. The first contains the list of the terms extracted from one input ontology while the second the list of terms extracted from the target ontology. Both lists are alphabetically ordered and allow multiple selection. This way te user can select one or more items from the left list and link them to one or more items from the right list by clicking one of the buttons in the toolbar at the top. Apart from the load button, which load the terms from files in the lists, the toolbar contains five buttons associated to the five classes of alignment: eqv, hypo, hyper, related and disjointed. When one of these five button is pressed a list of selected alignments in the form (term1;term2;relation) is printed out to the console and can be copied and pasted in a csv file containing all the ground truth alignments.

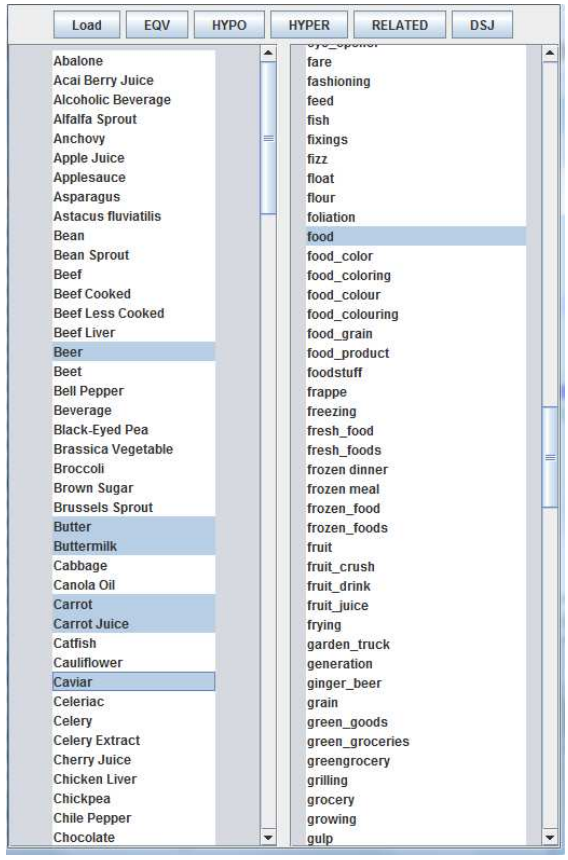
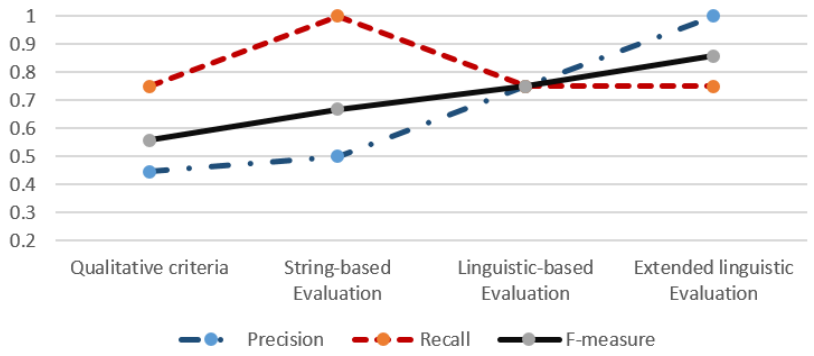


Figure 5.9: GUI for alignment ground truth generation

5.5 Evaluation of the whole methodology from a knowledge reuse perspective

Since the methodology described in this dissertation consists of different phases, involving matching and integration tasks, and every phase produces intermediate results, which would require specific comparisons with other similar matching systems or approaches, the discussion here concentrates on a global level, thus evaluating how efficiently and effectively, in terms of *Precision* and *Recall*, the candidate reference models have been selected to be reused in the integration

Figure 5.10: Reference models selection results



phase of the workflow, considering the methodology as a whole.

For the sake of this global analysis, also in this case, the standard definitions of precision and recall are used by introducing two sets of reference models, namely, the relevant reference models set (described later) and the retrieved reference models set, i.e., those resulting by applying the proposed methodology. The relevant models have been individuated manually, from the list in Table 4.1, by averaging the score assigned them by a group of experts. This task has led us to select model 1), 2), 6) and 8) as relevant. They represent a kind of ground truth for the analysis described here. Figure 5.10 shows an overview of the experimental results by plotting the maximum values of precision, recall and F-measure in four different moments of the whole procedure, namely, at the end of the first phase (reference models retrieval) after applying the qualitative criteria to make a first selection of relevant models, at the end of string-based matching phase, at the end of the linguistic-based matching phase and, finally, at the end of the extended linguistic-based matching phase. The figure shows a common behaviour regarding the precision and recall curves: while the first measure increases, the second one decreases. In this context, this is due to the effect of applying increasingly sophisticated matching methodologies in order to retrieve the candidate reference models, that ameliorate the precision at the expense of the recall. The F-measure continuously increases throughout the matching phases as result of the precision rise, despite of the fluctuating trend of the recall.

