Ph.D. Thesis

Arianna Pipitone

February, 15 2012

Abstract

Ontologies have been designed to capture the semantic knowledge of a domain in a machine understandable form. Current standards for managing ontologies like OWL are lacking in linguistic grounding, and are not able to achieve a clear link with natural language. Bridging this gap, unskilled users could be able to infer the information described in the ontology and it would be possible either producing or parsing utterances about the represented domain automatically. Moreover, as in the case of enterprises, it could be very useful to extract information from external documental corpora that are related to the same domain. Many attempts have been made with the aim to create a natural language interface to ontology but very few of them use grammars during interaction; such interfaces are focused only on verbalizing information contained in the ontology, while it is often necessary to give exhaustive answers to the users's queries by retrieving data outside of the knowledge base. The work presented in this thesis has been inspired by theories in the field of Cognitive Linguistics, and in particular by the Construction Grammar, to create a grammar-based tool for ontology verbalization that combines OWL ontologies with Fluid Construction Grammar (FCG). Currently, FCG is the only computational implementation of Construction Grammar that performs both production, and parsing using the same set of constructions. The main idea is to compute both lexical and grammatical constructions (the meaning-form couples) in the FCG from the ontology. To achieve this goal, a suitable set of rules based on linguistic typology have been defined to infer semantics and syntax from the RDF triples inside the OWL structure, while combining them as the poles of constructions in the FCG. To allow verbalization from external documents related to the ontology, synonymous constructions have been defined, which are matched to free text. Computing all possible syntactic forms for the same meaning was achieved using linguistic rules. The information retrieval procedure outlined above allows semantic annotation of the text as a side effect. A system for automatic generation of contents for Semantic MediaWiki from standard Wikipedia pages has been implemented in this respect. The combination of OWL with FCG and the integration of external documents in the ontology are the core of the system whose theoretical background, modeling, design, and evaluation with respect to other contributes in this research field form the main subjects of this thesis.

Acknowledgements

Simply I am grateful to my supervisor, Roberto; he has made my doctoral experience a life experience, supporting me during my hard moments and bearing me. Discussions with him made me more aware and his comments are always improved my work.

Thanks to Giuseppe Russo, which was my first referent and whose kindness and gentlness are unforgettable for me.

Thanks Vincenzo, Alessandra, Valeria and Agnese, you have made my experience more agreeable and I think I found something that is similar to friendship!

And thanks to my family without which I would not be who I am today, a happy doctorated wife and mother.

Contents

List of Figures

Chapter 1

Introduction

Natural language production and understanding about a domain are the main subjects of this thesis. The research fields involved in achieving these tasks are Natural Language Processing (NLP) and Knowledge-Based Systems (KBS) that are two subfields of Artificial Intelligence, and complement each other. Machines become intelligent when they achieve the precise semantics necessary for computational purposes. If a system *understands* a domain then it can *produce* natural language utterances about this domain: it becomes able to automatically answer domain natural language queries.

A number of developmental stages have been reached in NLP research regarding the evolution of language in artificial agents. Most of the results obtained so far in this field are related mainly to the emergence of lexicons (Batali (2002), Cangelosi and Parisi (2002), Minett and Wang (2005), Steels (2005)). It is obvious that a non-trivial grammar background produces better results. Grammar in the context of knowledge verbalization depends on the domain representation that is the focus for the *Knowledge Representation (KR)* field, a sub-field of KBS.

Generally a domain is represented by conceptual models into machines (internal domain KR) but a set of documents that could not have a structure (as plain texts) and without direct links to this internal representation (external domain KR) can describe the domain too. Systems employ their internal knowledge to become more self-aware and to infer new knowledge from external sources; this is a *Knowledge Acquisition (KA)* process that is perceived as the most difficult bottleneck in the design of KBS.

Internal knowledge represents what the agent knows about the domain; logic is necessary to formalize KR and it is relevant for reasoning in KA processes. Recent advances in the formalization of KR have been influenced by the principles of the Semantic Web. Making interoperable all the heterogeneous information sources in the Web allows applications to understand, use, reason, and share information about them. To this aim, a standard specification of *semantics*, *syntax*, and *structure* for representing web contents is necessary.

The Resource Description Framework (RDF) (Manola and Miller (2004)] starts from the general perspective that the knowledge about a domain is modeled as a set of entities that own properties, and are related to each other. RDF encapsulates this representation through a set of *triples* with the form

```
RESOURCE <property> VALUE
```

To model the triple, RDF defines three properties: *subject* to identify the resource, *predicate* to identify the property, and *object* to identify the value. A resource with these properties models an *RDF statement*; statements together form particular representation that is called *ontology*.

This thesis has been motivated by the aim to enrich the KR of a system with both external contents and rules for defining an usable grammar to verbalize the domain. The work presents a formalization of the RDF statements describing entities, and their properties in the ontology through a model inspired to Cognitive Linguistics. The Construction Grammar (CxG) theory is proposed in this thesis. CxG considers together both semantics, and syntax of a grammatical structure. In this perspective, a RDF triple becomes a grammatical unit (i.e. a *construction*) while the triple's semantics and its structure become the "poles" of this unit. The set of constructions (that is the set of RDF triples in the ontology representation) is the knowledge of the system about the domain. Luc Steels triggered the development of the *Fluid Construction Grammar* (FCG)(Steels (2011)) (FCG) an operationalization of CxG that produces and parses text using the same set of constructions. FCG has been used in the proposed software architecture to verbalize the domain. A natural language query is translated in the selection of a subset of constructions (the RDF subgraph), and the answer is produced from them.

Briefly, this thesis aims to use the models previously outlined *to model an agent's internal KR using external KR* and *to make a system able to verbalize the domain under investigation using its global knowledge (both internal, and external information sources)*. The system will be able to produce Natural Language sentences that can be answers to Natural Language queries. So, the proposed approach implements an intelligent *Natural Language Interface*. Figure 1.1 shows the discussed scenario.

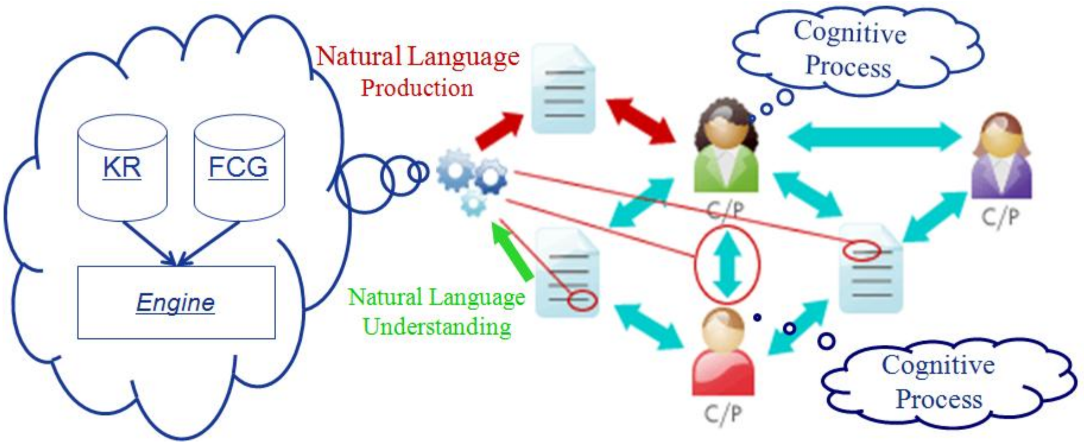


Figure 1.1: How our model is collocated into the Semantic Web scenario.

1.1 Motivations

Although several experiments have been made on the role of syntax and grammar (Hashimoto and Ikegami (1996), Steels (1998), Batali (1999), Smith (2003)) in agents' production and understanding, non trivial grammars that are well grounded in the agent's task arose only in few cases. Grammar emergence is a much more encompassing problem than simply devising a suitable lexicon for ontology statements. In grammar emergence experiments, the agent's knowledge model needs to be much more complex, and it is influenced by problems related to domain representation conventions. Integrating the model representation with rules for defining a grammar can be very useful for bridging this gap.

Knowledge representation languages built on RDF (as OWL, which is the W3C standard representation language for ontology) are becoming increasingly popular. Often, RDF/OWL formalisms are used to model an agent's internal KR for production and/or understanding purposes. In this respect, some problems arise when the agent has to deal with plain web contents. Raw data are shared by users with neither structure nor semantic annotations. Unskilled users don't mind to structure their contents. As a consequence an *incremental on-the-fly* strategy is required to enable reuse of existing data, while avoiding their loss, and to structure new contents. When retrieving plain Web contents, the agent must infer automatically standard structures from them without involving the user, then it shares this new form (on-the-fly mode) on the Web replacing the old one. Documents retrieved by users at least once, become structured (incremental solution), and massive standardization interventions are not necessary. Another problem is related to querying structured data by unskilled users. Formal query languages like SPARQL (Simple Protocol And RDF Query Language) (Prud'hommeaux and Seaborne (2008)) are required when accessing OWL information to map natural language queries onto the data structure. This poses significant difficulties for non-expert users, while experts need to be familiar with the existing ontology structure that is often hidden. Also these query languages do not explore contents that are external to the ontology even if they're interesting for the users, and are related to the domain. Finally, accessing external contents allows structuring them to standardize them according to an incremental on-the-fly approach. In (Hurtado et al. (2009)) it is demonstrated that casual users need to be able to access the data despite their queries do not match exactly the queried data structures.

According to the study on interfaces evaluation conducted in (Kaufmann and Bernstein (2007)), systems developed to support Natural Language Interfaces are perceived as the most acceptable by end-users. If a system understands (i.e. achieves semantic parsing of) free textual documents about a domain, it enriches its KB with new knowledge that in turn can be used to produce more exhaustive answers. This is particularly interesting for enterprises that use corporate ontology for their knowledge management. Corporate ontology is usually a logical schema of enterprise roles and concepts and relationship between them (TBox) 8Antoniou and Harmelen (2008)), while data and documents related to it are located into an external database. In this way, enterprises should be able to communicate between them sharing the knowledge representation but preserving documents and data through proper security strategies.

Our model allows linking the ontology and the documents related to the represented domain. Semantic annotation of text through the ontology itself allows unskilled users to query the system about the domain, and external corpora: the system will answer producing natural language sentences.

In Buitelaar et al. (2009) the main motivations in support of this position are reported. According to the authors, a natural language background for ontologies helps for:

- creating sound ontologies by human developers, which are so able to better understand the concepts represented in the KB and the relations between them;
- extracting information from plain text about topics related to the domain in order to develop ontology-based semantic parsers;
- generating utterances in natural language from ontology structures to make them understandable by human users.

1.2 Challenges and Solutions

According to Martin et al. (1986), a major challenge when building an intelligent system to understand free textual documents is the existent gap between the way the user thinks about the domain and how she describes it, and the way the domain knowledge is structured for computer processing. This implies that it is very important to consider the ontology structure and content.

Two ontologies describing identical domains can use different modeling conventions. As an example, Two different OWL structures represent a postal addresses either through a datatype property *address* in the class *Person* or using instances of a suited *Address* class. A NLP system aimed to verbalize the domain would have to support both types of conventions: *portable* or *transportable* NLP systems can be adapted easily to new domains (i.e. new ontologies covering the same domains). Some authors (Martin et al. (1986)) consider that building transportable systems poses a number of technical and theoretical problems because many of the techniques developed for specialized systems preclude automatic adaptation to new domains. (Kaufmann and Bernstein (2007) p.281) noted that portability affects retrieval performance: "the more a system is tailored to a domain, the better its retrieval performance is".

In general, talking about a "tailored domain" is more related to applications' requirements for a particular data representation than the domain itself. In this perspective, the designers cover applications' requirements and do not think about a general view of the domain. They work without following standard guidelines but according to application profiles or users needs. An effect of customization is that it does not guarantee a standardization of knowledge representations, which is a main objective for the Web 3.0.

In general, existing knowledge management systems tend to be either domain independent (i.e., portable) but with lower performance, or more domain-specific (i.e., portable only with prior customization) but with a much better performance. The caveat in the latter case is that customization tends to be very expensive as it is performed by experts (e.g., domain experts, language engineers); moreover in this case the ontology could be representative only of a part of the domain. It could be considered as a view of the world that depends by both the designer experiences, and the modeling activity performed by domain experts.

Systems using ontologies should be able to capture the semantics of the representation independently from syntax and modeling conventions in order to solve the portability problem. As a consequence they should be domain-independent. The basic approach proposed in this thesis is extracting semantics by means of the *semantic seed* of the ontology representation. Each fact in the ontology is represented using a triple, independently from modeling conventions. A semantic seed is the semantics of the *subject-verb-object* triple (SVO), and each triple in the ontology can be mapped onto a SVO triple. In the above example using either a datatype property or an instance of a particular class corresponds to some RDF triples in the OWL file. If the address of a person is modeled as either a class or a datatype property, the SVO form is always inferable: `<Person has_address Address>` is the SVO triple in the first case, while `<Person has_address String>` is the SVO triple for the datatype property modeling case.

The main challenge of the present work is to make a system independent from modeling conventions by using semantic seeds (or a taxonomy of them) to infer the SVO triples in the ontology, and to compute a suitable syntax to express them. In turn, such a syntax can be used for parsing and understanding textual contents: this position is supported by *linguistic typology*, a subfield of Linguistics that studies and classifies languages according to their structural features. Its aim is to describe and explain the common properties and the structural diversity of the world's languages. In linguistic typology, the subject-verb-object sentence structure is one of the most common structures in the world: it's the second after the subject-object-verb one, and they account for more than 75% of the world's languages (Crystal (1998)). The system proposed in this work searches free text documents, and tries to find all possible statements expressing a particular triple in the ontology that is different syntactic forms for the same semantics.

Another challenge for the presented work was the *ambiguity* problem: different meanings can exist for the same syntactic form. The impact of ambiguity problem is reduced considering that the semantic seed can disambiguate context. For example, an utterance containing the structure *born in* in relation to a person can bring two meanings: the place of birth or the year of birth. This ambiguity is resolved considering the components of the seed. As an example, if *born in* is modeled in the ontology as an object property connecting the class `Person` and another class (i.e. the object in the SVO structure) then the object component of this property has to be analyzed: if the object is a class like `Place`, the property means a place of birth, while if it is a class like `Date` or `Year` the property means a year of birth. If the property is modeled through a datatype property, the value of its range is useful to disambiguate this case. In fact, considering the Named Entity (Finkel et al. (2005), Finkel and Manning (2009)) categories of the property range, is possible to infer if it is a place or a year and hence to disambiguate the sense. In summary, disambiguation is intrinsic for each triple, and it is sufficient to apply the right inference according the form of the seed.

Expanding the ontology by creating either new triples or triple instances is an interesting topic. Full incremental ontologies are useful when it is necessary to retrieve new information from sources containing unstructured data and changing over time. In general, ontologies tailored for specific applications (as in the case of corporate ontologies) do not require to add new concepts and relations because their knowledge model is defined ad-hoc for some purposes. On the contrary they require adding new class and/or property instances. The approach proposed in this thesis addresses this challenge. An example is reported in Figure 1.2

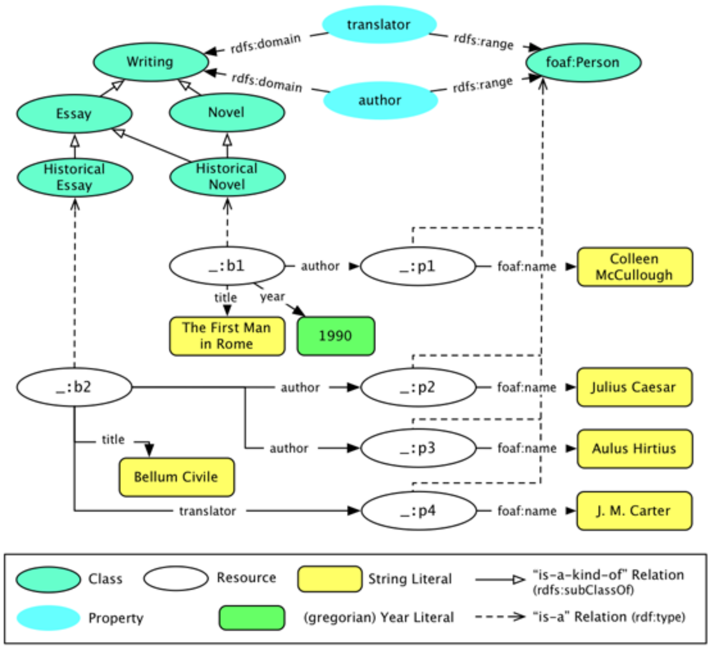


Figure 1.2: An ontology fragment that models books.

In this subgraph the b_1 instance of the *Historical Novel* class has two datatype properties and one object property linking it to the p_1 instance that is a *Person* according to the model in FOAF (Brickley and Miller (2010)). Properties for b_1

are described by the following set of SVO triples (without namespaces):

`<`b_1` is-a Historical Novel>`

`<`b_1` title The First Man in Rome>`

`<`b_1` year 1990>`

`<`b_1` author `p_1`>`

The string “1990” is a year because it has been declared explicitly by the designer as a Year Literal value. The same holds for “The First Man in Rome” that is the title of b_1 , and has been declared as String Literal.

Assuming a scenario where information is missing about the author of the *The First Man in Rome*, and the year when it was written, the correspondent nodes in the figure would be empty. The system implemented in this work devises the subsentence *was written* as a possible syntax both for both the `<year>` and the `<author>` relations. These relations are the semantic seeds to start computation.

When retrieving something like:

The First Man in Rome was written in 1990.

or

The First Man in Rome was written by Colleen McCullogh.

the *was written* subsentence alone does not help to infer if the text refers to either the year or the author. Disambiguation is achieved considering that a year can be modeled as a Date because it is a Year Literal, and that an author is a Person. Even if the year would have been modeled just as a String Literal, the system could be able to disambiguate the two sentences in the example using the name “year” of the datatype property belonging to $b1$. Such information is then added to the ontology.

Briefly, the ontology provides the semantic seeds that are enriched by means of a grammar analysis with a set of plausible syntactic forms. Then, such forms are searched for in the plain text: complete meaning is devised against the ontology structure using entities’ and properties’ types and names. Semantic annotation of the text is achieved as a side effect, where the tag set is built using such terms.

1.3 Logbook of Contributes

This section illustrates the working steps done to obtain understanding and verbalization of natural language sentences related to a domain that is modeled using ontology.

The first attempt to extract meaning from plain text using ontologies, has been made on Wikipedia pages to obtain automatically annotated Semantic MediaWiki (Krötzsch et al. (2006)) pages. Annotated Wikipedia documents were used to expand the knowledge representation of an Intelligent Tutoring System (ITS) focused on the Art domain, as it is shown in (Russo et al. (2009)). An ITS is an artificial agent aimed to interact with students to assess their skills, and to improve their knowledge in a particular domain. The course knowledge is modeled using the ontology which has the purpose to define and to connect the main concepts of the domain in order to build learning paths tailored to the user's learning needs. A knowledge not evolving over time is a severe limitation for ITSs: the agent is unable to reply to questions that are outside the ontology boundaries and is not capable to increase its own knowledge with respect to the interaction with users or to add new external information sources.