Chapter 6

Discussion

This chapter discusses the experimental results obtained in the previous one, both those coming from the matching strategy and those coming from the evaluation of the framework viewed as a whole system.

6.1 Alignment results discussion

As already stated in chapter 5, the confusion matrix reported at the end of the alignment evaluation procedure shows that the classifier presents good performances in terms of precision and recall for the equivalent and hyponym-hypernym cases, less good for the remaining cases (related and disjointed class). All the cases will be discussed in detail, in the following sub-sections, pointing out the different considerations for the false positive and false negative case.

6.1.1 Equivalent terms

The case of equivalent class presents very high values of accuracy and recall. This is due to the stronger methods the classifier adopts to decide for equivalent class, i.e., the *exact string matching* and the maximum *co-synonymy grade*. These methods minimize the risk for false positives, while the few false negatives can be eliminated relaxing the string similarity measures or adopting text pre-processing operation like singularization or stemming in order to recognize terms that refer to the same entity although they present different morphology.

6.1.2 Hyponymy or hypernymy

Also the hyponym and hypernym class presents high values of accuracy and recall. This is due to the choice of the threshold value t_1 used by the classifier to distinguish hyponyms (herafter hypernyms will be omitted because they follow the same fate of hyponyms) from disjointed or related cases. The threshold t_1 is fixed to an optimal value tuned by experiments (0.6). Specifically, the aligner classifier decides for hyponym class by comparing the Wu & Palmer measure (in case of single words) or the extended Wu & Palmer (for multi-words) w.r.t. t_1. Because of the value of t_1 is high, only actual hyponym terms will be cla ssified as such reducing the risk for false positives. Anyway, the false negative case matter too, so it is necessary to find a compromise on the basis of the cost function used to refine the resulting ontology. In particular:

- the case of *false positive hyponym* does not represent a great issue, since it requires the user to correct the integrated ontology by deleting false *subClassOf* axioms automatically created between the ontological classes created from the erroneous alignments. As already discussed, this operation in many ontology editors (e.g., NeOn Toolkit) means just clicking a button to eliminate the corresponding declaration. Thus, the cost in term of modifications is quite low, and so is the edit distance between the correct and the misleading version of the integration ontology.
- the case of *false negative hyponym* represents the biggest drawback since in this case real *subClassOf* relations will be missed and the user is required to adjust the integration ontology by adding the missed axioms. This implies a modification cost higher than just clicking a button because it will necessary to create a class (by inserting its localName and eventually a linguistic label) and declare it as *subClassOf* another existing class.

The number of false negatives can be reduced by taking into consideration the semantically grounded consequences, as discussed in chapter 5, since these exploit the inherent logics contained in the formal model of the input ontologies, thus, being able to improve the accuracy.

6.1.3 Related and disjointed

The remaining part of the classifier classifies the *related* class against the *disjoint* one. In this case, the most worrying situation is when predicted related or

(even worst) disjoint are actual hyponym or equivalent. This latter is extremely unlikely (in the used test only one case is present), while the first is more likely and belongs to the false negative hyponyms case previously discussed. All disjoint and related false positives do not imply any modification in the integration ontology since no explicit axioms are created for related and disjoint alignments.

6.2 Integration ontology considerations

The integration methodology proposed in this dissertation applied on the three selected reference models (as detailed in chapter 4) has resulted in an output ontology containing 236 `quivalentClass` declarations and 943 `subClassOf` declarations. Its inferred version (obtained with the Jena micro rules reasoner) contains in total 2174 `quivalentClass` declaration and 9474 `subClassOf` declarations. The refinement operations required to the users have consisted in deleting a limited number of false `subClassOf` axioms and in adding a limited number of missing `subClassOf` axioms. The number of such modifications is low w.r.t. the global dimension of the integrated ontology. The adding operation in particular have required the greatest effort. Nevertheless, the lesson learnt by the adoption of the proposed methodology to the case study, is that the creation of an integration ontology by the automatic framework results in a quite good quality ontology, which would have required a huge effort by the user if it had been made completely manually. Thus, the significant involvement of expert users needed to refine the resulting ontology is still justified by the big advantage of having a first version of the integrated ontology, but with a discrete level of quality.

Conclusions

This dissertation has proposed a multi-strategy methodology for ontology integration and reuse based on a combination of existing ontology matching techniques. Although the methodology aim at reducing the human intervention in all ontology integration phases, it does not neglect it, neither considers it as en element of weakness; on the contrary, user involvement is considered an essential *tuner* for the whole strategy as it incorporates precious user knowledge in the framework making it more effective. The experimentation within the Food domain has demonstrated that the adoption of such approach with the help of an *ad hoc* software framework can improve and simplify the creation of new ontology models, automatizing as much as possible the ontology creation task and promoting the knowledge reuse by properly identifying those models that belong to a specific interpretation of the domain under study among others. Furthermore, the synergistic use of information visualization techniques and the capabilities of new tools emerged in the landscape of Big Graph Data like Neo4J, together with its declarative graph query language (Cypher), has helped in visualizing the semantic coverage of reference input models, once converted into proper Semantic Networks, allowing a different kind of evaluation, which goes beyond the classical precision and recall measures and linguistic similarity, insofar it leverage new features like pattern-based queries and iconicity.

Appendix A

The list below provides a short description for each selected reference model filtered out from the corpus of retrieved references.

1. **National Cancer Institute Thesaurus** [1] by the American National Institutes of Health (NIH):
   The NCI Thesaurus is a reference terminology and biomedical ontology used in NCI systems. It covers vocabulary for clinical care, translational and basic research, and public information and administrative activities.

2. **AGROVOC Multilingual agricultural thesaurus** [2] by AIMS Advisory Board:
   AGROVOC is a controlled vocabulary covering all areas of interest of the Food and Agriculture Organization (FAO) of the United Nations, including food, nutrition, agriculture, fisheries, forestry, environment etc.

3. **BBC Food Ontology** [3] by BBC:
   The Food Ontology is a simple lightweight ontology for publishing data about recipes, including the foods they are made from and the foods they create as well as the diets, menus, seasons, courses and occasions they may be suitable for. Whilst it originates in a specific BBC use case, the Food Ontology should be applicable to a wide range of recipe data

[1] National Cancer Institute Thesaurus. Available online, https://ncit.nci.nih.gov/ncitbrowser/

[2] AGROVOC Multilingual agricultural thesaurus. Available online, http://aims.fao.org/vest-registry/vocabularies/agrovoc-multilingual-agricultural-thesaurus

[3] BBC Food Ontology. Available online, http://www.bbc.co.uk/ontologies/fo

publishing across the web.

4. **LIRMM Food Ontology** [4] by LIRMM Laboratoire:
   This ontology models the Food domain. It allows to describe ingredients and food products. Some classes are: food:Recipe, food:Food, food:FoodProduct, food:Dish, food:Ingredient, etc.

5. **The Product Types Ontology** [5] by E-Business and Web Science Research Group at Bundeswehr University Munich:
   This ontology contains 300,000 precise definitions for types of product or services that extend the schema.org and GoodRelations standards for e-commerce markup.

6. **Eurocode 2 Food Coding System** [6] by European FLAIR Eurofoods-Enfant Project:
   The Eurocode 2 Food Coding System was originally developed within the European FLAIR Eurofoods-Enfant Project to serve as a standard instrument for nutritional surveys in Europe and to serve the need for food intake comparisons.

7. **WAND Food and Beverage Taxonomy** [7] by WAND Company:
   The WAND Food and Beverage Taxonomy includes 1,278 terms including foods, beverages, ingredients, and additives. This taxonomy includes anything that somebody may consume as food, including some prepared foods. The WAND Foods and Beverages Taxonomy is ideal for restaurants, groceries, and food manufacturers.

[4] LIRMM Food Ontology. Available online, http://data.lirmm.fr/ontologies/food

[5] The Product Types Ontology. Available online, http://www.productontology.org/

[6] Eurocode 2 Food Coding System. Available online, http://www.danfood.info/eurocode/

[7] WAND Food and Beverage Taxonomy . Available online, http://www.wandinc.com/wand-food-and-beverage-taxonomy.aspx

8. **Food technology ISO Standard** [8] by ISO:
   International standard by ISO which provides a terminology for processes in the food industry, including food hygiene and food safety, food products in general, methods of tests and analysis for food ICS (International Classification for Standards) products, materials and articles in contact with foodstuffs and materials and articles in contact with drinking water, plants and equipment for the food industry.

9. **Foodon food Ontology** [9]:
   FOODON is a new ontology built to interoperate with the OBO Library and to represent entities which bear a “food role”. It encompasses materials in natural ecosystems and food webs as well as human- centric categorization and handling of food. The latter will be the initial focus of the ontology, and we aim to develop semantics for food safety, food security, the agricultural and animal husbandry practices linked to food production, culinary, nutritional and chemical ingredients and processes.