Retrieval of new contents is obtained through a mapping between the internal ontology of the agent and the ontology fragment extracted from the annotations in the Semantic MediaWiki page by an ad-hoc service function provided by this site. A big limitation of this approach was the small number of semantic wiki pages available, so the activity focused on their automatic creation developing a specific tool. Defining the basic idea of the thesis was the main contribute of that work: computing the set of (synonymous) syntaxes for a *semantic seed* into the ontology (that is the semantics of a SVO triple). The approach devised in that work was therefore to use explicitly the names of the ontology entities, along with the structural relationships between concepts and their properties. Two strategies have been followed that are discussed in the appendix.

- Using an annotated ontology: the *rdfs:comment* property for each component of a SVO triple is filled with the syntactic roles they take for a specific semantics. As an example, in the triple `<Person-has_address-Address>`, the classes *Person* and *Address* are both annotated with NN tags and the property *has_address* is annotated with V tag, that are respectively the Proper Name and Verb POS tags of PennTreeBank set(Marcus et al. (1993)). This means that a person and an address are syntactically expressed through a Proper Name or a chain of proper names, and the relation between them is a verb. In this way, the syntactic tree of a natural language sentence is extracted by a POS syntactic parser [Spitkovsky et al. (2011)], and it is mapped onto the syntactic annotation of the SVO triple to infer instances for the *Address* and *Person* classes. This approach isn't portable, and it requires the ontology to be annotated suitably.

- Defining a set of *typological* and *way-of-saying* linguistic rules to compute the syntactic expressions for each component of a SVO triple. Typological rules disambiguate the meaning of the SVO triple from the semantic seed structure. Way-of-saying rules compute a possible set of syntactic structures that are useful to express such meaning. This strategy allows obtaining portable systems with respect to both the domain and the actual ontology representation, and it is the approach mainly discussed in this thesis.

While remaining in the framework of building artificial agents for Intelligent Tutoring, the next step of the research described in this thesis was the attempt to produce natural language answers for the user automatically. The theoretical basis for this research was Cognitive Linguistics(Smith (1999)). The main goal in this activity was to find a suitable model to describe the semantic seed also from a grammatical perspective to embed understanding and production in the same model. The result of this study was modeling the semantic seed using the Construction Grammar (CxG) theory. The basic concept in CxG is *construction* that is a couple made by a syntactic structure, and the corresponding meaning. Constructions have been used in this approach to represent both the semantic and syntactic components of each entity in the ontology, and to relate them according to the SVO structure. On the implementation side, the Fluid Construction Grammar (FCG) framework by Luc Steels (Steels (2011)) has been used that is the only computational model for CxG. The system makes use of both constructions defined for the English language, and suitably defined rules to perform semantic parsing of plain text, and producing utterances. In this way the ontology is the only guide to produce a useful grammar for both understanding, and verbalization tasks. This result is supported by the SVO linguistic typology that establishes the strong possibility to find a grammatical structure corresponding to a certain semantics, and its linearization expressions into plain natural language sentences.

The proposed approach addressed the following issues:

- *Ambiguity*: ambiguities are resolved automatically, based on a ranking derived from the ontology structure. Such a ranking is based on *linguistic typological rules* that resolve the meaning of SVO triples according to the form of a semantic seed.
- *Portability*: the base lexicon for the domain under investigation is extracted automatically from the ontology, and it's integrated using *way-of-saying rules* based on Wordnet (Fellbaum (1998)) and Framenet (Ruppenhofer et al. (2005)). Way-of-saying rules specify both the structure, and the relevance of the information taken these sources. Some examples are: *take the Verb Phrase corresponding to a Verb definition* or *consider synonyms and hypernyms for a Name.* Way-of saying rules are used to generate a gazetteer to be processed by the Glen Gazetteer based on GATE (Cunningham et al. (2011)). Its main application is semantic annotation of free text based on the provided ontology. Annotated text is used to enrich the base lexicon. This approach is domain-independent. Changing the ontology means changing the lexicon, and this ensures portability.
- *Language independence*: it is a direct consequence of portability. The language depends on the ontology design that is on the language used to name entities in the ontology structure. typological and way-of-saying rules can be stated in languages that are different from English. Several thesauri have been implemented in different languages. Finally, the FCG framework can be used to state grammatical rules for production other than in English.

The proposed approach has been evaluated in two domains:

- Art domain, which contains facts about artists, artworks, artists' life and so on. The result of this work is WikiArt (Pirrone et al. (2009)), a wiki for Art, and the automatic creation of artists' wiki pages annotated semantically.
- Corporate ontologies that is ontologies to be used by enterprises. Experiments have been made using the ontology made available by the "Consorzio Operativo Monte die Paschi di Siena", which models financial products and the Connected Customers Groups management.

The results achieved so fare are positive. The system performance has been good for language independence and portability. Moreover, the system is not limited to the ontology representation but it retrieves external data, improving the quality of the answers with respect to other systems like AquaLog (Lopez et al. (2011)) and Freya (Damljanovic et al. (2011)). On the other hand, the system doesn't support direct interpretation of natural language queries. Moreover, the problem of grammatically incomplete statements (an article not included, gender is not well defined and so on) has to be taken into account. The present implementation uses PANTO (Wang et al. (2007)) to manage relaxed queries (ill-formed or incomplete) as well as the full-blown grammatically correct questions. User queries are converted in SPARQL to allow the system to identify the ontology fragments corresponding to the requests. Starting from the selected fragment, the system produces an enriched natural language answer composed by ontology concepts, verbalization, and external data. So PANTO is used only for analyzing the query, and not for production. In this respect, the research work will be focused towards understanding of dialogue utterances. The present approach analyzes full documents, performs semantic annotations, and uses these resources to enable production. Full understanding of single utterances containing anaphors and/or co-references requires an effort in managing the dialogue context, and the presented system will be adapted to address this issue.

1.4 Summary

Considering lexical background into semantic resources can support a domain-independent natural language understanding and production engine. The research presented in this thesis focused on the *portability* of NLP systems to obtain a model that contributes to standardize shared and unstructured contents, as in the case of web materials. The work proposes a *domain-independent* and *ontology-independent* model; it can be applied with whatever ontology representation, and in whatever domain. Ontology is considered as the starting knowledge of the system about the domain (that is its initial state) and it uses this view to capture semantics about the domain and to understand free text related to it. The adopted strategy is mainly based on matching linguistic patterns. The system searches for common ways of expressing a meaning represented by a triple in the ontology (i.e. the semantic seed). The main challenges of the work were extraction of such linguistic expressions, and the way to match them to free text taking into account the ambiguity problem. Finally, the presented approach is *language-independent* because it does not depend on the language used to fill the names of the ontology classes and properties, which is different from standard languages used for modeling knowledge through ontologies.

Chapter 2

State of the Art

2.1 Introduction

In this chapter we present the state of the art of the existent interfaces to ontological models. There are two different rasearch trends in this field; former has the aim to make ontology more understandable to unskilled users and proposes rules to translate the syntax of the representation language of the ontology to natural language syntax. Latter considers interfaces to ontology without a language that intermediates with user, but he must have some skills to use these interfaces that are simply and allow to browse the ontology regarless its syntax.
We describe the two strends in the next sections. At the end of this chapter we discuss about the limits of these approaches and we introduce our view to model a natural language interface to ontology.

2.2 The Ontology Verbalization Languages

Formal languages (Monin, 2003) have well-defined syntax, unambiguous semantics, and they can also support formal methods for writing software specifications, for supporting the knowledge acquisition process, and for reasoning [Fuchs et al., 2008]. Due to such properties they have been also mainly suggested as knowledge representation languages.

OWL (Web Ontology Language) is the formal ontology language recommended by W3C(Bechhofer et al. (2004)) for the Semantic Web, and it is useful to represent knowledge provided by domain experts in ontological format. OWL ontologies are build on RDF Schema (Manola and Miller (2004)) and define a specific XML representation (Boley (2001)) of a domain interns of classes, hierarchies, and relations between classes. The main advantage of using OWL to formalize ontologies is that they can be automatically machine-understandable; in turn, this makes further knowledge processing easier. On the other hand, OWL ontologies have become heterogeneous because there is not a standard specifying conceptual principles for modeling a domain, and how to fill their elements (classes and properties). Only the OWL syntax has been standardized. While many OWL ontologies are available for free, their concepts representation have labels that lack a systematic structure. For these reasons it is very difficult to merge them. OWL ontologies are criticized as regards their reuse, transformations to other domains and/or languages, and integration with other ontologies. So, natural language descriptions associated with ontologies have proven to be of major importance not only to support ontology developers and users, but also to assist in tasks such as ontology mapping, information extraction, and natural language generation.

The effort carried out by the scientific community in the development of principled models to associate more complex linguistic descriptions with arbitrary ontologies increased in recent years, and it is due to the problem of defining an ontology verbalization language. There are many languages (such as Hewlett et al. and Jarrar et al. (2006)) that have been suggested in order to represent OWL in a more natural way; however the major shortcoming of these approaches is that they lack any formal check that the resulting expressions are unambiguous. In this sense, better results are obtained from the approaches based on controlled natural languages that typically have a formal language semantics and come with a parser that can convert the statements of the OWL representation so that the natural language version becomes the primary representation interpretable by humans. On the other hand, these languages suffer of excessive constrains in semantics and syntax, and are strictly linked to English language.

2.2.1 Controlled Natural Languages

An important aspect of formal languages is their cognitive distance to the application domain that is not inherent in natural language. One way to bridge the gap between a natural language and a formal language is the usage of Controlled Natural Languages (CNLs) that can mediate between them; CNLs are languages governed by specific grammars and lexicons and they create correspondences between the language understandable by computers and language understandable by humans. A CNL is more readable than typical computer languages, it requires less training for people who write using them, and no training for people who read them.

Controlled Natural Languages are subsets of natural languages which are engineered in the sense that their grammar and vocabulary have been systematic restricted in order to reduce both ambiguity and complexity of full natural languages. These constraints usually have the form of writing rules (Kittredge). CNLs support precise communication:

- for stating requirements and specifications by humans to humans;
- for commands and assertions by humans to computers;
- for answers, explanations, and help from computers to humans.

There are two ways of enforcing control:

- Syntactic control: strict constraints on the permissible grammar.
- Semantic control: strict constraints on the semantics, but an allowance for any grammatical pattern that has a unique semantic interpretation.

Traditionally, CNLs have been grouped into two broad categories: *human-oriented CNLs* and *machine-oriented CNLs* (Huijsen).
The main objective of human-oriented CNLs is to improve the readability and comprehensibility of technical documents as in the case of maintenance documentation

(ASD Simplified Technical English) and to simplify and standardize human-human interaction in specific context like trade or air traffic control (Pool, 2006).
Machine-oriented CNLs are aimed to the improvement of the translatability of technical documents (Nyberg and Mitamura (2000)) and the acquisition, representation, and processing of knowledge (Fuchs et al. (2008)) in particular for the Semantic Web (Schwitter and Tilbrook (2008)).
Human- and machine-oriented CNLs have been designed for different targets, and their field of application are quite different.

There is a number of machine-oriented CNLs that have been designed for using as knowledge representation languages: they are *general-purpose languages* in the sense that they have not been developed for a specific domain. These languages can be used where traditional formal languages are used otherwise. The aim of these languages is to make domain specialists able to use an expressive knowledge representation language that is easy to learn, use and understand, while being fully processable by a computer at the same time. A number of CNLs are used as a front-end to formal languages used in Semantic Web.

Using CNLs has both advantages and disadvantages.

- Syntax: a CNL is more predictable, but it is harder for people to write.
- Semantics: a CNL is more natural and easier to write, but requires an "echo" to show exactly how each sentence is interpreted.

2.2.2 CPL

Computer Processable Language (CPL) (Clark et al. (2005)) is a language developed at Boeing Research and Technology that is capable of translating English sentences to a formal knowledge representation language.
The target language is the Knowledge Machine language (KM). KM is a frame-based language with first-order logic semantics. The KM interpreter employs a sophisticated engine for reasoning, including planning about actions, and using a situation calculus mechanism. The CPL interpreter directly resolves various types

of ambiguities using heuristic rules for prepositional phrase attachment, semantic role labeling, word sense disambiguation, metonymy resolution, compound noun interpretation and other language processing phases.

CPL accepts three kinds of sentences:

- *questions*;
- *ground facts*;
- *rules*.

Defining that NP means Noun Phrase sand PP means Prepositional Phrases, the CLP definitions for the questions sentences have five forms for which the mainly are the two followed:

- What is NP?
- Is it true tha Sentence?

In the case of ground facts, a basic CPL sentence takes one of the following three forms:

- There is-are NP
- NP verb [NP] [PP]
- NP is-are passive-verb [by NP] [PP]

Verbs for these phrases can be auxiliaries or particles, while nouns in a NP can be replaced by other nouns, prepositional phrases, or adjectives.

In the case of rules, CPL accepts sentence patterns of the form:

- IF Sentence [AND Sentence]* THEN Sentence [AND Sentence]*

During CPL parsing, rules that run in parallel to the grammar rules generate implified logical form for basic sentences. A ground KM assertions are created by the logical form too. Rules are entered by the user who writes CPL sentences through a set of seven rule templates. Three of these templates create standard logical implications and the rest describe preconditions and effects of actions. Each CPL sentence is interpreted interactively with user; system paraphrases its interpretation calling back to the user, allowing he to find and resolve misinterpretations. Sentences that express states add facts to a situation, and sentences that express actions trigger rules that update the situation, reflecting the effects of the action into the situation. The user can ask questions about an emerging situation directly in CPL.There is no explicit quantifier for these basic sentences and the disambiguation of word sense and semantic relationships is made by the inference engine that uses WordNet to make a "best guess" of word sense assignments.

There is a slimmed version of CPL named CPL-lite. Each CPL-Lite sentence corresponds to a single binary relation between two entities. There are three kinds of relations: nounlike relations (e.g. the age of <a> is <b>), verblike relations (e.g. <a> causes <b>), and preposition-like relations (e.g. <a> is during <b>). CPL and CPL-Lite have been mainly used to encode general and domain specific commonsense knowledge and to allow knowledge engineers to pose queries in a comprehensible way; CPL-Lite is more verbose and grammatically more restricted.

2.2.3 CLP for Semantic Web

Controlled Language for Semantic Web convert the OWL representation statements in human interpretable representation. ACE, Rabbit, and SOS are three controlled natural languages that have been designed to be used as interface languages to OWL ontologies. Apart from these languages there exist other CNL-based approaches to authoring OWL ontologies but we will not further discuss these languages.

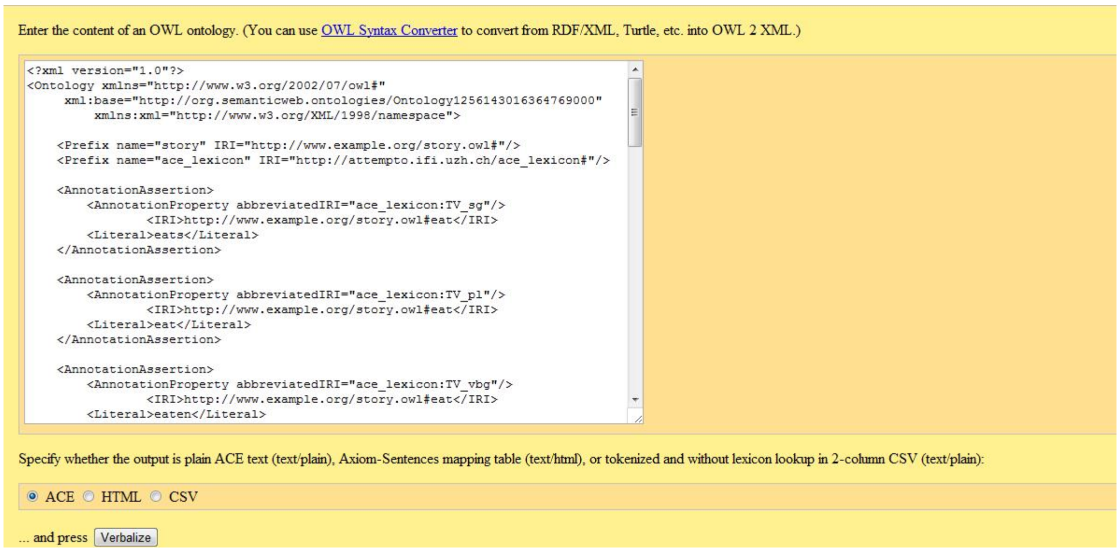
ACE

Attempto Controlled English (ACE) (Fuchs et al. (2008)) is one of the oldest natural language interfaces which have been developed to serve as a formal language for knowledge representation. Unlike NL which is ambiguous, vague and potentially inconsistent, ACE is a controlled English, precisely defined, tractable subset of full English that can automatically and unambiguously be translated into first-order logic.

ACE is defined by a small set of *construction rules* that describe its syntax and a small set of *interpretation rules* that disambiguate constructs that are ambiguous in full English. The vocabulary of ACE consists of a set of predefined *function words* (e.g. determiners, conjunctions, and pronouns), some predefined *fixed phrases* (e.g. there is, it is false that), and *content words* (nouns, proper names, verbs, adjectives, and adverbs). The language processor of ACE is based on grammars that are written in a definite clause grammar (DCG) notation. These DCGs are enhanced with feature structures to translate declarative and interrogative sentences into a first-order logic sentences using a set of discourse representation structures. In particular, ACE uses the DCG directly and anaphoric references are resolved only after construction of a discourse representation structure.

ACE is supported by many tools (AceRules translates ACE into rules and ACE View is a plugin for the ontology editor Protege) and a reasoner (RACE) has been developed and used for reasoning with it. It is important to note that the meaning of words in ACE is not predefined; the user is expected to define their meaning by the ACE sentences or import these definitions from an existing formal ontology. ACE is applied in domains such as software and hardware specifications, database integrity constraints, agent control, legal and medical regulations and ontology construction.

```
Every animal is something that is a cat or that is a goat.

Everything that is eaten by a goat is a leaf.

Everything that eats nothing but leaves is a goat.

Every human is a person that own an automobile.
```



Enter the content of an OWL ontology. (You can use OWL Syntax Converter to convert from RDF/XML, Turtle, etc. into OWL 2 XML.)

```
<?xml version="1.0"?>
<Ontology xmlns="http://www.w3.org/2002/07/owl#"
     xml:base="http://org.semanticweb.ontologies/Ontology1256143016364769000"
         xmlns:xml="http://www.w3.org/XML/1998/namespace">

    <Prefix name="story" IRI="http://www.example.org/story.owl#"/>
    <Prefix name="ace_lexicon" IRI="http://attempto.ifi.uzh.ch/ace_lexicon#"/>

    <AnnotationAssertion>
        <AnnotationProperty abbreviatedIRI="ace_lexicon:TV_sg"/>
                <IRI>http://www.example.org/story.owl#eat</IRI>
        <Literal>eats</Literal>
    </AnnotationAssertion>

    <AnnotationAssertion>
        <AnnotationProperty abbreviatedIRI="ace_lexicon:TV_pl"/>
                <IRI>http://www.example.org/story.owl#eat</IRI>
        <Literal>eat</Literal>
    </AnnotationAssertion>

    <AnnotationAssertion>
        <AnnotationProperty abbreviatedIRI="ace_lexicon:TV_vbg"/>
                <IRI>http://www.example.org/story.owl#eat</IRI>
        <Literal>eaten</Literal>
```

Specify whether the output is plain ACE text (text/plain), Axiom-Sentences mapping table (text/html), or tokenized and without lexicon lookup in 2-column CSV (text/plain):

[✓ ACE] HTML CSV

... and press Verbalize

Figure 2.1: OWL Verbalization in ACE.

```
Every human is something that is John or that is Mary.

Every man is a person.

Everything eats at most 1 thing.

Everything is eaten by at most 1 thing.

If X eats something that eats Y then X eats Y.

Everything that eats something is an animal.

Everything that is eaten by something is a food that is not an
automobile.

If X hate Y then Y eats X.

If X eats Y then Y hate X.

John is a man.

Everything that is an apple or that is a leaf is a food.
```

Also, there is Wiki for ACE through which is possible generating ACE text and obtaining its translation into OWL or SWRL, as shown in Figure 2.1. In the wiki, a predictive editor supports a user while generating sentences among a

subset of ACE. In the ACE wiki authors describe how an ontology that is written in OWL RDF/XML is verbalized in ACE and only the logical content of the OWL file is verbalized, excluded the information about annotations, versioning, import-structure; this tool generates many intelligible sentences. However, there were also phrases like "*Everything that is hasTopping by a Mushroom is something that is a MozzarellaTopping*", and the sentence "*Every American is a NamedPizza*" linked to the generic names used for classes and instances.
In general the ACE rule makes sense, but it can be thwarted by the names used in the ontology. In Fliedl et al. (2010) authors are trying to resolve just such issues. They propose a rigid naming convention to make it easier to verbalize the ontology. We think it is a good proposal, because it is 'blaming' the ontologists for failing natural language generation (NLG) systems, and syntactic verbalization should not be the guiding principle when adding knowledge to the ontology.