[8] International classification for Standards (ISC). Available online, http://www.iso.org/iso/home/store/catalogue_ics.htm

[9] Foodon Ontology Project. Available online, http://foodontology.github.io/foodon/

Bibliography

[1] Jérôme Euzenat. Semantic precision and recall for ontology alignment evaluation. In *IJCAI*, pages 348–353, 2007.

[2] World Wide Web Consortium (W3C). W3c semantic web activity. Technical report, 2011.

[3] Ronald J Brachman, Hector J Levesque, and Raymond Reiter. *Knowledge representation*. MIT press, 1992.

[4] John F Sowa. Knowledge representation: logical, philosophical, and computational foundations. 1999.

[5] Joseph C Giarratano and Gary Riley. *Expert systems*. PWS Publishing Co., 1998.

[6] John F Sowa. *Principles of Semantic Networks: Explorations in the representation of knowledge*. Morgan Kaufmann, 2014.

[7] Stuart Jonathan Russell, Peter Norvig, John F Canny, Jitendra M Malik, and Douglas D Edwards. *Artificial intelligence: a modern approach*, volume 2. Prentice hall Upper Saddle River, 2003.

[8] Bruce G Buchanan, Edward Hance Shortliffe, et al. *Rule-based expert systems*, volume 3. Addison-Wesley Reading, MA, 1984.

[9] Franz Baader. *The description logic handbook: Theory, implementation and applications*. Cambridge university press, 2003.

[10] Ian Horrocks. Ontologies and the semantic web. *Communications of the ACM*, 51(12):58–67, 2008.

[11] Thomas R Gruber. A translation approach to portable ontology specifications. *Knowledge acquisition*, 5(2):199–220, 1993.

[12] N BorstW. Construction of engineering ontologies. *University of Twenty, Enschede, Center for Telematica and Information Technology*, 1997.

[13] Rudi Studer, V Richard Benjamins, and Dieter Fensel. Knowledge engineering: principles and methods. *Data & knowledge engineering*, 25(1-2):161–197, 1998.

[14] James Hendler. Agents and the semantic web. *IEEE Intelligent systems*, 16(2):30–37, 2001.

[15] Antonia Cataldo and Antonio M Rinaldi. An ontological approach to represent knowledge in territorial planning science. *Computers, Environment and Urban Systems*, 34(2):117–132, 2010.

[16] Namyoun Choi, Il-Yeol Song, and Hyoil Han. A survey on ontology mapping. *ACM Sigmod Record*, 35(3):34–41, 2006.

[17] Antonio M Rinaldi. An ontology-driven approach for semantic information retrieval on the web. *ACM Transactions on Internet Technology (TOIT)*, 9(3):10, 2009.

[18] Antonio M Rinaldi. Improving tag clouds with ontologies and semantics. In *2012 23rd International Workshop on Database and Expert Systems Applications*, pages 139–143. IEEE, 2012.

[19] Antonio M Rinaldi. Document summarization using semantic clouds. In *Semantic Computing (ICSC), 2013 IEEE Seventh International Conference on*, pages 100–103. IEEE, 2013.

[20] Tullio Tolio, Marco Sacco, Walter Terkaj, and Marcello Urgo. Virtual factory: An integrated framework for manufacturing systems design and analysis. *Procedia CIRP*, 7:25–30, 2013.

[21] Pavel Shvaiko and Jérôme Euzenat. Ontology matching: state of the art and future challenges. *Knowledge and Data Engineering, IEEE Transactions on*, 25(1):158–176, 2013.

[22] E Paslaru Bontas, Malgorzata Mochol, and Robert Tolksdorf. Case studies on ontology reuse. *Proceedings of the IKNOW05 International Conference on Knowledge Management*, 74, 2005.

[23] Enrico G Caldarola, Antonio Picariello, and Antonio M Rinaldi. An approach to ontology integration for ontology reuse in knowledge based digital ecosystems. In *Proceedings of the 7th International Conference on Management of computational and collective intElligence in Digital EcoSystems*, pages 1–8. ACM, 2015.

[24] Enrico G Caldarola and Antonio M Rinaldi. An approach to ontology integration for ontology reuse. In *Information Reuse and Integration (IRI), 2016 IEEE 17th International Conference on*, pages 384–393. IEEE, 2016.

[25] Gianfranco Modoni, Enrico Caldarola, Walter Terkaj, and Marco Sacco. The knowledge reuse in an industrial scenario: A case study. In *eKNOW 2015, The Seventh International Conference on Information, Process, and Knowledge Management*, pages 66–71, 2015.

[26] Enrico G Caldarola, Marco Sacco, and Walter Terkaj. Big data: The current wave front of the tsunami. *Applied Computer Science*, 10(4), 2014.