Unlike majority of similar systems, ACE supports both knowledge representation and querying. However, the querying is feasible only if the knowledge is generated using ACE sentences. That is, the questions supported work well only if the knowledge has been generated by ACE. ACE supports yes/no and WH-queries. For example, if the ACE sentence which was translated to the knowledge representation was

```
A customer inserts a card.
```

we can ask a question such as:

```
Does a customer insert a card?
```

This feature is unique to ACE and a few other similar systems like CPL.

PENG

PENG (White and Schwitter (2009)) is a CNL that has been designed for an incremental parsing approach and was the first CNL that was supported by a dynamic predictive editor. PENG is similar to ACE but differently from it PENG covers a smaller but fully tractable subset of English, becoming a more light-weight language. The predictive editor provided by PENG is a text- and menu-based writing support enforcing the grammatical constraints of the CNL via lookahead information while a text is written and generating a paraphrase that clarifies the interpretation for each sentence that the user enters. For each word form that the user digits using editor, the chart parser prompts to the user a list of options to inform he about how the sentence structure can be continued ensuring that the text follows the CNLs rules so that it can be translated unambiguously into the formal target language (first-order logic) and be processed by a theorem prover. we use an example from the TPTP problem library. PENG has recently been used for the construction of an interface to a situation awareness system but the language can be used for similar applications to ACE. As well ACE PENG processor uses DCG but in a different manner. DCG sentences are transformed it into a notation processable by a top-down chart parser. The anaphoric references are resolved during the parsing process and a discourse representation structure is built up.

Sydney OWL Syntax

Sydney OWL Syntax (SOS) (Cregan et al. (2007)) is a controlled natural language that has been designed to satisfy the requirements of a modern high-level interface language to OWL. The key properties of SOS are:

- writing OWL ontologies in a well-defined subset of English through a non-logicians support;
- expressing existing ontologies in the same subset of English.

SOS uses the terms of the application domain plus some other terms to convey the meaning of the information. SOS does not allow to say the same thing in different way, enforcing a one-to-one mapping between controlled natural language and OWL Functional-Style Syntax (FSS). Furthermore, the language uses only limited references to OWL constructs like classes and properties. SOS uses only very little linguistic knowledge in order to deal with plural forms and compound constructions (e.g. 'has ... as a part'). A particularly interesting feature of SOS is variables usage - as known from high school math textbooks - which allows to express certain axioms in a very compact and natural way. The language provides specific constructs ('fully defined as' and 'partly defined as') that indicate the logical status of a definition.

In principle, SOS supports nesting of expressions to any level but deep nesting results in structures which are difficult to understand by people. Therefore, it is recommended that authors limit the depth of nesting up to three levels using an authoring tool.

2.3 Flexible Grammars vs Controlled ones

The linguistic approaches presented so far, consider grammars with strict constraints. Even if they are valid solutions to bring human language and formal ones, they exhibit many restrictions in syntax and semantics. Few approaches are able to parse unstructured contents external from ontology. In fact, ontology verbalization languages mentioned above use of a fixed connection between the syntax of the ontology statements and the syntax of their possible verbalizations. The range of lexicons is limited to predefined phrase structures corresponding to the specific meaning of the statement. Often, in these systems the user is guided in topic editing in order to obey such constrains (as in the case of PENG), and to ensure the selection of customized sentences. For these reasons, CNLs are not suitable for understanding, and semantic annotation of free text.

To illustrate a limitation of CNLs, the reader can refer to Kaljurand (2007) where a bidirectional mapping of a ACE fragment to OWL 1.1 (without data properties) is described. This mapping captures all semantically different OWL constructs as different ACE sentences; often there are many possibilities for expressing the same OWL axiom. For example, all the following sentences

```
John likes no man that owns a car.
No man that owns a car is liked by John.
```

```
Every man that owns a car is not liked by John.
If a man owns a car then it is false that John likes the man.
```

map to the same OWL *SubClassOf* -axiom. On the other hand, the mapping does not differentiate between all syntactic forms offered by OWL i.e. syntactically different OWL constructs can end up in the same ACE structure, given that they are semantically equivalent. Moreover, all Controlled Languages are strictly linked to English so it's hard to extend them for covering other spoken languages.

This thesis claims that *achieving effective ontology verbalization implies taking into account not only ontology contents but also external information sources*. Moreover, *it's better to define rules for automatic on-the-fly production and understanding of sentences about the ontology using the language employed to fill its elements*. In this way the ontology structure is used twice: as a knowledge base to provide semantics, and as a term repository to provide the lexicon and the base grammar. Rules are purposed to enrich both the lexicon, and the syntactic structure used to produce and/or understand phrases. This approach faces efficiently ambiguity problems, anaphors, and co-references via explicit disambiguation against the names and roles of the ontology components. Trivially, a NLI architecture deigned according to the previous principles is independent from a particular spoken language.

The grammatical setup described above will be denoted as *flexible grammar* throughout the rest of this thesis.

A flexible grammar is not based on static mapping of ontology statements to language fragment. Rather, it devises semantics locating the ontology statements involved in the input request (a user query or a document). Next the necessary syntax and lexicons to express such statements are computed. A verbalization language relying on a flexible grammar is more expressive and closed to natural language than controlled ones.

The flexible grammar proposed in this work computes the semantic seed (that is the SVO triple) for an ontology statement, and creates all possible ways to express its components. The syntactic forms for subject, verb, and object are searched for separately. The statement is verbalized combining the partial structures obtained for the triple components.

2.3.1 Flexible Grammar Settings

In a grammar context *conceptual categories* specify the meaning of the words. In the case of controlled languages conceptual categories are fixed for each statement of the ontology, and require hard manual annotation work. Indeed most conceptual categories are *multi-referential* like in the approach proposed in this thesis. Multi-referential categories may apply to several entities in a non-static form that depends on various conditions like the user's spoken language or the contents to be parsed.

Multi-referential categories can be formally made explicit as predicates that take some arguments. Hence, the meaning of a word w_i becomes $c_i(?x)$; here, a symbol starting with a question mark represents a variable. When the variable is replaced by a constant value referring to an actual entity as in $c_i(e_1)$, the result is the predicate that states that category c_i is true for the entity e_1. Crucially, the introduction of variables allows to represent meanings involving several entities. For example, the following expression can be used to describe scene where there are both an apple, and a table:

$apple(?x) \cap table(?y)$

Predicates using the same variable are applied to the same entity that owns multiple properties simultaneously. The phrase 'the apple is yellow and the table is round' (at the same time) can be expressed as:

$yellow(?x) \cap apple(?x) \cap round(?y) \cap table(?y)$

One of the main functions of a grammar is precisely to express these kind of *linking dependencies* between different components of the meaning of a sentence. For example, English language expresses such dependencies using mainly the word order. As a consequence, meaning-form mapping is not as simple as one might argue. Dependencies force a verbalization system to use complex mapping rules.

Maybe, a rule could connect multiple predicates on the meaning side to a particular words order on the form side.

Moreover, a way for referring to the different parts of meaning and form without having to specify in detail what they should be, is necessary. It doesn't make sense to have a specific construction for a 'yellow apple', and another one for 'red apples'. Form-meaning mappings are needed between dependencies structure, and semantic/syntactic categories. As an example, objects and features should be connected to adjectives and nouns. In short, mappings should enable representing phrases, sub-phrases, syntactic features like numbers, and so on.

Many contemporary linguistic formalisms use *feature structures* or *attribute-value matrices* for this scope. Next, the *unit structures* will be also introduced that are a variant of feature ones. Every part of an utterance will be represented by a unit. Every unit holds information about a word or a sentence.

A unit structure holds several units together, and allows cross-referencing between them. Entries in a not controlled ontology verbalization language will be represented as mappings between abstract unit structures. Typically, a structure will specify a set of semantic features (the *meaning pole*), and the other one will specify a set of syntactic features (the *form pole*). Devising if a mapping between a meaning pole and a form one applies to a particular unit structure will require the unification between one of these poles and the actual structure representing the utterance being processed. The application of the mapping will require a merge operation between the opposite poles of the mapping with the structure. This issue will be deepened in the chapters devoted to Construction Grammar.

2.4 Other Ways for Interfacing to Ontology

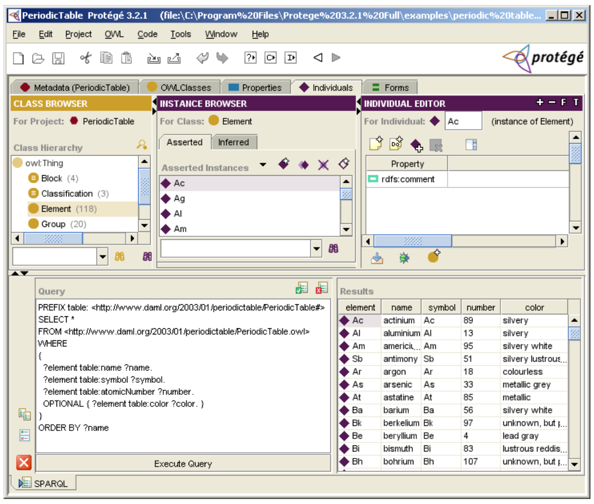
In addition to languages for ontology verbalization evaluated in the previous section, there are many other methods for interfacing to the information from the knowledge representation of a system, that we studied for our experiments when we wanted to evaluate our model as an Intelligent Natural Language Interface. We needed a hearer component that accepts in input the natural language requests of the users and maps them to OWL/RDF ontology of the system; starting from the subgraph that matches the user's query and from its triple components, our system was able to produce correct and expanded answers, retrieving contents not only from the ontology but from external information sources too.

In the first part of this section we illustrate those approaches that aim is to create visual ontology browsing tool for users, and that not have any grammatical background for production or understanding. These tools require basilar training for who wants to use them. In a second section, we illustrate the tools that interpret user's queries and interface them to the OWL ontology, converting the requests in SPARQL queries or directly outputting the correspondents OWL statements. These approaches have a basilar grammatical background and they do not require any training for users which can interface with them in natural language.

2.4.1 Visual Ontology Browser

One of the most popular tools which, among other features, allows browsing ontologies and knowledge bases is *Protégé*[Knublauch et al. (2004)]. For querying, users can use a template-based form where they insert parts of a triple they are interested in, and the missing parts are searched in the semantic repository. If they are found they will be given as a result. Another way to browse an ontology with Protégé is by writing SPARQL queries [Prud'hommeaux and Seaborne (2008)] by special panel (see figure 2.2): results are given in the form of triples. This framework is very useful for experts who are familiar with query languages although they also have to be experienced Protégé users.

At same time TAP system was developed by [Guha and McCool (2011)]. The TAP core has two graphical interfaces for querying ontologies that accept as input the node which is described by an URI, and then return a graph describing the given URI. Search ia a third TAP interface that receives an input string and returns all resources whose title properties contain the string. Title property is specific to the TAP knowledge base. For the same purpose the property *rdfs:label* can be used granting a more widespread approach.



```
PREFIX table: <http://www.daml.org/2003/01/periodictable/PeriodicTable#>
SELECT *
FROM <http://www.daml.org/2003/01/periodictable/PeriodicTable.owl>
WHERE
{
  ?element table:name ?name.
  ?element table:symbol ?symbol.
  ?element table:atomicNumber ?number.
  OPTIONAL { ?element table:color ?color. }
}
ORDER BY ?name
```

element	name	symbol	number	color
◆ Ac	actinium	Ac	89	silvery
◆ Al	aluminium	Al	13	silvery
◆ Am	americiu...	Am	95	silvery white
◆ Sb	antimony	Sb	51	silvery lustrous...
◆ Ar	argon	Ar	18	colourless
◆ As	arsenic	As	33	metallic grey
◆ At	astatine	At	85	metallic
◆ Ba	barium	Ba	56	silvery white
◆ Bk	berkelium	Bk	97	unknown, but p...
◆ Be	beryllium	Be	4	lead gray
◆ Bi	bismuth	Bi	83	lustrous reddis...
◆ Bh	bohrium	Bh	107	unknown, but p...

Figure 2.2: Query panel where user can enter SPARQL query; pressing the Execute Query button the query results will be shown on the right-hand side.

TAP interfaces were tested with RDF files maintained by W3C. In addition, for larger applications dealing with musicians, athletes, places, and so forth, HTML scrapers are built to get the data from the popular sites such as Amazon or All-Music; their Web crawler is dynamically locating and converts relevant pages into machine readable data so that they become available for searching by the TAP interfaces. The knowledge base is dinamically built and contains many millions of triples. While trying to improve traditional search by using an annotations of the search terms,Guha and McCool (2011) describe the following problems:

- Denotation: The biggest problem in the denotation of the concept relevant in the search query is the ambiguity, which is solved by using heuristic rules that prefers some denotations of concept respect others, for example the popularity of a term (Paris as the capital of France preferred to Paris Hilton), the user profile (if she is a ditzy girl, Paris Hilton is preferred to Paris as the Capital of France), and the search context.

- What data to show: that means which data to pull from the semantic web. The node that is the selected denotation of the search term provides a starting point. The next problem is which subgraph around this node to show. A more balanced subgraph is produced by using heuristic rules based on the average branching factor (i.e. bushiness) of the graph around the anchor node.

- How to show selected data: the main problem is the presentation of the resulting data/triples.

Probably due to its visual similarities to common search engines, TAP Search interface has received a lot of attention. The main goal with this interface was to make search engines capable of interpreting the different occurrences of the same input string to different semantic concepts. In Search interface, this problem is solved by asking the user to choose between available options. Four types of question are supported by the TAP for which are handled by 37 patterns that contain the rules of how to handle and answer them. If the input query/question is not recognised as belonging to one of these 37 patterns, the answer is not returned. The question's types are:

1. Searching of entities e.g. *Paris Hilton*
2. Properties of entities e.g. *deducted days in jail for Paris Hilton*
3. Comparison of entities e.g. *girls flightyer than Paris Hilton*
4. Composite queries e.g. *countries with almost an Hilton Palace Hotel*

In comparison to TAP and Protégé users, KIM [Popov et al. (2004)] users are restricted in what they can search for, or they need to be familiar with the underlying ontology structure because this tool simpifies the browsing process providing predefined query templates, where users can construct SeRQL queries using a *form-based interface*. These kind of interfaces are exhaustive for repetitive searches but not for ad hoc queries [Tran et al. (2010)].

Faceted search interfaces are different from these approaches because facets are generated dynamically starting from the user's query, and are not predefined such as in the case of form-based. It has been argued that faceted search browsers are extremely helpful in cases when the user's information need is vague Hyvönen

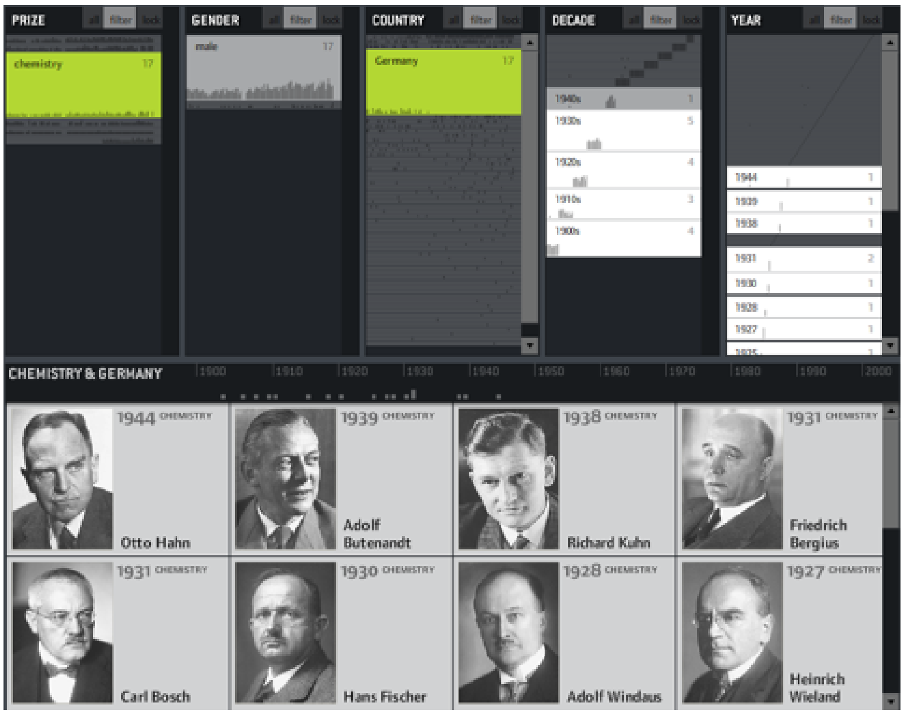


Figure 2.3: Example data based on the Noble prize winners dataset used in the Flamenco facet browser.

et al. (2005). The example of faceted search is displayed in Figure 2.3, that is a demonstration of the*elastic list principle* for browsing multi-facetted data structures. Clicking any number of list entries a query to the database is generated for a combination of the selected attributes. If we create an "impossible" configuration, our selection will be reduced until a match is possible. Elastic lists enhance traditional facet browsing approaches by

- visualizing relative proportions (weights) of metadata values by size;
- visualizing unusualness of a metadata weight by brightness;
- animated filtering transitions.

Even without any intention to search, the user can browse the available categories and explore the knowledge step-by-step. Different approaches can also be combined to introduce a hybrid query which combines a keyword query with the precise structured query.

2.4.2 Natural Language Interfaces

In this section are illustrate those approaches based on linguistics grammars that try natural language queries to an ontology; they are based on manually annotations of syntax and are for some aspects similar to Controlled interfaces as ACE, discussed in the previous chapter. Differently from them but they not fixed syntax for the queries, but they try to understand them interacting with users.

ORAKLE

ORAKEL is an NLI to knowledge bases (Cimiano et al. (2007)) which supports *factual questions*; these kinds of questions start with WH-pronouns such as who, what, where, etc. Factual here means that answers are ground facts as found in the knowledge base, and not complex answers to why or how questions that require explanation.

Respect to other similar systems ORAKEL supports for compositional semantic building, that makes it able to handle questions involving quantification, conjunction and negation; it is the most important advantage of this tool. The answers are generated by a specific ORAKEL component (the *Answer Generator*) that evaluates the query with respect to the knowledge base and presents the answer to the user.

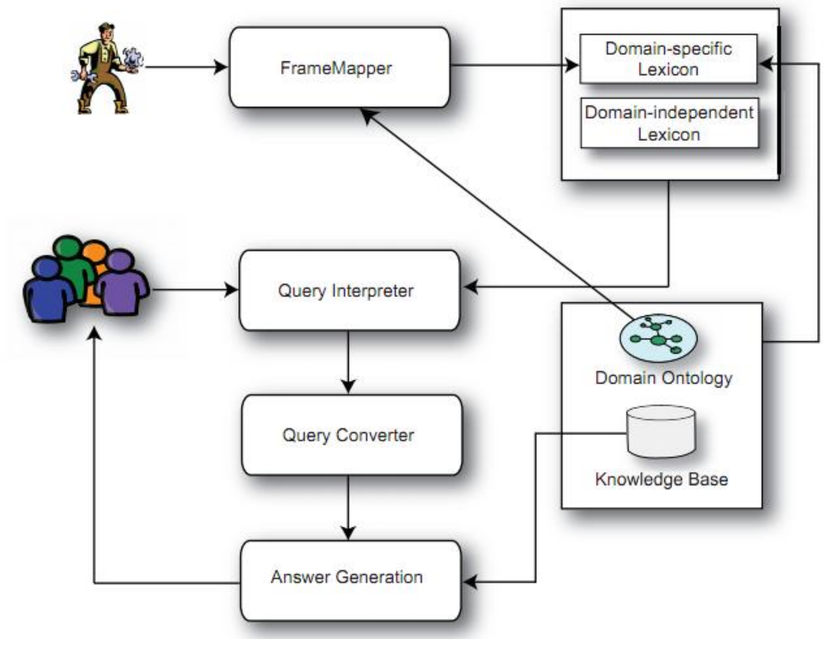


Figure 2.4: ORAKEL system architecture.

ORAKEL's lexicon is composed of two parts:

- *General Lexicon* which is shared among different domains, where are stored words such as what, which, etc...
- *Domain-specific lexicon* composed by two parts: the ontological lexicon generated automatically from the domain ontology (it contains lexical entries and the semantics of instances and concepts which are typically represented by proper nouns and nouns respectively) and the mapping lexicon to match ontology relations with words: this part is created manually and contains mappings of subcategorisation frames to ontology relations.

Subcategorisation frames are essentially linguistic structures with arguments, e.g. verbs with their arguments, nouns with their arguments, etc.
Subcategorisation frames are built by the domain designer who are not expert in computational linguistics, although they are expected to have some very basic knowledge of subcategorisation frames. The coverage of the lexicon for frames is being created during user interaction sessions by *Query Interpreter* and *Query Converter* module, illustrated in figure 2.4; in several iterative cycles, the adaptation is performed increasly, so lexicon is customized to user. The customisation system of ORAKEL is designed so that in each iteration, the created lexicon is extended and therefore the system is expected to give better performance. Consequently, the more time users spend customising the system, the better the performance of the system is expected to be.

One weak point of the approach implemented in ORAKEL is that it maps ontology relations to words. This approach assumes that all classes and instances have understandable and useful lexicalisations from users, which is not always true. Moreover, while the user interaction is used for customisation, with regard to the end users, the system either interprets the question and returns the answer, or it fails. Hence, the end-user has no control over the overall process of interpreting the NL into the formal language.

PANTO

PANTO [Wang et al. (2007)] is a portable NLI to ontologies. It accepts generic natural language queries and outputs SPARQL queries (Prud'hommeaux and Seaborne (2008)). Based on a special consideration on nominal phrases, it adopts the triple-based data model to interpret the parse trees output. This triple model is composed only by relations between classes or instances in the ontology, and does not consider other OWL statements that specific the type of relations as Simmetric Property and so on. Using PANTO it is possible to express queries in natural language even without considering the syntax of RDF or OWL, the formal query language, or the schema and vocabulary of ontologies. To help make sense of the words in the NL queries and map them to the entities (concepts, instances or relations) in the ontology, existing tools such as WordNet (Fellbaum (1998)) are used to extract synonymous. PANTO extracts nominal phrases in the parse trees of the query (which is obtained applying Stanford Parser (Spitkovsky et al. (2011)) to the query), and couples them to form the *QueryTriples* intermediate representation. Then, by utilizing knowledge in the ontology, PANTO maps QueryTriples to *OntoTriples* which are represented with entities in the ontology. Finally, together with targets and modifiers extracted from the parse trees, OntoTriples are traduced as SPARQL.

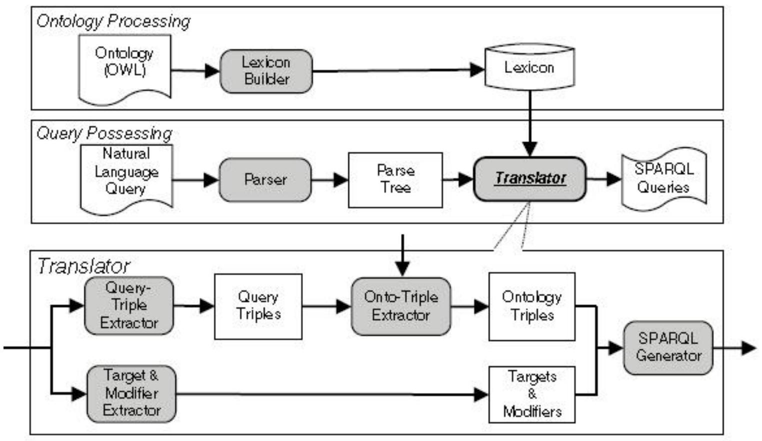


Figure 2.5: PANTO overview.

According to Wang et al. (2007), there is no specification for what types of questions are supported. WordNet is used for the vocabulary extension, and the user lexicon is configurable - there is no need to manually customise the system unless the user is interested in adding associations to the ontology resources in order to improve the system's performance.

Chapter 3

Construction Grammar and Fluidity

3.1 Introduction

Human languages are inferential communication systems gives them a number of special properties. Between them, main property is that languages can be openended: at any moment the set of available conceptualizations and linguistic conventions can be expanded by speakers if they need to express something that was not yet conventionally expressible in the language, because hearers are assumed to be intelligent enough to figure out what was meant and possibly adopt any innovations introduced by speakers.

This *fluid* character of human language helps to make them adaptive to the needs of language users that keep changing as human societies evolve and become more complex. One of the explicit goals of Fluid Construction Grammar is to try and deal with fluidity of the languages, confering Construction Grammar this property.
This topic is discussed more extensively in Steels and van Trijp (2011).

3.2 Construction Grammar

Construction Grammar (CxG) is an approach for studying linguistic structure first proposed in [Fillmore (1985), Fillmore et al. (1988), Lakoff (1990)] that shares certain assumptions both with formal liguistic theories (as Jackendoff (1990)) and with cognitive one, that is the study of the language in its *cognitive function*, where cognitive refers to the crucial role of intermediate informational structures in our encounters with the world.

Construction grammar theories consider *constructions* as the basic units of language. While what makes up a construction has been different for the different theories of construction grammar, it is generally approved that a construction is a *"syntactic pattern which is assigned one or more conventional functions in a language, together with whatever is linguistically conventionalized about its contribution to the meaning or use of the structures containing it"* (Fillmore et al. (1988) pag. 36). Adapting to changing language patterns easily, constructions consider both semantics and syntactics of lexicon and are easier to manage than words as the atomic unit. For this reason, costructions can be semantically computed and it allowes to integration of constructions into bigger collections.

Different construction definitions can only be posited if there is something about the form or meaning of the construction itself that is not given from ordinary compositional processes, from the literal meaning, from the processes of conversational reasoning, or from other constructions that exist in the language (Kay and Fillmore (1999)).
What makes a construction a construction is that it possesses *"properties of form (syntactic and phonological) and meaning (semantic and pragmatic)"* (Croft and Cruse (2004) pag. 256).

3.2.1 Linguistic Requirements

The linguistic perspective of CxG is in the general line of cognitive grammar (Langacker (1999)) and more specifically construction grammar (Goldberg (1995)). This means the following (Steels and Beule (2006)):

- *CxG is usage-based.* It means that words available to speakers and hearers consist of patterns which can be highly specialized, perhaps pertaining to a single case, or much more abstract, covering a wide range of events. New sentences are constructed or parsed by assembling this patterns using the unify and merge operators (defined later in this chapter.)

- *The grammar and lexicon are modeled by symbolic units.* A symbolic unit associates aspects of semantics with aspects of syntax. They feature a semantic pole and a syntactic pole. This is are the specified patterns in the previous step; they may be bi-directional, and so are usable both for production and for parsing (as in the case of Fluid Construction Grammar) or be uni-directional (such as Embodied Construction Grammar).

- *There is a continuum between grammar and the lexicon.* Not only can templates be at different levels of abstraction, but there is also no formal differences between the structures of lexical and grammatical entries. In the case of lexical entries, the syntactic pole tends to be a lexical stem and the semantic pole covers some concrete predicate-argument structure.
  In the case of grammatical constructions, the syntactic pole contains various syntactic categories that constrain the sentence, and the semantic pole is based on semantic categories, but otherwise there is no formal difference between the two types of templates.

- *Syntagmatic and Paradigmatic Compositionality.* To produce or parse a phrase, templates can be combined (several templates all matching with different parts of the meaning in production or with parts of the sentence in parsing are simply applied together) or integrated (using hierarchical templates that combine partial structures into larger ones).
  A part from this syntagmatic composition, there is also a paradigmatic compositionality, that means the possibility that several templates are covered and each contribute additional constraints to the final sentence. Both forms of compositionality are completely supported with the unify and merge operators defined later.

- *Schematization occurs through variables and categorization.* A template has the same form as an association between a semantic structure and a syntactic structure, in other words both poles of a template are *feature structures*. However, templates are more abstract (or schematic) in because variables are used instead of units and values, and syntactic or semantic categories are introduced to restrict the possible values of the semantic and syntactic pole. These categories are often established by syntactic or semantic categorization rules.

3.2.2 Generative Grammar vs Construction Grammar

Traditional generative grammarians have dealt primarily with ideas which they consider to be "core" and have virtually ignored what they considered "peripheral". The periphery holds idiosyncratic phenomena such as idioms that are stored in the lexicon and which cannot undergo the rules or transformations of generative. Also, this idiom cannot undergo the passive transformation. Since this structure cannot be understood on the basis of its parts nor can it undergo most transformations, it must be stored in the lexicon as a whole idiom.

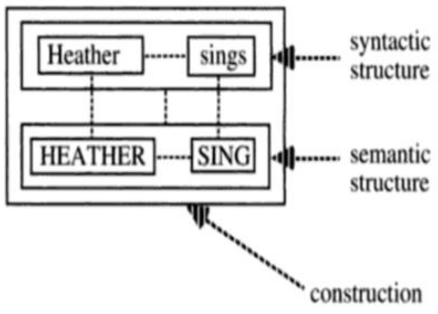
Construction grammarians, however, are interested in peripheral phenomena because they believe that "*fundamental insights can be gained from considering such non-core cases, in that the theoretical machinery that accounts for non-core cases can be used to account for core cases*" (Goldberg (1995)). What is in the core is more regular, occurs with more frequency and thus, should be easier to learn. Since what is in the periphery is more difficult and must be "*learned inductively on the basis of the input, constructionists point out that there is no reason to assume that the more general, regular, frequent cases cannot possibly be*" (Goldberg (2006): 14). Furthermore, regular patterns historically evolve from less regular ones. Consequently, the theoretical machinery required to handle the sources of the irregular patterns could be used to account for the regular patterns as well.

Construction grammar is generative in the sense that it attempts to account for the infinite number of possible grammatical expressions of a language while also attempting to account for the infinite number of disallowed expressions of said language (Goldberg 1995). But, unlike generative grammar, construction grammar is monostratal; it is not transformational, meaning that one sentence is not derived from the deep structure of another. There are "*no underlying levels of syntax, nor any phonologically empty elements*" (Goldberg (2006):10). Rather than deriving one sentence from another, the two sentences may simply represent different construals of the same situation (Langacker (2002)).

Construction grammar theories also differ from generative theories in that construction grammar "*emphasize[s] that languages are learned - that they are constructed on the basis of the input together with general cognitive, pragmatic, and processing constraints*" (Goldberg (2006): 3) while most generative theories believe that humans have an innate sense of language. Constructionist theories claim that, "*language learners bring to the task a host of pragmatic and cognitive abilities which they employ to great effect in the task of language learning*" (Goldberg and Casenhiser (2006):5).
When learning a language, human must learn more than just what grammatical class a word belongs to. One must also notice the subtleties in meaning that make some sentences grammatical and others ungrammatical when using words of the same grammatical class. To do so, the learner must combine his knowledge of syntax with his knowledge of semantics and pragmatics.

Constructionists are also interested in "*accounting for the conditions under which a given construction can be used felicitously, since this is taken to be part of speakers' competence or knowledge of language*" (Goldberg (1995):6) while generativists "*held that the nature of language can best be revealed by studying formal structures independently of their semantic or discourse functions*" (Goldberg (2006):4). Understanding when a construction can be used felicitously involves both pragmatic and semantic factors which generativists have compartmentalized as completely separate from the syntax of a sentence. Thus, constructionists are interested in how the different aspects of linguistics work together to form grammatical and discourse appropriate sentences while generativists study each aspect independent of all the others.

In figure 3.1 are shown the different representations of the two grammatical theories discussed in this section, that have in common the distinction in the sentence



(a) Construction Grammar

$[\ [\text{Heather}]_{NP}\ [\text{sings}]_{VP}]_{S}$

(b) Generative Grammar

Figure 3.1: Semplified CxG andGenerative Grammar representations of the sentence *Heater sings*

structure of Subject and Predicate components. In the Construction Grammar theory however links between semantics and syntactic structure are represented.

3.3 Fluid Construction Grammar

In recent year only very few demonstrations on non-trivial grammars are considered in grounded artificial production and understanding. Part of the problem of grammar emergence for artificial verbalization is that a grammar is more encompassed of lexicon. Studies and experiments on grammars require powerful techniques from symbolic processing; existent formalisms are not rather strongly linked to one or the other linguistic theory, as happened for example for the Head Driven Phrase Structure Grammar (Hpsg [Pollard and Sag (1994)).

Hpsg is a kind of dependency grammar and it is centered around the *head* of a phrase that is linked through the *head-dependent* relations to other roles of the words in the same phrase. Roles can be among *modifiers*, *specifiers* and *complements*. Althought this theory is very useful and the roles computing is simply, not every linguistic phenomenons can be defined in terms of them and there is not general consensus for these settings. The primary goal of Hpsg is to build a theory of the knowledge embodied in the human brain, and not to build agents that use this knowledge (Pollard (1997)).

Most other formalisms basically find a minimal but necessary set of grammatical rules and principles such that *empirical linguistic data* satisfy the grammar: questions of how and why such linguistic data could be produced, learned and evolve are not considered. These formalisms control semantics and synactic categories for their purposes, closing them.

In order to overcome these limitations, the Artificial Intelligence Lab of the University of Brussels in the person of Luc Steels together with people at Sony CSL in Paris have for many years developed a formalism that can handle both production and parsing and that would be adequate to study natural language grammars: the result of these efforts was the designe of a framework named Fluid Construction Grammar (FCG).
It is the first available operationalization of Construction Grammar that uses many existing and widely accepted notions in theoretical and computational linguistics, as the *feature structures* that represents syntactic and semantic information during parsing and production, and *abstract templates* for the representation of lexical and grammatical usage patterns, as in Ivan A. Sag and Bender (2003) or Bergen and Chang (2003).

These properties make the framework appropriate for our experiments, because we build constructions from semantics deduced from an ontology coupled with computable syntax by our specific rules. Also, allowing production and parsing throught the same constructions, we can verbalize about the ontology and understand related free text by them. FCG is based on general operations of *unification* and *merging*. Differently from other formalisms, FCG attempts the investigation of the origins and evolution of semantics and syntactic categories considering them free and it is more concerned with things like flexibility, learning, invention, usage and other creative aspects of language. Freely categories is the strength feature of the framework.

3.3.1 FCG Linguistic Feautures

Basing on Construction Grammar, the FCG mantains the general features of the cognitive grammars in general, that we have shown in the previuos chapter. FCG is usage-based, and it is based on syntagmatic and paradigmatic compositionality. Also it uses symbolic units to model grammar and lexicon and there is not differences between template for grammatical constructions and lexical ones; it grants a continuum between constructions themselves. Finally it is characterized by schematization.

However, being a formalism to model emergent natural language-like grammar, FCG has a specifics characteristic, that is the *freely definable semantics and syntactic categories* for constructions; often linguistic formalisms set these categories and it can be a limitation when the modeling of grammars may be depend to knowledge representations, that is our case.

This openness of categories is in line with the Radical Construction Grammar approach which argues that linguistic categories are not universal and subject to evolution (Croft (1991)).
Next we show the basic concepts of FCG.

3.3.2 Feauture Structures

As mentioned at the end of the previous chapter, unit structures hold the information about the utterance to process. They are represented as a list of units. For instance, let consider the sentence "Paul hates Janet".
As a first step it is necessary to build a unit for each word in the sentence, and one additional super-unit called Top-unit to keep the other three units together. Units are represented as *lisp like lists* (see e.g. Steele (1990)).) for which the entire list is delineated by plain brackets as:

```
(Janet-unit (form ((String "Janet")))).
```

The above expression is a unit. It is a list where first element is the `unit's name`, in the example the symbol Janet-unit, and second element is a list of type `(form ((String "Janet")))`, which is the only `feature` of this unit. The feature's name again is the symbol `form` and its value is the list `((String "Janet"))` and so on.