[27] Enrico Giacinto Caldarola, Antonio Picariello, and Daniela Castelluccia. Modern enterprises in the bubble: Why big data matters. *ACM SIGSOFT Software Engineering Notes*, 40(1):1–4, 2015.

[28] Enrico Giacinto Caldarola and Antonio Maria Rinaldi. Big data: A survey - the new paradigms, methodologies and tools. In *Proceedings of 4th International Conference on Data Management Technologies and Applications*, pages 362–370, 2015.

[29] Giorgos Flouris, Dimitris Plexousakis, and Grigoris Antoniou. A classification of ontology change. In *SWAP*, 2006.

[30] Jérôme Euzenat, Pavel Shvaiko, et al. *Ontology matching*, volume 18. Springer, 2007.

[31] Helena Sofia Pinto and João P Martins. Ontologies: How can they be built? *Knowledge and Information Systems*, 6(4):441–464, 2004.

[32] Yannis Kalfoglou and Marco Schorlemmer. Ontology mapping: the state of the art. *The knowledge engineering review*, 18(01):1–31, 2003.

[33] Jeff Heflin and James Hendler. Dynamic ontologies on the web. In *AAAI/IAAI*, pages 443–449, 2000.

[34] Natalya Fridman Noy and Mark A Musen. Smart: Automated support for ontology merging and alignment. *Proc. of the 12th Workshop on Knowledge Acquisition, Modelling, and Management (KAW'99), Banf, Canada*, 1999.

[35] Natalya Fridman Noy and Mark A Musen. Algorithm and tool for automated ontology merging and alignment. In *Proceedings of the 17th National Conference on Artificial Intelligence (AAAI-00). Available as SMI technical report SMI-2000-0831*, 2000.

[36] Natalya F Noy and Mark A Musen. Anchor-prompt: Using non-local context for semantic matching. In *Proceedings of the workshop on ontologies and information sharing at the international joint conference on artificial intelligence (IJCAI)*, pages 63–70, 2001.

[37] Hans Chalupsky. Ontomorph: A translation system for symbolic knowledge. In *KR*, pages 471–482, 2000.

[38] Gerd Stumme and Alexander Maedche. Fca-merge: Bottom-up merging of ontologies. *IJCAI*, 1:225–230, 2001.

[39] Deborah L McGuinness, Richard Fikes, James Rice, and Steve Wilder. The chimaera ontology environment. *AAAI/IAAI*, 2000:1123–1124, 2000.

[40] Abadie Nathalie. Schema matching based on attribute values and background ontology. *12th AGILE International Conference on Geographic Information Science*, 1(1):1–9, 2009.

[41] Gio Wiederhold. Large-scale information systems. *Database Applications Semantics*, page 34, 2016.

[42] Samira Babalou, Mohammad Javad Kargar, and Seyyed Hashem Davarpana. A comprehensive review of the ontology matching systems by a focus on large ontologies. In *International Congress of Chemical and Process Engineering. Prague, Czech Republic*, pages 357–373, 2014.

[43] Mathieu d'Aquin, Anne Schlicht, Heiner Stuckenschmidt, and Marta Sabou. Ontology modularization for knowledge selection: Experiments and evaluations. In *International Conference on Database and Expert Systems Applications*, pages 874–883. Springer, 2007.

[44] Chiara Del Vescovo, Damian DG Gessler, Pavel Klinov, Bijan Parsia, Ulrike Sattler, Thomas Schneider, and Andrew Winget. Decomposition and modular structure of bioportal ontologies. In *International Semantic Web Conference*, pages 130–145. Springer, 2011.

[45] Xiang Zhang, Gong Cheng, and Yuzhong Qu. Ontology summarization based on rdf sentence graph. In *Proceedings of the 16th international conference on World Wide Web*, pages 707–716. ACM, 2007.

[46] Erhard Rahm. Towards large-scale schema and ontology matching. In *Schema matching and mapping*, pages 3–27. Springer, 2011.

[47] Erhard Rahm. The case for holistic data integration. In *East European Conference on Advances in Databases and Information Systems*, pages 11–27. Springer, 2016.

[48] Isabel F Cruz, Flavio Palandri Antonelli, and Cosmin Stroe. Agreementmaker: efficient matching for large real-world schemas and ontologies. *Proceedings of the VLDB Endowment*, 2(2):1586–1589, 2009.

[49] Ernesto Jiménez-Ruiz and Bernardo Cuenca Grau. Logmap: Logic-based and scalable ontology matching. In *International Semantic Web Conference*, pages 273–288. Springer, 2011.