Generally, units will be represented as lists, with the first element that represents the unit's name and all remaining elements its features. Features will also be represented as lists, again with thefirst element as its name and a value as second element. The name must always remain the first element in the list; unit structures will be lists of units. Hence, in the example, there is a unit structure that contains a unit for each word in the phrase "Paul heats Janet" and one additional super-unit (called *Top-unit*) to keep together the other three units. They look like:

```
((Top-unit (syn-subunits (Paul-unit Janet-unit Heat-unit))
(form ((meets Paul-unit Heat-unit)
(meets Heat-unit Janet-unit))))
(Paul-unit (form ((string "Paul"))))
(Janet-unit (form ((string "Janet"))))
(Heat-unit (form ((string "heats")
(stem "heat")))
(syn-cat ((lex-cat verb)
(number sing)
(person 3rd)))))
```

Many other linguistic formalisms (e.g. hpsg and ecg) represent feature structures with a boxed notation or as attribute value matrices instead of with the bracketed lisp-like notation shown here. In such a notation the above unit structure for the sentence "Paul heats Janet" could be represented as:

$$\begin{bmatrix} \text{Top-unit} & \\ \text{FORM} & \left\langle \text{meets}(\boxed{1},\boxed{2}),\ \text{meets}(\boxed{2},\boxed{3}) \right\rangle \\ \text{SYN-SUBUNITS} & \left\langle \begin{array}{l} \boxed{1}\begin{bmatrix}\text{Paul-unit} & \\ \text{FORM} & \langle\text{STRING "Paul"}\rangle\end{bmatrix} \\ \boxed{2}\begin{bmatrix}\text{Heat-unit} & \\ \text{FORM} & \langle\text{STRING "heats"}\rangle\end{bmatrix} \\ \boxed{3}\begin{bmatrix}\text{Janet-unit} & \\ \text{FORM} & \langle\text{STRING "Janet"}\rangle\end{bmatrix} \end{array}\right\rangle \end{bmatrix}$$

In this notation lists are typically delineated with hooked brackets (like <this>.) Both representations are more or less similar. Whatever the notation used, a unit structure can easily be extended.
In FCG, semantic and syntactic information are kept in different unit structures. Syntactic units normally contain the features *syn-subunits*, *form* and *syn-cat*. Semantic units typically have the features *sem-subunits*, *meaning* and *sem-cat*. The sem-cat feature describes information about the semantic category of the unit, as for example if it is an object or a person. The fact that semantic and syntactic information is kept in different structures reflects that constructions in language are meaning-form mappings.

So a lexical construction for "Paul" is a mapping between a syntactic pattern selecting for the string "Paul" in the form feature of a syntactic unit, and a semantic pattern introducing the predicate 'Paul (?x)' in the meaning feature of the correspondent semantic unit.

3.3.3 FCG Template

FCG templates arises how the constructional 'pattern' should be modeled. Generally they are formulated as partially specified unit-structures as:

`((?unit (form ((string "Paul")))))`
for the syntactic pole of "Paul" construction and:
`((?unit (meaning (((Paul ?x))))))`
for the semantic pole. These again look like unit structures containing only one unit but with a variable name (as we shown in chapter 2 any symbol starting with a question mark is a variable), reflecting the fact that the construction shouldn't care about the name of the unit it selects for. Unit structures that contain variables like this are called *unit structure templates* or *short templates.* They actually specify a set of unit structures.

3.3.4 Unification and Merging

The operation that decides whether a template matches a specific unit structure is called *unification* between the template and the unit structure. This operation allows that the unit structure contains more units than specified by the template. The result of unifying a template with a particular unit structure is a set of sets of bindings of variables to actual values.
For example, the unification of the template:

```
((?unit (form ((string "Paul")))))
```

with the unit structure

```
((Mary-unit (form ((string "Paul"))))
```

is a set containing one set of bindings:

[?unit/Paul-unit].
This can also be represented as:
(((?unit . Paul-unit))).

A set of bindings specifies how to make a structure from a template: by substituting the variables in the template by the values they are bound too. It also allows a particular unit in the template to contain more features than specified. The operation is also insensitive to the order of the units in the structure or of the features in a unit. Considering structure templates, we can proceed with specifying constructions as mappings between a semantic and a syntactic template as in:

```
((?unit (meaning ((Paul ?x)))))
<-->
((?unit (form ((string "Paul")))))
```

Generally, the semantic pole is written above the double arrow and the syntactic pole below it. The unification of the syntactic pole with a unit structure results in the set of bindings and of substitution of these bindings in the semantic pole. The operation that takes a unit structure and forms a new extended structure as specified by a template is called merging.

While parsing an utterance, the right (syntactic) pole of a construction is unified with the syntactic structure to see whether it applies. If it does, the left (semantic pole) is merged with the initially empty semantic structure to yield a new semantic structure. While producing a sentence the semantic pole is unified with the semantic structure and, if successful, the syntactic pole is then merged with the initially empty syntactic structure to yield a new syntactic structure.

Summarily, with FCG the information about an utterance are represented with unit structures. Because it is always possible to add units to a structure, or features to a unit or values to a feature, this representation is powerful. We have also set first steps towards representing rules of language as bi-directional mappings between structure templates. These mappings can be used both for parsing forms and for producing them.

3.3.5 The J-operator and others

Many other aspects, in particular those that require manipulation of the hierarchical structure of unit structures, require more powerful operations than the basic unification and merging of unit structures. To that end, the author extended FCG with the so called *J-operator*.

Both the left-pole and the right-pole can be marked with the J-operator (Beule and Steels (2005)). In this way, the semantic pole of constructions (lexical or grammatical) can decompose the meaning to be expressed (which originally resides in the top node of the semantic structure) and the syntactic pole can group units together into a larger pole.

Summarily J-operator has three arguments: a *daughter-unit*, a *parent-unit* and a set of *pending-subunits*. Units marked with the J-operator are ignored during unification. When the construction applies, a new unit is introduced and mapped to the first argument of the J-operator. The second argument should already have been bound by the unification process to the parent unit from which the new unit should depend. The third argument specifies the set of units that will be pulled into the newly created unit. The new unit can contain additional slot specifications, specified in the normal way, and all variable bindings resulting from the unification are still valid.

Briefly when producing, the conditional units of the semantic pole of a construction are matched against their correspondents in the transient structure, but the J-units are ignored. If a match succeeds, the J-units of the semantic pole are merged with the semantic pole of the transient structure, followed by the syntactic pole of the construction merging with the syntactic pole of the transient structure (both the J-units and the conditional units). In parsing, the conditional units of the syntactic pole of a construction are matched against their correspondents in the syntactic pole of the transient structure, but the Junits are ignored. If a match succeeds, the J-units of the syntactic pole are merged with the syntactic pole of the transient structure, and all units of the semantic pole of the construction are merged with the semantic pole of the transient structure.

FCG uses other special operators. These can be regarded as special directives informing the unification and merging engine that something special needs to be performed. An example operator that is used frequently is the *includes operator* "==": a list of which the first element is this operator unifies with all lists that at least contain the other elements in the includes list. For example, the includes list
(== a c b)
unifies with all of the lists below:
(a c b),
(a b c),

(a c b d e),
(e c d c b a)
and so on. As can be seen, the order in which the elements appear is of no importance, only the fact that they are.

The use of operators like the includes operator is the reason why FCG unification may return multiple results. For example, unifying the list
(== ?x)
with the list
(== a b c)
results in the following set of three solutions:
[?x/a],
[?x/b],
[?x/c]

The *uniquely-includes operator* "=1" specifies that each of the elements should occur in the target, which may still include more elements, but that there should only be one value for the same feature. Again the ordering of the elements no longer matters. This information not only helps the matcher by avoiding consideration of unnecessary additional hypotheses, it also impacts merging, because without this operator the additional category-value pair would simply be added even if another value already is present.

Chapter 4

Model Description

4.1 Introduction

This chapter details the proposed formalization aimed to incorporate RDFs/OWL standard ontology into FCG for obtaining an expanded natural language interface. The NLI devised in this work is expanded because it produces answers by retrieving data not only inside the ontology but also from external unstructured information sources that are related to the topic dealt with. The main external information source used in the implementation is the Web.

The main idea is to build lexical and grammatical constructions starting from the ontology statements, and to use them in the FCG framework for both producing and parsing utterances related to represented domain. Such constructions are integrated with those describing the grammar of the spoken language under investigation to increase the production understandability.

A set of rules inferring syntactic patterns for single words or sentences using WordNet and FrameNet is used to overcome the limited expressivity of the syntactic poles in the ontology constructions. In turn, new patterns can be matched with the semantic pole of a construction. In this way, synonyms are defined for a concept that are represented as a set of constructions with the same semantic pole.

4.2 The Ontology Model

This section presents the model used for defining ontologies, and resources. Rules for computing constructions will be defined using this model. For the purposes of this work, the ontology O is a tuple:

$$O :< C_S, I_S, PD_S, PO_S, T_S, L_S, S_{PD}, S_{PO} >$$

Where:

- C_S is the set of classes;
- I_S is the set of individuals;
- PD_S is the set of datatype properties that enables the following mappings:

  $C_i \rightarrow T_i$

  $I_i \rightarrow T_i$

  where $C_i \subset C_S, T_i \subset T_S$ and $I_i \subset I_S$;
- PO_S is the set of object properties that enables the following mappings:

  $C_i \rightarrow C_j$

  $C_i \rightarrow I_i$

  $I_i \rightarrow I_j$

  where $C_i, C_j \subset C_S$ and $I_i, I_j \subset I_S$;
- T_S is the set of literal datatypes;
- L_S is the set of literal strings used in the ontology as values for T_i, where $T_i \subset T_S$;
- S_{PD} is the set of datatype statements, where each datatype statement is a triple from the set:

$$< C_S, I_S > \times PD_S \times < L_S >$$

- S_{PO} is the set of object statements, where each object statement is a triple from the set:

$$< C_S, I_S > \times PO_S \times < C_S, I_S >$$

Triples can be represented in the form $T(s, p, o)$ for both object and datatype statements. In a triple $T(s, p, o)$, s is the subject, p is the property, and o is the object.

An *ontological resource* OR contains the elements that can be selected using URIs, hence:

$$OR :< C_S, I_S, PD_S, PO_S >$$

The elements in OR define the formal vocabulary of the ontology.

We assume that:

- Each elements in OR corresponds to an atomic lexical construction;
- Each element in $S_{PO} \cup S_{PD}$ corresponds to a grammatical construction;
- An OWL sub-graph is a taxonomy of elements of $S_{PO} \cup S_{PD}$, so it is a taxonomy of constructions, and it can be regarded as a (non atomic) construction;
- An OWL ontology conveys also semantic information, and it can be used to express a (partial) construction grammar related to the domain.

The problem of building a NLI for the domain at hand can be regarded as generating all possible lexical and grammatical constructions for each element $e_i \in OR$. Let LC be the set of lexical constructions computed from the ontology, while GC is the set of grammatical ones. Generating a grammar to verbalize the ontology statements can be modeled through three mapping functions:

$l : OR \rightarrow LC$

$\vec{g} : (S_{PO} \cup S_{PD}) \cap OR \rightarrow GC$

$\vec{v} : GC \cap LC \cap G \rightarrow V$

Here, l applies rules that compute the set $LC_{e_i} \subset LC$ of all the lexical constructions (including synonyms) for each term $e_i \subset OR$.
$g_i \in \vec{g}$ computes the set $GC_{e_i} \subset GC$ of all the grammatical constructions for the statements containing e_i.
$v_i \in \vec{v}$ creates the proper combination of lexical construction and grammatical ones (including constructions for the language grammar G) to obtain a verbalization representative of e_i and of its relations in the model.

The following sections deal with computing constructions from OWL ontologies along with their use by an automatic system able to both speak and answer queries about the represented domain.

4.3 Statements, Topics and Scenes

The majority of the approaches based on construction grammar look for the *skeletal meaning* of an utterance. Skeletal meaning is a representation of the meaning as a “skeleton” where each component is connected with the other ones. For example, a skeletal meaning structure related to the statement *the dog eats the cat*, and expressed using first-order logics (FOL) predicates looks as:

```
(eating-event ?eat)
(eater ?eat ?dog)
(eated ?eat ?cat)
```

Grammatical constructions related to an utterance depend on its skeletal meaning because it states which components are required by the constructions themselves; since RDF statements in the OWL graph are arranged in the form of skeletal meanings, one can argue that they can be simply translated in a grammatical construction.

As already stated, this work is focused mainly on the extraction of the semantic seed from ontology statements in terms of a SVO triple that is a transitive sentence. Transitive sentences can be described by the following Backus-Naur Form (BNF) using a simplified Penn Treebank II notation:

$S \rightarrow NP \;\; V \;\; NP$

$NP \rightarrow JJ \;\; N$

Here, S is the whole sentence, while NP is a Noun Phrase. The V and N tags indicate verbs and nouns respectively, regardless of their morphology, and incorporate both prepositions (P) and articles (DT). Similarly, JJ are adjectives regardless of their gradability.

The structure described above represents the target skeletal meaning of the present work. In what follows, the steps for building constructions aimed to perform understanding and verbalization in terms of such a skeletal meaning are reported.

The first step is a suitable axiomatization of the RDF language to describe the meaning of the ontology elements. The basic predicate used in the axiomatization will be a unary predicate in the form(X $?x$). In the following, the description of ontology components is reported. In particular:

- classes are described by predicates (C_i $?c$) $\forall C_i \in C_S, i = 1...n_c$;
- instances are described by predicates (I_i $?i$) $\forall I_i \in I_S, i = 1...n_i$;
- both object and datatype properties are described by predicates (P_i $?s$ $?p$ $?o$) $\forall P_i \in PO_S \cup PD_S, i = 1...n_p$ where $?p$ identifies the property itself, while $?s$ and $?o$ identify respectively the subject and the object of the property that is its domain and range.

A particular family of predicates is used to model the datatype values belonging to T_S. These values are the XSD datatypes [w3C]. The corresponding FOL predicates have the form $(E_i\ ?e), i = 1, \dots, D$ where D is the size of the XSD datatypes list.

Given the form of each possible predicate, a *topic* is the meaning of whatever ontology statement where the term "meaning" refers to the interpretation of the structure of the statement. As an example, the predicate $(C_1\ ?c)$ means "the variable $?c$ is the class C_1 in the ontology". In this way different classes, properties, instances, and values are described along with their interconnections.

Topics enable the creation of abstract templates in the FCG framework that are not related to a particular domain. Such templates can be extracted automatically from the analysis of the ontology structure, and will form the `meaning` units in the semantic poles of the constructions describing the domain. Actual names of the ontology elements will form the `form` counterparts to be embedded in the corresponding syntactic poles. As already stated, names of the ontology elements are the components of the base lexicon used in understanding and/or production tasks so they cannot express semantics directly. Rather, semantic roles are coded by the ontology structure.

A topic described by the triple $T(s, p, o)$ consists of an event relation p connecting the subject s to the object o. Such description is generated according to the following rules:

$$T(s, p, o) \to P(s, p, o)R(s)O(o)$$

$$P(s, p, o) \to (P_i\ ?s\ ?p\ ?o), i \in \{1, \dots, n_p\}$$

$$R(x) \to C(x)|I(x)$$

$$O(x) \to R(x)|E(x)$$

$$C(x) \to (C_i\ ?x), i \in \{1, \dots, n_c\}$$

$$I(x) \to (I_i\ ?x), i \in \{1, \dots, n_i\}$$

$$E(x) \to (E_i\ ?x), i \in \{1, \dots, D\}$$

A topic description is always connected: all predicates are linked through their arguments. As an example, the expression $(C_1\ a)(C_3\ b)$ can never describe a topic because the two predicates are not linked that is they cannot express a structure inside the ontology. Such an expression states only that the ontology contains the classes C_1 and C_3 separately. There is no information about the role they play in the ontology. A correct topic is obtained by adding a property predicate: $(P_2\ a\ p\ b)(C_1\ a)(C_3\ b)$. As a consequence, a topic description with k predicates has exactly $k-1$ variable equalities as part of its interpretation.

4.3.1 Building Topics from OWL

According to the axiomatization reported before, the topics in a given OWL subgraph are:

- each class in the subgraph;
- each individual in the subgraph;
- each object relation between classes;
- each object relation between instances;
- each `is-a` relation between individuals and classes;
- each datatype relation between a class and a datatype value;
- each datatype relation between an individual and a datatype value.

Given the subgraph in figure 4.1, the following topics can be extracted:

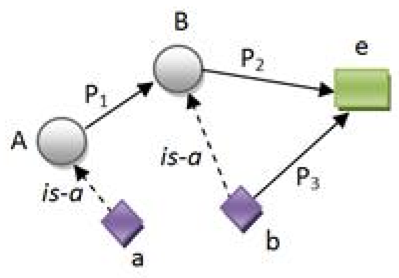


Figure 4.1: A small ontology subgraph.

$(C_1\ A)$,
$(C_2\ B)$,
$(I_1\ a)$,
$(I_2\ b)$,
$(E_1\ e)$,
$(P_1\ A\ p_1\ B)(C_1\ A)(C_2\ B)$,
$(P_2\ b\ \text{is-a}\ B)(I_2\ b)(C_2\ B)$,
$(P_3\ a\ \text{is-a}\ A)(I_1\ a)(C_1\ A)$,
$(P_4\ b\ p_2\ e)(I_1\ b)(E_1\ e)$,
$(P_5\ B\ p_5\ e)(C_2\ B)(E_1\ e)$.

Again, the reader should note that there is no description containing just two predicates: topics are described by either facts or triples.

The following rules for mapping OWL structures onto predicates hold.

1. *Class definitions* map onto $(C_i\ ?x)$ predicates for lexical constructions.

2. *Individual definitions* map onto either $(I_i\ ?s)$ predicates for lexical constructions or $(P_i\ ?s\ ?p\ ?o)(I_j\ ?s)(C_k\ ?o)$ predicates for grammatical ones.

3. *Subclass definitions* map onto $(P_i\ ?s\ ?p\ ?o)(C_j\ ?s)(C_k\ ?o)$ predicates for grammatical constructions.

4. Given an *object property definition*, if both the property range and the do-

main are classes then it maps onto a $(P_i\ ?s\ ?p\ ?o)(C_j\ ?s)(C_k\ ?o)$ predicate for grammatical constructions, otherwise (individuals) it maps onto a $(P_i\ ?s\ ?p\ ?o)(I_j\ ?s)(I_k\ ?o)$ predicate.

5. Given a *datatype property definition*, if the property domain is a class then it maps onto a $(P_i\ ?s\ ?p\ ?o)(C_j\ ?s)(E_k\ ?o)$ predicate for grammatical constructions, otherwise (individual) it maps onto a $(P_i\ ?s\ ?p\ ?o)(I_j\ ?s)(E_k\ ?o)$ predicate.

4.4 Defining Semantic Categories

Both semantic and syntactic categories are needed to allow the definition of abstract constructions. This section will focus on semantic categories, while syntactic ones will be covered in the next section. As reported in previous sections, semantics is coded by the roles of the various ontology elements. As an example, the semantic pole identifying a datatype property looks as follows:

```
((?Top (sem-subunit
   (== (?property-unit ?concept-unit ?datatype-unit))))
   (property-unit
      (sem-cat (== (property ?s ?p ?o)))
   (concept-unit
      (sem-cat (== (concept ?s))))
   (datatype-unit
      (sem-cat (== (feature ?o))))))
```

This template does not select a particular triple made exactly by the P_i, C_j and E_k elements. The `property`, `concept` and `feauture` semantic categories are used instead to make the pole more general and applicable to all possible property-concept-feature combinations. A complete set of semantic categories have been devised for each possible topic.

The semantic category of a single predicate is simply its type:

- `(concept x)` for any of the *(C_i x)* predicates,
- `(individual x)` for any of the *(I_i x)* predicates,
- `(feature x)` for any of the *(E_i x)* predicates and
- `(property s p o)` for any of the *(P_i s p o)* predicates.