[50] Toralf Kirsten, Anika Gross, Michael Hartung, and Erhard Rahm. Gomma: a component-based infrastructure for managing and analyzing life science ontologies and their evolution. *Journal of biomedical semantics*, 2(1):6, 2011.

[51] DuyHoa Ngo and Zohra Bellahsene. Yam++: A multi-strategy based approach for ontology matching task. In *International Conference on Knowledge Engineering and Knowledge Management*, pages 421–425. Springer, 2012.

[52] David Aumueller, Hong-Hai Do, Sabine Massmann, and Erhard Rahm. Schema and ontology matching with coma++. In *Proceedings of the 2005 ACM SIGMOD international conference on Management of data*, pages 906–908. Acm, 2005.

[53] Christiane Fellbaum. Wordnet. *The Encyclopedia of Applied Linguistics*, 1998.

[54] Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. Dbpedia: A nucleus for a web of open data. In *The semantic web*, pages 722–735. Springer, 2007.

[55] Fausto Giunchiglia, Pavel Shvaiko, and Mikalai Yatskevich. S-match: an algorithm and an implementation of semantic matching. In *The semantic web: research and applications*, pages 61–75. Springer, 2004.

[56] Jetendr Shamdasani, Tamás Hauer, Peter Bloodsworth, Andrew Branson, Mohammed Odeh, and Richard McClatchey. Semantic matching using the umls. In *The Semantic Web: Research and Applications*, pages 203–217. Springer, 2009.

[57] Olivier Bodenreider. The unified medical language system (umls): integrating biomedical terminology. *Nucleic acids research*, 32(suppl 1):D267–D270, 2004.

[58] J Euzenat, M Ehrig, and RG Castro. Towards a methodology for evaluating alignment and matching algorithms. *Technical Report, Ontology Alignment Evaluation Initiative (OAEI)*, 2005.

[59] Heiko Paulheim, Sven Hertling, and Dominique Ritze. Towards evaluating interactive ontology matching tools. In *Extended Semantic Web Conference*, pages 31–45. Springer, 2013.

[60] Jérôme Euzenat, Christian Meilicke, Heiner Stuckenschmidt, Pavel Shvaiko, and Cássia Trojahn. Ontology alignment evaluation initiative: six years of experience. In *Journal on data semantics XV*, pages 158–192. Springer, 2011.

[61] Guus Schreiber. *Knowledge engineering and management: the CommonKADS methodology*. MIT press, 2000.

[62] Mari Carmen Suárez-Figueroa, Asunción Gómez-Pérez, Enrico Motta, and Aldo Gangemi. *Ontology engineering in a networked world*. Springer Science & Business Media, 2012.

[63] Barry Smith, Michael Ashburner, Cornelius Rosse, Jonathan Bard, William Bug, Werner Ceusters, Louis J Goldberg, Karen Eilbeck, Amelia Ireland, Christopher J Mungall, et al. The obo foundry: coordinated evolution of ontologies to support biomedical data integration. *Nature biotechnology*, 25(11):1251–1255, 2007.

[64] Amir Ghazvinian, Natalya F Noy, and Mark A Musen. How orthogonal are the obo foundry ontologies? *Journal of biomedical semantics*, 2(2):1, 2011.

[65] Zuoshuang Xiang, Mélanie Courtot, Ryan R Brinkman, Alan Ruttenberg, and Yongqun He. Ontofox: web-based support for ontology reuse. *BMC research notes*, 3(1):1, 2010.

[66] Nelson K. Y. Leung, Sim Kim Lau, Joshua Fan, and Nicole Tsang. An integration-oriented ontology development methodology to reuse existing ontologies in an ontology development process. In *Proceedings of the 13th International Conference on Information Integration and Web-based Applications and Services*, iiWAS '11, pages 174–181, New York, NY, USA, 2011. ACM.

[67] Tejal Shah, Fethi Rabhi, Pradeep Ray, and Kerry Taylor. A guiding framework for ontology reuse in the biomedical domain. In *2014 47th Hawaii International Conference on System Sciences*, pages 2878–2887. IEEE, 2014.

[68] Matteo Gaeta, Francesco Orciuoli, Stefano Paolozzi, and Saverio Salerno. Ontology extraction for knowledge reuse: The e-learning perspective. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, 41(4):798–809, 2011.

[69] Mélanie Courtot, Frank Gibson, Allyson L Lister, James Malone, Daniel Schober, Ryan R Brinkman, and Alan Ruttenberg. Mireot: The minimum information to reference an external ontology term. *Applied Ontology*, 6(1):23–33, 2011.