Furthermore, the semantic category of a conjunction of predicates is determined according to the following rules:

- The conjunction `(concept s)(concept o)(property s p o)` is defined as `(object-statement)` ,
- The conjunction `(individual s)(concept o)(property s p o)` is defined as `(is-a-statement)` ,
- The conjunction `(concept s)(feature o)(property s p o)` is defined as `(datatype-statement)` ,
- The conjunction `(individual s)(individual o)(property s p o)` is defined as `(object-statement)`,
- The conjunction `(individual s)(feature o)(property s p o)` is defined as `(datatype-statement)`.

A summary table of the correspondences outlined above is reported.

Topic	Semantic Category
(C_i ?x)	`(concept ?x)`
(I_i ?x)	`(individual ?x)`
(E_i ?x)	`(feature ?x)`
(P_i ?s ?p ?o)	`(property ?s ?p ?o)`
(P_i ?s ?p ?o)(C_i ?s)(C_j ?o)	`(object-statement ?s ?p ?o)`
(P_i ?s ?p ?o)(I_i ?s)(C_j ?o)	`(is-a-statement ?s ?p ?o)`
(P_i ?s ?p ?o)(C_i ?s)(E_j ?o)	`(datatype-statement ?s ?p ?o)`
(P_i ?s ?p ?o)(I_i ?s)(I_j ?o)	`(object-statement ?s ?p ?o)`
(P_i ?s ?p ?o)(I_i ?s)(E_j ?o)	`(datatype-statement ?s ?p ?o)`

4.5 Defining Syntactic Categories

Syntactic categories have been defined as the counterparts of the semantic ones in order to produce and/or parse statements according to the SVO structure. Syntactic categories obey to the set of production rules stated in the following.

$(T\ ?e\ ?a\ ?p) \rightarrow (NP\ ?a)(Verb\ ?e\ ?a\ ?p)(NP\ ?p)$

$(NP\ ?a) \rightarrow (Adjective\ ?a)(NP\ ?a)$

$(NP\ ?a) \rightarrow (Noun\ ?a)$

Moreover, the $(Datatype\ ?s\ ?p\ ?o)$ syntactic category has been defined purposely to model datatype values.

Syntactic categories map onto semantic ones according to the following correspondences:

1. The `(concept ?x)` and `(individual ?x)` semantic categories map onto the $(NP\ ?x)$ syntactic one.
2. The `(feature ?x)` semantic category maps onto the $(NP\ ?x)$ syntactic one.
3. The `(is-a-statement ?s ?p ?o)` semantic category maps onto the $(Verb\ ?s\ ?p\ ?o)$ syntactic one.
4. The `(property ?s ?p ?o)` semantic category maps onto the $(Verb\ ?s\ ?p\ ?o)$ syntactic one if the `rdf:ID="PropertyName"` field in the OWL property definition is a Verb, else it maps onto the $(Datatype\ ?s\ ?p\ ?o)$ syntactic category if the `rdf:ID = "PropertyName"` field in the property definition is a NP. The ontology is preprocessed by a POS tagger to label the `rdf:ID="PropertyName"` field accordingly.
5. The `(object-statement ?s ?p ?o)` and `(datatype-statement ?s ?p ?o)` semantic categories map onto the $(Verb\ ?s\ ?p\ ?o)$ syntactic category if the predicate $(P\ ?s\ ?p\ ?o)$ in the statement is a Verb, else they map onto the $(Datatype\ ?s\ ?p\ ?o)$ syntactic category if the predicate $(P\ ?s\ ?p\ ?o)$ in the statement is a NP, according to the previous correspondence.

4.6 Constructions

4.6.1 Lexical Constructions

A valid lexical construction in FCG notation for the system proposed in this work is reported below:

```
((?Top (meaning (== (C4 ?x)))
          ((J ?new ?top
                   (sem-cat ((concept ?x)))))
<-->
((?Top (form (== (string ?new ‘‘Artist’’))))
          ((J ?new ?top
                  (syn-cat (NP ?x)))))
```

This rule includes both semantic and syntactic category information; it maps the string “Artist” to the meaning (C_4 $?x$), specifies that $?x$ pertains to the semantic category *concept*, and that the syntactic category of “Artist” is (NP $?x$).

Lexical constructions are form-meaning mappings for particular words; they represent the concepts, the individuals, and the properties in the ontology. These words are inferred by *typological rules*. Given an ontology statement, such rules compute first a syntax for its entities based on their declaration in the statement itself, then associate them to the proper role in the SVO structure.

Processing lexical constructions continues by applying the *way-of-saying* rules. Such rules infer new linguistic patterns of for the same semantic description on the basis of its the role in the triple. They build suitable gazetteers for each term outputted from the typological rules. A gazetteer contains the syntax for a term depending on its SVO role. Gazetteers building process makes use of WordNet and FrameNet as data sources.

In the production task, a lexical construction is triggered by a meaning. For example, the lexical construction for the *Artist* concept looks for a meaning ($C_i ?x$) where the logic variable $?x$ can be bound to either a particular artist the system wants to talk about or to the artist concept. The unification process involves only the semantic poles of constructions and if it is successful, the syntactic poles are merged with the linguistic structure that is being built.

When parsing a sentence, lexical constructions are used in the opposite way. Syntactic poles are unified first, and then the information of semantic poles is merged with the linguistic structure to infer the meaning. Concretely, the result of applying a set of lexical constructions to a topic description is a unit structure containing one top-unit and a number of subunits, just as it was explained in the previous chapter.

In the proposed application, the top unit will contain a number of covered meaning predicates, because all subunits are the result of applying lexical constructions, and do not contain any subunits themselves. They do contain a `meaning` and `sem-cat` feature on the semantic side, as well as `form` and `syn-cat` features on the syntactic side. Hence, after application of a set of lexical constructions, the semantic and syntactic structures will look like:

```
((Top (sem-subunits (S1 ... Sn))
          (S1 (meaning M1)
          (sem-cat SemCat1))
          ...
          (Sn (meaning Mn)
          (sem-cat SemCatn)))
```

and

```
((Top (syn-subunits (S1 ... Sn)))
          (S1 (form (string S1 F1))
          (syn-cat SynCat1))
          ...
          (Sn (form (string Sn Fn))
          (syn-cat SynCatn)))
```

where `M1 ... Mn` are lists of predicates with arguments that are also present in the associated semantic and syntactic category lists `SemCat1 ... SemCatn` and `SynCat1 ... SynCatn` respectively.

In the production phase, the system selects a set of lexical constructions according to the structure of the ontology subgraph that matches the user's query. When parsing, the system can use all the lexical constructions or it can select a subset using suitable strategies. As an example, a document TOC can provide information about the constructions to be used. The TOC structure can be converted onto an OWL graph that is mapped onto the domain ontology to select the statements of interest. This strategy has been implemented in the system detailed in Appendix A, where the semantic annotation of Wikipedia articles is discussed. Wikipedia pages are always structured through a TOC that can be used to select the proper lexical constructions. Given the OWL subgraph of interest, the difference between production and parsing using lexical constructions is only the unification process.

However, the verbalization of the ontology is complete when the linguistic structure does not still contain equalities between variables in different units. In fact, it is easy to see that whenever there is more than one unit (apart from the top unit) there will be equalities. This is because the topic description is always completely connected.

We assume that all equalities present in the topic description (i.e. all linking relations contained in it) need to be resolved (expressed) before the syntactic structure is rendered into an utterance. An equality can be resolved either because it is part of the meaning of one lexical construction or else because it is covered by a grammatical construction.
Grammatical construction is required to resolve the equality between both units by imposing an order on them.

4.6.2 Grammatical Constructions

The grammatical constructions are more complex than lexical ones, because they not only have to unify with the top-unit of the linguistic structure, but also with some of the units that were created before by lexical or other grammatical constructions. Moreover they resolve the equality.

A construction that covers the variable equality left uncovered at the end of the previous section looks as follows:

```
((?Top (sem-subunits (== ?unit1 ?unit2)))
(?unit1 (sem-cat (== (SemCat1 ?x))))
(?unit2 (sem-cat (== (SemCat2 ?x))))
((J ?new ?Top)
(sem-cat ((SemCat1 ?x)))))
<->
((?Top (syn-subunits (== ?unit1 ?unit2))
(form (== (meets ?unit1 ?unit2))))
(?unit1 (syn-cat (== (Adjective ?x))))
(?unit2 (syn-cat (== (NP ?x))))
((J ?new ?Top)
(syn-cat ((NP ?x))))).
```

The fact that the new unit is of semantic category `SemCat1` and of syntactic category `NP` can be deduced in the unification phase. On the syntactic side, this construction imposes an order on its subunits.

Given the subgraph in figure 1, the lexical entries for the class definitions of A, B and object property definition p_1 look as:

```
((?Top (meaning (== (C1 ?x))))
((J ?new ?top
(sem-cat ((concept ?x)))))
<->
((?Top (form (== (string ?new "A"))))
((J ?new ?top
(syn-cat ((NP ?x)))))))
```

((?Top (meaning (== (C_2 ?x))))
((J ?new ?top
(sem-cat ((concept ?x)))))
<->
((?Top (form (== (string ?new "B"))))
((J ?new ?top
(syn-cat ((NP ?x)))))))

((?Top (meaning (== (p_1 ?s ?p ?o))))
((J ?new ?top
(sem-cat ((property ?s ?p ?o)))))
<->
((?Top (form (== (string ?new "p1"))))
((J ?new ?top
(syn-cat ((NP ?x)))))))

and the grammatical construction looks as:

```
((?Top (sem-subunits (== ?subject ?property ?object)))
(?subject (sem-cat (== (concept ?x))))
(?property (sem-cat (== (property ?x ?y ?z))))
(?object (sem-cat (== (concept ?z))))
((J ?new ?top)
(sem-cat (object-statement ?x ?y ?z)))
<->
((?Top (syn-subunits (== ?subject ?predicate ?object)))
(syn-cat (==1 (pos (Verb))))
(TAG ?form (form (== (meets ?subject ?predicate)
(meets ?predicate ?object)))))
(?subject
(form (string ?news (stem "A")))
(syn-cat NP))
(?predicate
(form (string ?newp (stem "$p_1$")))
(syn-cat Verb))
(?subject
(form (string ?newo (stem "B")))
(syn-cat NP))
((J ?new ?top (?subject ?predicate ?object))
?form (syn-cat Verb))))
```

4.7 Modeling the Process

We define four main steps for the implementation of our strategy, shown in figure 4.2. Input of the process is the ontology subgraph, that can be computed as we illustrated in the previous section: in production it corresponds to user's requests and in parsing it can be the full ontology or the subgraph related to the document structure, if it exists. How these functions are really implemented and what tools help us to obtain the results are described in the next chapter.
At this time we only model the process abstracting from its implementation that is obviously influenced from used instruments.

Typological Linguistic Rules Application The ontology subgraph is converted in the FOL predicates according to prescriptions described in the section 4.3. Output of this phases is a list (that we call *FOL list*) of predicates or conjunctions of them, that are used in the next step to compute syntax to use in the constructions. In this list there are facts with one predicate (the single fact) and facts that are conjunction of three predicate that model SVO structure. Future extensions, as including other OWL constructs and statements, will be related to expand these rules; considering the syntax of the statement they will define the correspondent FOL facts to represent them through SVO triple.

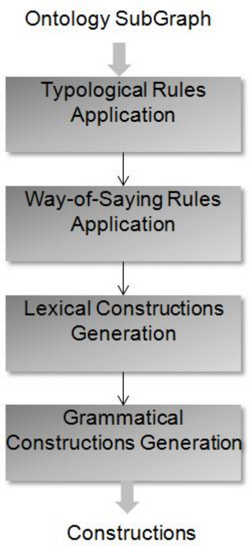

Ontology SubGraph
Typological Rules Application
Way-of-Saying Rules Application
Lexical Constructions Generation
Grammatical Constructions Generation
Constructions

Figure 4.2: From ontology model to constructions: step by step.

Way-of-saying Rules Application The second step of process creates the gazetteers for each fact in the FOL that is a single fact. Facts that are conjunctions of predicates are referred when computing the right syntaxs of the components. For the fact *f*, that can be both *(X ?x)* or *(P ?s ?x ?o)* predicate, the correspondent gazetteer is *Gaz(f)*. The creation of gazetteers is based on rules that appy some functions in different order according few conditions. These functions are modeled as follow:

STEMLEMM(x) that executes both stemming and lemmatization of the word x to reduce inflectional forms and sometimes derivationally related forms of it to a common base form. For instance words as *"fishing"*, *"fished"*, *"fish"* and *"fisher"* correspond to the root word *"fish"*. The result of this mapping of text will be something like:
the boy's cars are different colors
the boy car be differ color.

We have integrated this function with other features: if the word is a compound word (for example composed by more words without blank spaces between them or other separation symbols) the function determines the components words and stem them too.

For instance, the entity *PizzaMargherita* or the property *has-children* became *Pizza Margherita* and *have children* using this function.

POS(x) that returns the part-of-speech tag of the word (or words chain) x computed using Tree Bank Pos Tagger;

SYN(x) that returns synonyms of x;

HYPO(x) that returns hyponyms of x;

HYPER(x) that returns hyperonyms of x;

LEXUN(x) that returns lexical units related to the word x. Lexical units are concepts that describe a frame (event) related to the word.

VERBDEF(x) that returns the Verb Phrase of the dictionary definition of word x.

For each single fact *f* of type *(X ?x)* and *(P ?s ?x ?o)* in the FOL list, first syntax is computed by *STEMLEMM(?x)* function that can returns more values. *STEMLEMM(?x)* output is added to *Gaz(f)*.
For each $(P_i\ ?s\ ?p\ ?o)$ fact template in the FOL list the following rules are applied:

Rule 1 If POS(?p) is a *VP* then add SYN(?p), HYPO(?p), HYPER(?p), LEXUN(?p) to $Gaz[(P_i\ ?s\ ?p\ ?o)]$;

Rule 2 If POS(?p) is a *NP* and FOL list contains $(P_i\ ?s\ ?p\ ?o)(C_j ?s)(I_k ?o)\|$ $\|(P_i\ ?s\ ?p\ ?o)(C_j ?s)(C_k ?o)\|(P_i\ ?s\ ?p\ ?o)(C_j ?s)(E_k ?o)$ then add VERBDEF(?s), VERBDEF(?p), SYN(VERBDEF(?s)), HYPO(VERBDEF(?s)), HYPER(VERBDEF(?s)), LEXUN(VERBDEF(?s)) to $Gaz[(P_i\ ?s\ ?p\ ?o)]$;

Rule 3 If POS(?p) is a *NP* and FOL list contains $(P_i\ ?s\ ?p\ ?o)(I_j ?s)(I_k ?o)\|$ $\|(P_i\ ?s\ ?p\ ?o)(I_j ?s)(E_k ?o)$ then find $(P_l\ ?x$ is-a $?s)(C_i ?x)(I_j ?s)$ and add VERBDEF(?x), SYN(VERBDEF(?x)), HYPO(VERBDEF(?x)), HYPER(VERBDEF(?x)), LEXUN(VERBDEF(?x)) to $Gaz[(P_i\ ?s\ ?p\ ?o)]$;

Some examples will clarify the rules. Considering the triples `< Book year Date >` the typological linguistic rules produce the FOL list that looks as:
$(C_1\ book)$,

(C_2 *date*),
(P_1 *book year date*),
(P_1 *book year date*)(C_1 *book*)(C_2 *date*)

Then, the way-of-saying module considers the (P_1 *book year date*) predicate. For this triple the rule 1 does not fire because POS(year) is not a verb phrase. Rule 2 instead fires because the POS(year) is a noun phrase (NP) and moreover the (P_1 *book year date*)(C_1 *book*)(C_2 *date*) predicate exists in the FOL list. This rule is important when the relation is expressed by a noun phrase because generally *to express a relation it is necessary a verb phrase.* In our example, to say when a book was written generally is not used the noun *year* but a verb as *write* or *was published* and so on.
For this reason, it is possible to refer to the vocabulary definitions of both the subject and the property of the triple to try to compute a verb to express the relation. The function VERBDEF(book) extracts the verb phrase contained in the vocabulary definition of *book* term, that is "a written work or composition that *has been published*".
Inserting the verb phrase *has been published* and its synonyms, hyponyms and for each of them their conjugations to the Gazeetter of the *year* term, we have addressed the expressiveness of the relation according to its semantics.

Rule 3 is useful when the relation is a noun phrase but it is expressed between an instance that can be a proper noun and not a generic concept and it is impossible to find a definition in the dictionary for each proper name (as *Pizza Margherita* or *Pablo Picasso*). For these cases, first the class of the instance is computed (find (P_l $?x$ is-a $?s$)($C_i ?x$)($I_j ?s$)), and then the rule continues as the previous one.

For all other single facts in the FOL list the gazetteers are computed following the order SYN(?x), HYPO(?x), HYPER(?x), LEXUN(?x), eccept for the ($I_i ?x$) single facts that are proper name (as *Atlanta city*). For these kind of single facts the gazetteers contain only the STEMLEMM of the proper name.

Lexical Constructions Generation Lexical constructions are built using the set of single facts gazetteers computed in the previous step. From a gazetteer synonyms lexical constructions are generated; they have the same semantics category (defined as we shown and dependent from the fact) but different syntactic poles, that are the elements of the gazetteer itself.

Grammatical Constructions Generation Conjunctions of predicates in the FOL list are used in this phase, because they specify the structure of the lexical constructions in a sentences.

Constructions are then integrated with grammatical constructions of the used language. In this way other components and information of the unit structures are fitted in the constructions themself and they help to produce and parse text in a verbose manner. Some of these information, for example, regard the gendre of the concept (if it masculine or feminine), or the singolarity and so one, and are actually manually annotated.
The integration rules to obtain automatically more completed grammatical constructions must be improved and it is included in the future works.

4.8 From triples to constructions, an example

Let consider the triple <*Mary work-organization University*>.
Gazetteers for work-organization and University are shown in figure 4.3 and 4.4.

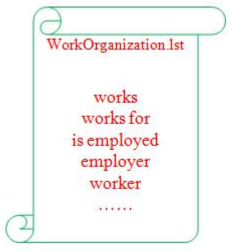


Figure 4.3: Gazetteer for Work-Organization relation that is a NP.

There is only a lexical construction for the proper name *Mary* because it cannot have synonyms.