[70] María Poveda Villalón, Mari Carmen Suárez-Figueroa, and Asunción Gómez-Pérez. The landscape of ontology reuse in linked data. 2012.

[71] Paul Doran. Ontology reuse via ontology modularisation. In *KnowledgeWeb PhD Symposium*, volume 2006. Citeseer, 2006.

[72] Michel Dumontier and Natalia Villanueva-Rosales. Three-layer owl ontology design. In *WoMO*. Citeseer, 2007.

[73] Harith Alani. Position paper: ontology construction from online ontologies. In *Proceedings of the 15th international conference on World Wide Web*, pages 491–495. ACM, 2006.

[74] Miriam Fernández, Iván Cantador, and Pablo Castells. Core: A tool for collaborative ontology reuse and evaluation. 2006.

[75] Rik Van Bruggen. *Learning Neo4j*. Packt Publishing Ltd, 2014.

[76] Paul Shannon, Andrew Markiel, Owen Ozier, Nitin S Baliga, Jonathan T Wang, Daniel Ramage, Nada Amin, Benno Schwikowski, and Trey Ideker. Cytoscape: a software environment for integrated models of biomolecular interaction networks. *Genome research*, 13(11):2498–2504, 2003.

[77] Weiwei Cui and Huamin Qu. A survey on graph visualization. *PhD Qualifying Exam (PQE) Report, Computer Science Department, Hong Kong University of Science and Technology, Kowloon, Hong Kong*, 2007.

[78] William T Tutte. How to draw a graph. *Proc. London Math. Soc*, 13(3):743–768, 1963.

[79] Helen Purchase. Which aesthetic has the greatest effect on human understanding? In *Graph Drawing*, pages 248–261. Springer, 1997.

[80] Helen C Purchase, Robert F Cohen, and Murray James. Validating graph drawing aesthetics. In *Graph Drawing*, pages 435–446. Springer, 1996.

[81] Thomas MJ Fruchterman and Edward M Reingold. Graph drawing by force-directed placement. *Software: Practice and experience*, 21(11):1129–1164, 1991.

[82] Peter Eades. A heuristics for graph drawing. *Congressus numerantium*, 42:146–160, 1984.

[83] Emden R Gansner and Stephen C North. Improved force-directed layouts. In *Graph Drawing*, pages 364–373. Springer, 1998.

[84] Jaap Kamps and Maarten Marx. Visualizing wordnet structure. *Proc. of the 1st International Conference on Global WordNet*, pages 182–186, 2002.

[85] Christopher Collins. Wordnet explorer: applying visualization principles to lexical semantics. *Computational Linguistics Group, Department of Computer Science, University of Toronto, Toronto, Ontario, Canada*, 2006.

[86] Christopher Collins. Docuburst: Radial space-filling visualization of document content. *Knowledge Media Design Institute, University of Toronto, Technical Report KMDI-TR-2007-1*, 2007.

[87] Zlatan Dragisic, Valentina Ivanova, Patrick Lambrix, Daniel Faria, Ernesto Jiménez-Ruiz, and Catia Pesquita. User validation in ontology alignment. In *International Semantic Web Conference*, pages 200–217. Springer, 2016.

[88] Andrew Smith. *The Oxford encyclopedia of food and drink in America*, volume 2. OUP USA, 2013.

[89] Natalya F Noy, Nigam H Shah, Patricia L Whetzel, Benjamin Dai, Michael Dorf, Nicholas Griffith, Clement Jonquet, Daniel L Rubin, Margaret-Anne Storey, Christopher G Chute, et al. Bioportal: ontologies and integrated data resources at the click of a mouse. *Nucleic acids research*, 37(suppl 2):W170–W173, 2009.

[90] Mikael Laakso and AO Kiviniemi. The ifc standard: A review of history, development, and standardization, information technology. *ITcon*, 17(9):134–161, 2012.

[91] Bianca Scholten. *The road to integration: A guide to applying the ISA-95 standard in manufacturing*. Isa, 2007.

[92] Jakob Beetz, Jos Van Leeuwen, and Bauke De Vries. Ifcowl: A case of transforming express schemas into ontologies. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, 23(01):89–101, 2009.

[93] Steven Joseph Fenves. *Core product model for representing design information*. US Department of Commerce, Technology Administration, National Institute of Standards and Technology, 2001.

[94] Bernd Bruegge and Allen H Dutoit. *Object-Oriented Software Engineering Using UML, Patterns and Java-(Required)*. Prentice Hall, 2004.

[95] Brian McBride. Jena: Implementing the rdf model and syntax specification. In *Proceedings of the Second International Conference on Semantic Web-Volume 40*, pages 23–28. CEUR-WS. org, 2001.

[96] Vladimir I Levenshtein. Binary codes capable of correcting deletions, insertions and reversals. In *Soviet physics doklady*, volume 10, page 707, 1966.

[97] Zhibiao Wu and Martha Palmer. Verbs semantics and lexical selection. In *Proceedings of the 32nd annual meeting on Association for Computational Linguistics*, pages 133–138. Association for Computational Linguistics, 1994.

[98] James H Martin and Daniel Jurafsky. Speech and language processing. *International Edition*, 710, 2000.

[99] Dekang Lin. An information-theoretic definition of similarity. In *ICML*, volume 98, pages 296–304. Citeseer, 1998.

[100] Antonio M Rinaldi. A content-based approach for document representation and retrieval. In *Proceedings of the eighth ACM symposium on Document engineering*, pages 106–109. ACM, 2008.

[101] Yuhua Li, Zuhair A Bandar, and David McLean. An approach for measuring semantic similarity between words using multiple information sources. *Knowledge and Data Engineering, IEEE Transactions on*, 15(4):871–882, 2003.

[102] Natalya F Noy, Deborah L McGuinness, et al. Ontology development 101: A guide to creating your first ontology, 2001.

[103] Jim Webber. A programmatic introduction to neo4j. In *Proceedings of the 3rd annual conference on Systems, Programming, and Applications: Software for Humanity*, pages 217–218. ACM, 2012.

[104] Enrico G Caldarola, Antonio Picariello, and Antonio M Rinaldi. Big graph-based data visualization experiences: The wordnet case study. In *Knowledge Discovery, Knowledge Engineering and Knowledge Management (IC3K), 2015 7th International Joint Conference on*, volume 1, pages 104–115. IEEE, 2015.

[105] Enrico Giacinto Caldarola and Antonio M Rinaldi. Improving the visualization of wordnet large lexical database through semantic tag clouds. In *Big Data (BigData Congress), 2016 IEEE International Congress on*, pages 34–41. IEEE, 2016.

[106] Enrico Giacinto Caldarola, Antonio Picariello, and Antonio M Rinaldi. Experiences in wordnet visualization with labeled graph databases. In *International Joint Conference on Knowledge Discovery, Knowledge Engineering, and Knowledge Management*, pages 80–99. Springer, 2015.

[107] Enrico G. Caldarola, Antonio Picariello, Antonio M. Rinaldi, and Marco Sacco. Exploration and visualization of big graphs - the dbpedia case study. In *Proceedings of the 8th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management - Volume 1: KDIR, (IC3K 2016)*, pages 257–264, 2016.

[108] Enrico G. Caldarola, Antonio Picariello, and Antonio M. Rinaldi. Wordnet exploration and visualization in neo4j - a tag cloud based approach. In *Proceedings of the The Eighth International Conference on Information, Process, and Knowledge Management, Venice, April 2006*, pages 94–99, 2016.

[109] Stephen G Kobourov. Spring embedders and force directed graph drawing algorithms. *arXiv preprint arXiv:1201.3011*, 2012.

[110] Karamjit Kaur and Rinkle Rani. Modeling and querying data in nosql databases. In *Big Data, 2013 IEEE International Conference on*, pages 1–7. IEEE, 2013.

[111] Florian Holzschuher and René Peinl. Performance of graph query languages: comparison of cypher, gremlin and native access in neo4j. In *Proceedings of the Joint EDBT/ICDT 2013 Workshops*, pages 195–204. ACM, 2013.

[112] Thomas H Cormen. *Introduction to algorithms*. MIT press, 2009.

[113] Daniel Delling, Peter Sanders, Dominik Schultes, and Dorothea Wagner. Engineering route planning algorithms. In *Algorithmics of large and complex networks*, pages 117–139. Springer, 2009.

[114] Yiming Yang. A study of thresholding strategies for text categorization. In *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 137–145. ACM, 2001.

[115] Johannes Fürnkranz. Round robin classification. *Journal of Machine Learning Research*, 2(Mar):721–747, 2002.

[116] David MW Powers. The problem of area under the curve. In *Information Science and Technology (ICIST), 2012 International Conference on*, pages 567–573. IEEE, 2012.

[117] Hong-Hai Do, Sergey Melnik, and Erhard Rahm. Comparison of schema matching evaluations. In *Net. ObjectDays: International Conference on Object-Oriented and Internet-Based Technologies, Concepts, and Applications for a Networked World*, pages 221–237. Springer, 2002.

[118] Marc Ehrig and Jérôme Euzenat. Relaxed precision and recall for ontology matching. In *Proc. K-Cap 2005 workshop on Integrating ontology*, pages 25–32. No commercial editor., 2005.