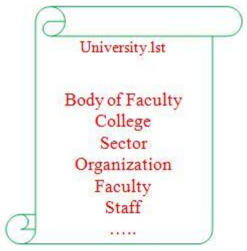


Figure 4.4: Gazetteer for University concept.

```
(def-lex-stem-rule Mary
((?Top (meaning (== (I1 ?m))))
((J ?new ?Top) (sem-cat ((individual ?m)))
<->
((?Top (form (== (string ?new "Mary"))))
((J ?new ?Top)
(form (== (stem "Mary")))
(syn-cat ((NP ?m)
(lex-cat Proper-Noun)
(number sing))))))
```

For the class *Univeristy* instead the gazetteer is computed following our rules and some synonyms lexical constructions are:

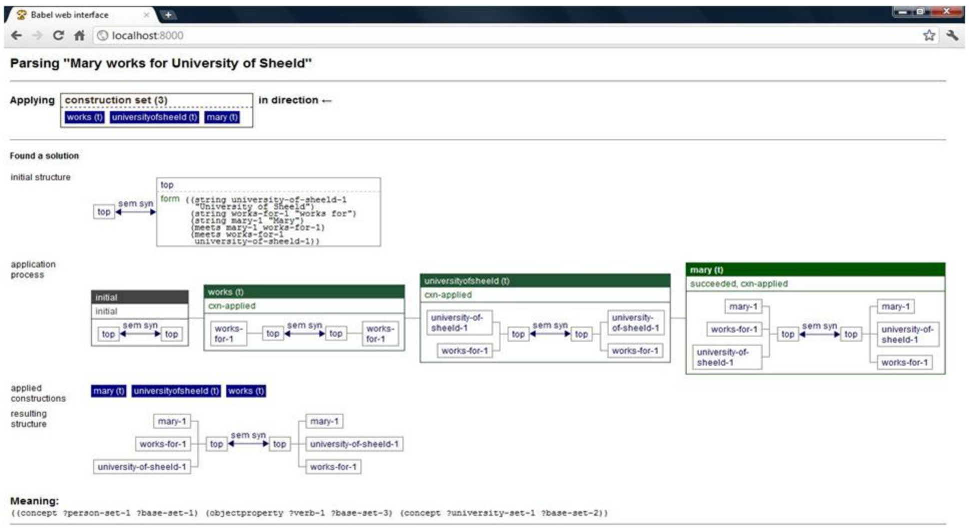
```
def-lex-stem-rule University
((?Top (meaning (== (C1 ?u))))
((J ?new ?Top) (sem-cat ((concept ?u)))
<->
((?Top (form (== (string ?new "university"))))
((J ?new ?Top)
(form (== (stem "university")))
(syn-cat ((NP ?u)
(lex-cat Common-Noun)
(number sing))))))

(def-lex-stem-rule University-3
((?Top (meaning (== (C1 ?u))))
((J ?new ?Top) (sem-cat ((concept ?u)))
<->
((?Top (form (== (string ?new "college"))))
((J ?new ?Top)
(form (== (stem "college")))
(syn-cat ((NP ?u)
(lex-cat Common-Noun)
(number sing))))))

(def-lex-stem-rule University-5
((?Top (meaning (== (C1 ?u))))
((J ?new ?Top) (sem-cat ((concept ?u)))
<->
((?Top (form (== (string ?new "faculty"))))
((J ?new ?Top)
(form (== (stem "faculty")))
(syn-cat ((NP ?u)
(lex-cat Common-Noun)
(number sing))))))
```

Similary, for the relation *work-organization* some synonyms lexical constructions are:

```
(def-lex-stem-rule-temp workOrganization-1
((?Top (meaning (== (P1 ?p ?s ?o)
((J ?new ?Top)
(sem-cat ((property ?p ?s ?o)))))
<->
((?Top (form (== (string ?new "works"))))
((J ?new ?Top)
(form (== (stem "works")))
(syn-cat ((constituent V))))))
```

```
(def-lex-stem-rule-temp workOrganization-2
((?Top (meaning (== (P1 ?p ?s ?o)
((J ?new ?Top)
(sem-cat ((property ?p ?s ?o)))))
<->
((?Top (form (== (string ?new "employer"))))
((J ?new ?Top)
(form (== (stem "employer")))
(syn-cat ((constituent NP))))))
```

Next figures show the FCG interface during parsing and production of sentence *Mary works for University of Sheeld.*


Parsing "Mary works for University of Sheeld"

Applying construction set (3) in direction ←

Found a solution

initial structure

```
form ((string university-of-sheeld-1
        "University of Sheeld")
      (string works-for-1 "works for")
      (string mary-1 "Mary")
      (meets mary-1 works-for-1)
      (meets works-for-1
        university-of-sheeld-1))
```

application process

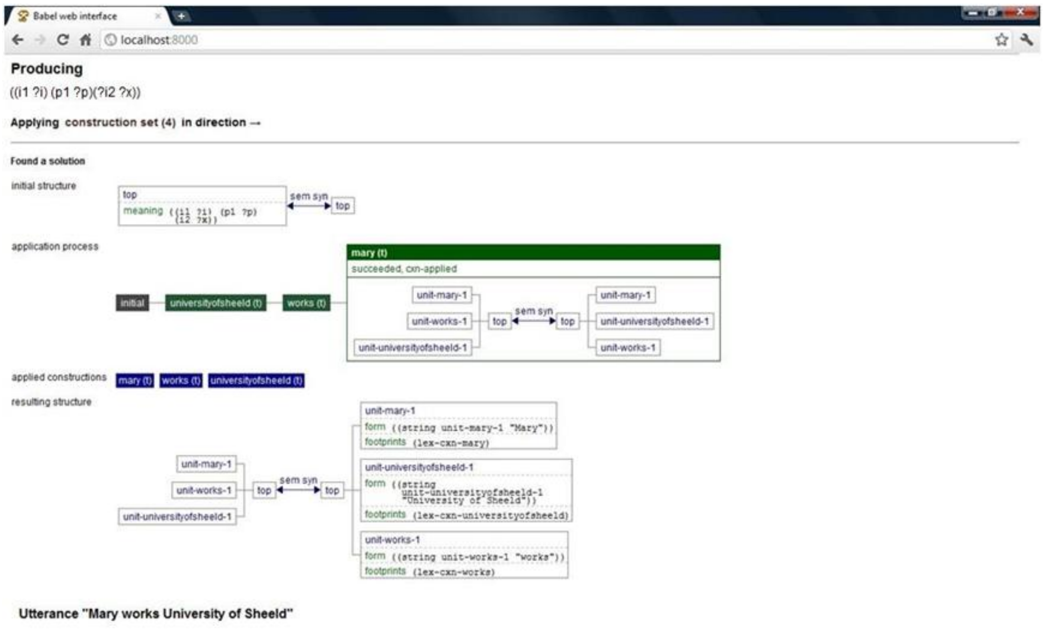
applied constructions

resulting structure

Meaning:

```
((concept ?person-set-1 ?base-set-1) (objectproperty ?verb-1 ?base-set-3) (concept ?university-set-1 ?base-set-2))
```

Figure 4.5: FCG interface that shows the parsing of "Mary works for University of Sheeld".


Producing

((i1 ?i) (p1 ?p)(?i2 ?x))

Applying construction set (4) in direction →

Found a solution

initial structure

```
top
meaning ((i1 ?i) (p1 ?p)
         (i2 ?x))
```

application process

applied constructions mary (t) works (t) universityofsheeld (t)

resulting structure

```
unit-mary-1
form ((string unit-mary-1 "Mary"))
footprints (lex-cxn-mary)
```

```
unit-universityofsheeld-1
form ((string
       unit-universityofsheeld-1
       "University of Sheeld"))
footprints (lex-cxn-universityofsheeld)
```

```
unit-works-1
form ((string unit-works-1 "works"))
footprints (lex-cxn-works)
```

Utterance "Mary works University of Sheeld"

Figure 4.6: FCG interface that shows the parsing of "Mary works University of Sheeld".

Chapter 5

Some Instruments and Architecture

5.1 Introduction

The implementation of the model described in the previous chapter is based on the usage of a set of tools that help us to realize the steps we have defined. These tools are generally useful in Natural Language Processing research fields and have an undisputed role in the community of semantics approaches.

Among all FrameNet and WordNet sources are worthies of attention because they return values that conditioning results of our approach.They are used to the aim to obtain from them the different syntaxs that are related to a word or a set of them; they implement the functions that are used by our way-of-saying module. How these information are returned and what they mean are discussed in the next two sections.

After a briefly illustration of them we 'll describe how they interact with the components of our system and the other natural language processing tools. All together they constitute our framework architecture.

5.2 FrameNet

Frame semantics is a subfiled of semantics that studies the combined meaning of a structure of related concepts.
Differently from other semantic approaches, frame semantics provides no meaning representation for single words, but for the so-called *frames*: a frame typically represents an event, a situation and describes the meaning of the objects in the event, including its participants, properties and other related conceptual roles, that are defined *frame elements*.

For example, the COMMERCIAL_TRANSACTION frame represents a situation where a buyer, a seller, money and goods are always presents. Roles of these entities are defined as *core* roles, because they are always in this situation. A frame can also have optional *non-core* role, that can not be characteristics for the situation. In the example, concepts as the medium of exchange, the currency of the money and the payment rate per unit are non-core role for the commercial transaction frame. What non-core roles exist is shown next.

A single words can evoke a frame or be a property of the same. In general, frames can be evoked by a number of words which have a semantically related meaning. Examples for the previously mentioned COMMERCIAL_TRANSACTION frame include the words "buy" and "sell". Most of these so-called lexical units (LU) are verbs and nouns but other grammatical categories (e.g. adjectives and adverbs) are possible.

The Berkeley FrameNet project is the project that currently builds a frame semantic lexicon for English that includes over 10,000 LUs (Ruppenhofer et al., 2006). Annotations are done manually and include tagging of lexical units and frame element fillers (FEF).

5.2.1 FrameNet Structure

The FrameNet database is organized around three main elements, which have been developed following Fillmore's theory about frame semantics:

Semantic frame: a semantic frame is the conceptual structure that describes a particular type of situation, object or event and the participants involved in it, for example Apply heat, Color, Judgment.

In the FrameNet database, frames has a definition, for example Achieving first is described as *"A Cognizer introduces a New idea into society"*, where Cognizer and New idea are the main participants defined in the situation.

The frame definitions can have different information and do bit follow strict format constraints. In some cases, they include information about the syntactic behavior of lexical units or also add information about the realization of frame elements. We report an example from the Appearance frame: "In this class of perception words, a Phenomenon, typically expressed as External Argument, and its perceptual characteristics are given some description".

Lexical unit (LU) or target: a LU is a word, a multiword or an idiomatic expression that evokes a specific frame. Differently from WordNet synsets (Fellbaum, 1998) that are next discussed, lexical units in the same frame can belong to different grammatical categories. For instance, in the Achieving frame (that is the first frame) the LUs include verbs, nouns and adjectives: *coin.v, coinage.n, discover.v, discovery.n, invent.v, invention.n, inventor.n, originate.v, originator.n, pioneer.n, pioneer.v* and *pioneering.a.* Summarily, all such LUs would evoke the same event, described in the Achieving first definition.

Frame elements (FEs): a FE is a frame that specifies semantic roles. With verbal LUs, they are usually realized by the syntactic dependents of the verb. FEs are classified into *core, peripheral* and *extra-tematic*, basing on how central they are to a frame. A core FE is conceptually necessary to the situation described in a frame because it contributes to characterize it uniquely. For example, in Achieving first, Cognizer and New idea are core frame elements because

A*peripheral* FE does not characterize uniquely a frame. Indeed, it is usually recurring in different frames and marks such notions as Time, Place, Means, etc. The *extra-thematic* FE differently from the peripheral type, introduces an additional state or event. For this reason, these FEs do not conceptually belong to the frame they appear in and have a somewhat independent status. They can even evoke a larger frame embedding the reported situation.

5.3 WordNet

WordNet is a freely lexical database for the English language, and it combines a dictionary and thesaurus toghether, that may be browsed online and downloaded for local usage at its official web site. The aim of WordNet is the mapping of the relationships between words similar to the way the humans keep in mind and uses language.

Project development started in 1985, when a group of psychologists and linguists directed by George A. Miller at Princeton University's Cognitive Science Library develops their own lexical database for their research goals into the fields of psycholinguistics and psycholexicology. The objective of the project was to create a resource which would keep into account the humans processing languages.

In 1993 first results of the group's effort were published, and the main was a structured lexicon which provided a valid alternative to traditional alphabetical dictionaries and thesauri. Since WordNet is distributed under a BSD style license, it may be used free of charge. Development on WordNet has continued since its inception and remains ongoing at Princeton University. WordNet has grown considerably since is development began. The database is currently at version 3.0 and contains over 155,000 words which are grouped into roughly 117,000 synsets totaling approximately 207,000 word-sense pairs.

5.3.1 WordNet Types

Since different words follow different grammatical rules, WordNet makes the distinction between four of the primary word types in the English language. *Nouns, verbs, adjectives, and adverbs* are the defined WordNet Types.

The nouns category contains words which refer to concepts, qualities, entities, actions, or state and are used as the subject of a verb. Words classified as verbs may serve as the predicate of a sentence and describe an action, occurrence, or state of existence. Adjectives are words that modify nouns. The final word classification stored in WordNet, the adverb, is similar to the adjective and contains words which modify word types other than nouns.

Each group has its unique set of semantic and lexical connections which link that group's members. Thus, the possible relationships shared between WordNet synsets vary depending on the type of word. Noun synsets may be connected as hypernyms, hyponyms, coordinate terms, holonyms, and meronyms. Verb synset relationships include hypernyms, troponyms, entailments, and coordinate terms. Possible adjective relationships are related nouns, similar to, and participle of verb. Adjective synsets may be related as root adjectives. Words may also be connected through lexical relations such as antonyms.

5.3.2 WordNet Relationships

In WordNet, words and the relationships between them are hierarchical organized in taxonomies which may be found in the natural sciences. Words which are closely related to each other may be found in the same branch of the hierarchy's tree. Each word has a set of synonyms, called *synset*. Formally, a synset is a set synonymous words that have the same meaning and that can be substituted for each other; substitution in context does not change the overall meaning of the utterance in which they are contained. Synsets are the foundation upon which the WordNet database is built.

Words which have multiple meanings or "word senses" are added in more than one synset. WordNet provides a *polysemy counter* for each word that tracks the number of synsets in which the word is contained. Also WordNet provides a *frequency score* which computes how often a word will appear in a specific sense.

As shown, WordNet separates its words into one of four basic word types including nouns, verbs, adjectives, and adverbs. The database is able to track semantic relationships connecting synsets that are specific to each word type, as well as lexical relationships between individual words. By searching for a word in WordNet, not only are synsets which contain the word in various senses returned, but the semantic and lexical relationships which are applicable for each synset may be displayed as well. This allows the user to quickly find words related to their original query. The following subsection shows further details on these possible word relationships.

5.3.3 Noun, Hypernyms and Hyponyms

The first noun relationships stored in the WordNet database is *hypernymy.* A word *X* is a hypernym of word *Y* when the semantic range of word *Y* lies within that of word X. An example of this relationship can be shown by the synsets containing the words "ananas" and "edible fruit". "Edible fruit" is a hypernym of "ananas" because all ananas are edible fruits. As such, "edible fruit" would be a hypernym of "cherry" and "apple" as well.

Hyponyms are words whose semantic range lie within that of a hypernym. Going back to the previous example, the words "ananas", "cherry", and "apple" would all be related to the word "fruit" by hyponymic relationships, which is to say that they are all hyponyms of "fruit". If multiple words share a hypernym, as "ananas", "cherry", and "apple" do, then they are said to be *coordinate* or *sister terms*, the third type of noun relationship tracked by WordNet.

5.3.4 Noun, Holonyms and Meronyms

Another noun relation which WordNet defines for synsets is *holonymy.* A word *X* is a holonym of word *Y* if word *Y* represents *a part* or member of word *X.*
To provide an example of this relationship, in a car there are a handbrake and a wheel. so the word "car" is a valid holonym of the words "handbrake" and "wheel".

Other semantic relationship for the noun type is the direct opposite of the holonymic relationship, the *meronymy.* This relationship refers to two words *X* and *Y* where word *Y is a part of* word X. In this case, word *Y* is a meronym of word *X*. To use the same example, the words "handbrake" and "wheel" share a meronymic relationship with the word "car", as a car is made up of handbrake and wheel.

Closely to the holonymic and meronymic semantic relationships there is the *the domain category* concept. These categories refer to groups of synsets whose members represent the domain to which another synset belongs.
The synsets containing "animal" and "human" are domain categories of the synset containing "body" when referring to the entire structure of an organism.

5.4 System Architecture

In this section we describe the components that implement our model. Each component provides a specific function that realizes the steps of our model previously discussed.

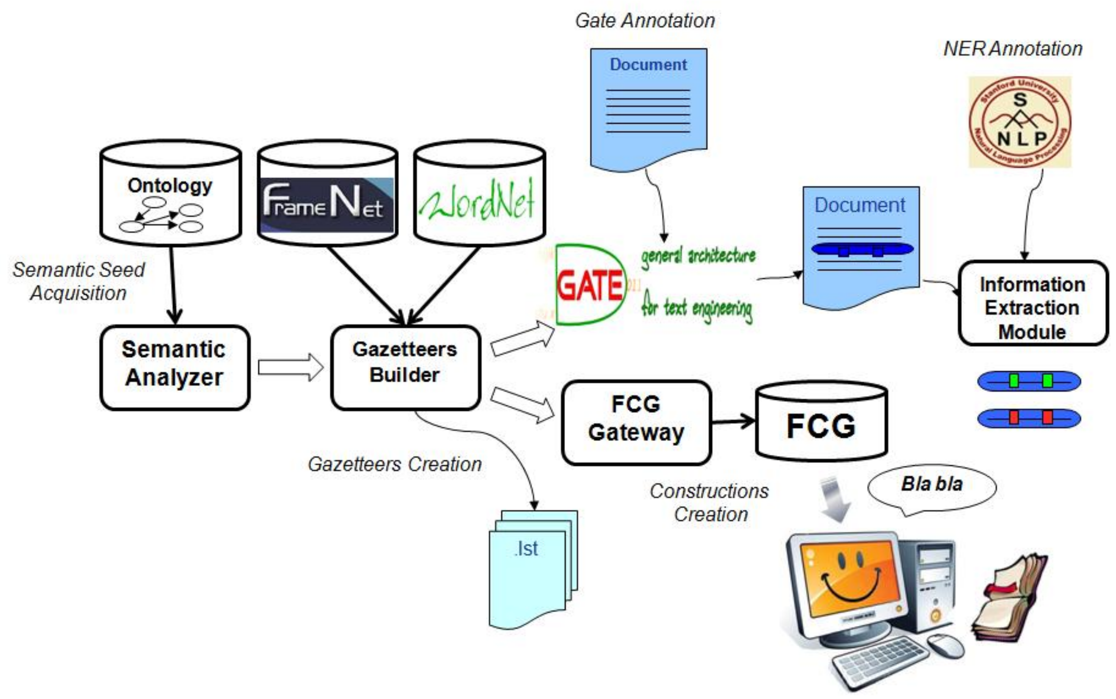
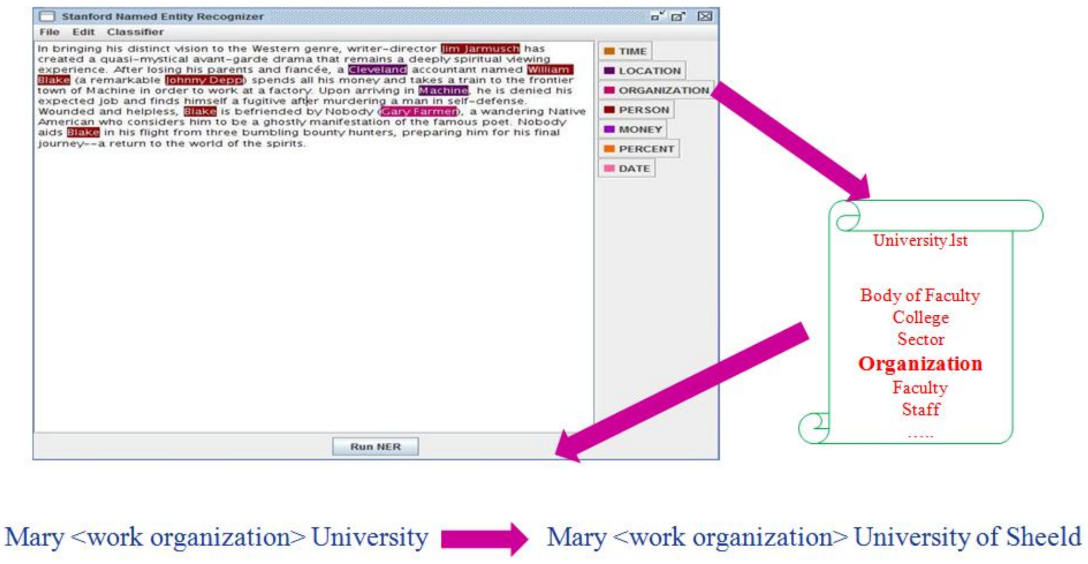


Figure 5.1: System's overview.

First component is the *Semantic Analyzer*, that extracts the information contained in the ontology and represents them through SVO triples using the FOL predicates that we have defined in our axiomatization (facts with form *(X ?x)* or *(P ?s ?p ?o)* and conjunctions of them) that is formalized by our typological rules. Moreover the Semantic Analyzer disposes the information it extracts for building the syntactic poles of the lexical constructions and to that end it retrieves and stores the labels from the ontology (value of *rdf:ID* or *rdfs:label*) correspondents to the SVO components. The labels are a first syntaxs to express ontology entities. The Semantic Analyzer is entirely realized in Java language and it browses ontology by Jena (McBride (2002)).

The well structured information from the Semantic Analyzer are then inputted to the *Gazetteer Builder* component that uses WordNet (Fellbaum (1998)) and FrameNet (Ruppenhofer et al. (2005)) sources for creating the gazetteers; these gazetteers are filled applying our way-of-saying rules. At this point the bricks to build the constructions are available.
These bricks are then processed by the *FCG Gateway* that writes both lexical and grammatical constructions in the FCG syntax that is the Lisp list notation, (for each gazetteer it creates synonyms lexical constructions) and puts them into FCG framework. Summarily, this component realizes a bridge between our framework and the FCG one, because our framework is coded in Java language while the FCG framework is based on Common Lisp. To that end the FCG Gateway uses the *Cl + J* interface that allow to write Lisp construction through Java language. These interface was developed by Jean-Claude Beaudoin, a freelancer of Common Lisp tools, that has made available its work for our objectives.

Also, these gazetteers are used from GATE (Cunningham et al. (2011)) for annotating plain documents through the domain ontology. In these branch of framework, we have developed another component, the *Information Extraction Module* that is able to extract new instances from free text that may be put into the ontology (if they do not already exist). Summarily, the Information Extraction Module uses a Named Entities Recognizer (Finkel and Manning (2009)) to compute the categories of some entities in the text and if these entities are linked through relations that are represented in the ontology for which subject and object represents the same category recognized for the entity, then the entity can be istantiate for the object or subject of the ontology relations. Figure 5.2 shows an example of this instantiation.



In bringing his distinct vision to the Western genre, writer-director Jim Jarmusch has created a quasi-mystical avant-garde drama that remains a deeply spiritual viewing experience. After losing his parents and fiancée, a Cleveland accountant named William Blake (a remarkable Johnny Depp) spends all his money and takes a train to the frontier town of Machine in order to work at a factory. Upon arriving in Machine, he is denied his expected job and finds himself a fugitive after murdering a man in self-defense. Wounded and helpless, Blake is befriended by Nobody (Gary Farmer), a wandering Native American who considers him to be a ghostly manifestation of the famous poet. Nobody aids Blake in his flight from three bumbling bounty hunters, preparing him for his final journey--a return to the world of the spirits.

Figure 5.2: If the category outputted by Ner is in the gazetteers of the entity itself, system is able to infer new instance if it does not exist.

Chapter 6

Main Experiments

6.1 Introduction

The main claim of this thesis was the model definiton for integrating ontology into Fluid Construction Grammar framework; to that aim we defined a set of rules that allows system to create a lexicon for syntactically structuring the meaning of an ontology subgraph. This lexicon depends from the semantic roles covered by the entities themself in the specific graph context.
The result of our rules defined in a previous chapter is a set of units structures for semantically parsing external documents related to domain under investigation and for producing about each available contents (external and internal from the ontology).

With our experiments we demonstrate that these rules lead to good results because they define the really way-of-saying to express the event modeled in the ontology and useful by humans. This result has effects in both natural language processes, the production and the parsing. In this chapter we present these main experiments and for clarity we describe them through two baselines: former is related to the parsing and latter is related to the production.

To diversify the results, we present the two baselines in two different application domains. The parser is applyed to documents from the Art domain and the

production is realized about topics of the enteprice domain, in particular the domain related to the managing of a set of connected clients of a bank; for both cases we use a specific ontology, *OntoArt* ontology for the first case, that was modeled by us, and a corporate ontology provided by *Consorzio Monte Paschi di Siena* for the latter case.
The choose is not casual, but it depends by the availability of documental corpora: for the art domain we may refer to a large amount of documents that are shared on the Web, the Wikipedia articles related to artists biography. For the corporate ontology we cannot show the documents related to it because they are reserved by rights, so here we present a small fragment of the ontology that we use to produce sentences about it.

Moreover, for each baseline we define specific evaluation parameters, because the experiments are radically differents; infact, if to evaluate the goodness of the parser we have to adopt a statistical approach that keeps in account the number of annotated words in a large set of documents, to evaluate the production we have to refer to the parameter that represents the ability to compose sentences from meaning, that is the *compositionalilty degree.* This is more clear next.

6.2 Baseline Experiment 1

In this section we illustate the results wa have reached parsing a set of documental corpora shared on the Web, the Wikipedia articles about artist biography. Figure 6.1 represents a fragment of ontology we use for doing that, the OntoArt ontology. Accurate process description through which we elaborate each page to obtained a semantically annotated Wiki pages (that are then shared by Semantic MediaWiki (Krötzsch et al. (2006)) web site) is illustrated in the Appendix A; in this appendix section we also illustrate how we select the concepts from the ontology. The strategy is made possible because each Wikipedia pages has a Table of Contents (TOC) that give to the page a first structure, and we use it to extract pertinent and relevant concepts from the ontology.

As explained, in order to illustrate these kind of experiments we have to adopt

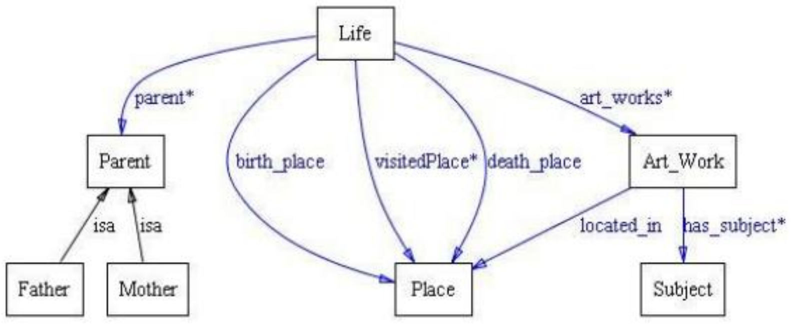


Figure 6.1: A fragment of the Art domain ontology.

a statistical approach. In particular we refer to *Zipf's law.*
Zipf's Law, named after the Harvard linguistic professor George Kingsley Zipf (1902-1950), is a an empirical law formulated using mathematical statistics and based on observation rather than theory. It is often true of a collection of instances of classes, e.g., occurrences of words in a document and for this reason we believe it adapts to our evaluations.

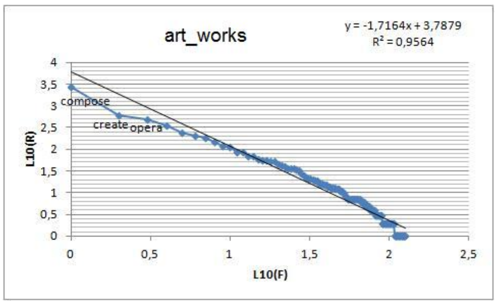
The law refers to the fact that many types of data in the physical and social sciences can be approximated with a Zipfian distribution (one of a family of related discrete power law probability distributions); formally it says that the *frequency of occurrence of an instance of a class is roughly inversely proportional to the rank of that class in the frequency list.*
More exactly, suppose a word occurs f times and that in the list of word frequencies it has a certain rank (a position), r. Then if Zipf's Law holds we have (for all words)

$$f = \frac{a}{r^b}$$

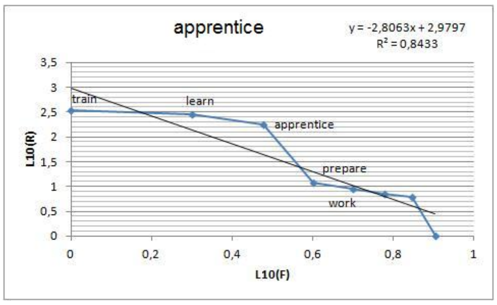
where a and b are constants and b is close to 1. Taking the logarithm of each side of the equation we obtain

$$log(f) = log(a) - b * log(r),$$

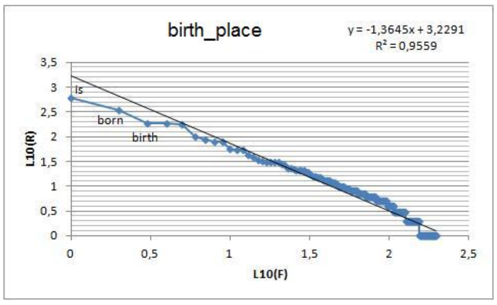
where the *log* function can be to any base, such as e or 10. Thus if Zipf's law



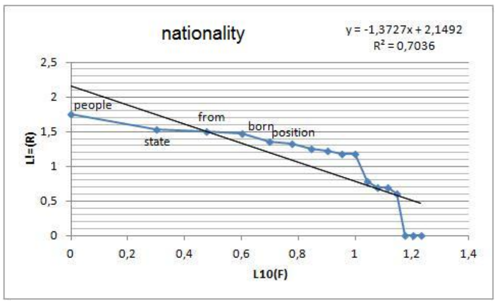
(a) *art_works* relation between Artist and Art_Work concepts.



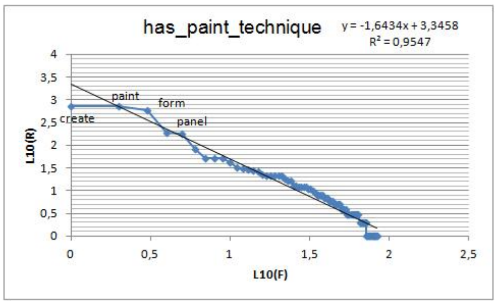
(b) *apprentice* relation between Artist and Person concepts.



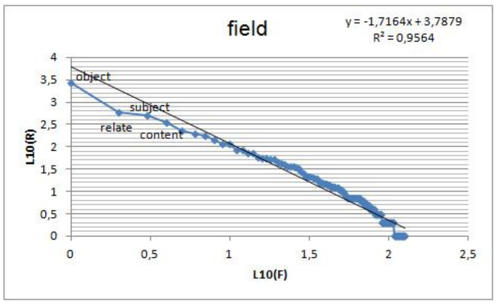
(c) *birth_place* relation between Artist and Place concepts.



(d) *nationality* relation between Artist and Nationality concepts.



(e) *has_paint_technique* relation between ArtWork and PaintTechnique concepts.



(f) *field* relation between ArtWork and Field concepts.

Figure 6.2: Zpfian distribution on 20 Wikipedia pages of words or sentences in the gazetteers of 6 triples.

holds of a particular collection then if we graph $log(f)$ against $log(r)$ the resulting graph will be a straight line with slope close to -1.

A consequence is that $(if\ b = 1)$ a word of rank k occurs $1/k$ times as often as the most-frequently-occurring word. We see this from: $f(k)/f(1) = (a/k)/(a/1) = 1/k$. And for every k occurrences of a word of rank j there are approximately j occurrences of a word of rank k.

We assume that: *when a word or a sentence with a meaning has a hight frequence in a set of documents related to the argument for which the meaning is pertinent, it means that this word or this sentence is a useful human way of saying this meaning.*
When we say that the meaning is pertinent for the argument we mean that the meaning has sense for it; for example, if the argument is the life of a musician, we can speack about what sings he has composed, so the fact that represents the sings for a musician is a pertinent meaning for the musician life argument.
Considering that Wikipedia articles are written by human editors without constraints, these pages are well suited for our observations, that is to verify if the way-of-saying rules compute right words to express something.

In figure 6.2 are shown some graphs that represent the Zipfian distributions on 20 Wikipedia pages related to artists's biography of words from gazetteers filled following our rules that are applyed on triples from OntoArt ontology. Really experiments were executed on a larger set of pages, about 100, but for ensuring clarity for these kinds of figures here we present the results only on a smaller subset (it is not a limit for our observations). Observing the figures we note that the order followed to fill the gazetteers respect the really frequence distributions of the word or the sentence, that means the first words or sentences that the way-of-saying rules put in the list follows the effective degree of usage of them.
For example, the subgraph 6.2(c) that represents the distribution for the concept *birth_place* shows that to express where someone was born is widely used the word *born* and not the NP *birth_place*, that is what our rules compute. Similary for others subgraphs, for which specifications are represented in the same figure.

It is a confirmation that our incremental approach is a good way to compute adeguate syntax considering a semantics, that is intrinsic in an ontology model. Perhaps this is the main interesting results of our model.

Another observation is related to the correctness of the semantic annotations, for which we have to evaluate if the meaning we attribute to words in the text is right. It means, considering the same example, that we have to evaluate if the year of born of the an artist is correctly annotated with the *birth_date* property and not with others properties or not totally annotated. For this kind of relevations we can not regardless from a manual evaluations of the results, because nothing can tell us if they are corrects if he is not a human. This is often true for many processing resources such as tokeniser, POS taggers and so on.
To evaluate the goodness of semantics annotations we refer to four statistical parameters: *precision*, *error rate*, *recall* and the *F-measure*. Precision, error rate, recall and F-measure are also appropriate choices when assessing performance of an automated application against a trusted gold standard.

Precision measures the number of correctly annotations as a percentage of the number of annotations identified. In other words, it measures how many of the annotations that the system identified were actually correct, regardless of whether it also failed to annotate correct items. The higher the precision, the better the system is at ensuring that what is identified is correct.

Error rate is the inverse of precision, and measures the number of incorrectly identified annotations as a percentage of the annotations identified. It is sometimes used as an alternative to precision.

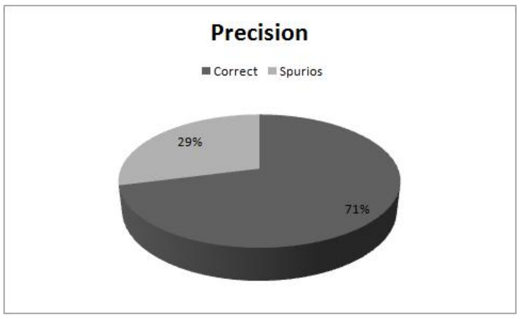
Recall measures the number of correctly identified annotations as a percentage of the total number of correct annotations. In other words, it measures how many of the annotations that should have been identified actually were identified, regardless of how many spurious identifications were made. The higher the recall rate, the better the system is at not missing correct items.

Clearly, there must be a tradeoff between precision and recall, for a system can easily be made to achieve 100% precision by identifying nothing (and so making no mistakes in what it identifies), or 100% recall by identifying everything (and so not missing anything). The *F-measure* is often used in conjunction with Precision and Recall, as a weighted average of the two. False positives are a useful metric when dealing with a wide variety of text types, because it is not dependent on relative document richness in the same way that precision is. By this we mean the relative number of entities of each type to be found in a set of documents. Formally, defining the variables P and R respectively the precision and the recall:

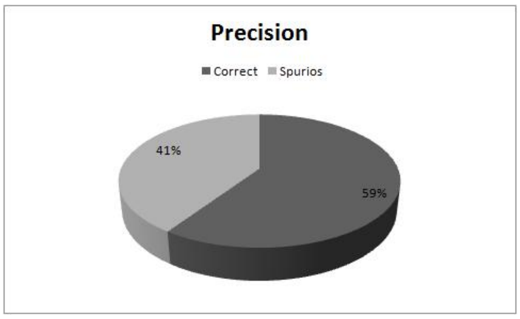
$$P = \frac{CorrectAnnotations}{CorrectAnnotations\ +\ SpuriosAnnotations}$$

$$R = \frac{CorrectAnnotations}{CorrectAnnotations\ +\ MissingAnnotations}$$

$$F - measure = \frac{2*(P\ \ X\ \ R)}{R\ +\ P}$$



(a) Evaluation of incremental approach.



(b) Evaluation without incremental approach.

Figure 6.3: Precision measure on 100 Wikipedia pages.

During parsing we have observed two things:

- if we apply an *incremental approach* that searches terms ona at a time in the text from the gazetteers following their order and stops when a term is found, we obtain better results; the motivation is that in a free text there is generally only a sentence where the specific meaning is expressed, and when we find it to search more is idle. Moreover, our strategy first searches the useful words or sentences to express something and the results are obviously further improved respect the case in which they are not so ordered;

- if we search the full gazetteers in the text, without an incremental approach, and so we do not stop when one term is found, the semantic annotations have a smaller rate of correctness because words that not have a specific meaning are however annotated through it.

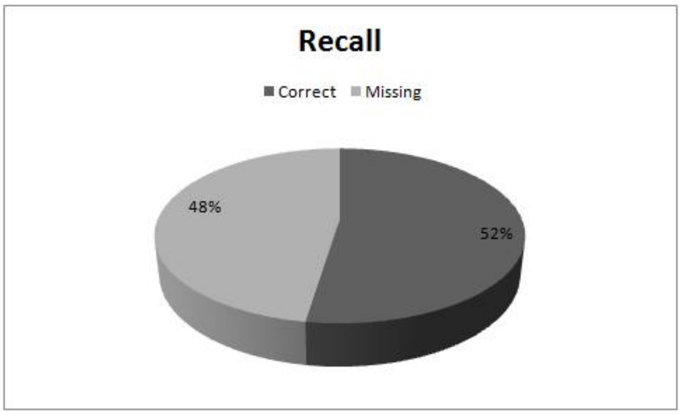


Figure 6.4: Recall measure on 100 Wikipedia pages.

Figure 6.3 shows the percentage of correctness of results in the semantic annotations of no-incremental approach versus incremental one. We reiterate that an annotation is considered correct when it is present and it specifies right meaning. Recall measure is shown in figure 6.4; mesurament in this case is however influenced by ontology, because the missing results are computed not considering what entities are modeled into the ontology but only the meaning that could be understood by humans.

6.3 Baseline Experiment 2

In this set of experiments we evaluate the production of the system considering the corporate ontology provided us by Operative Consortium of Monte Paschi of Siena. Because it is an italian bank, the ontology is filled using italian language and we have changed the versions of instruments to apply our framework; in particular, we have to use Italian Wordnet (Pianta et al. (2002)), available at *http://multiwordnet.fbk.eu/english/home.php*. This ontology models the Groups of Connected Clients of the bank, that are the sets of clients with specifics characteristics for which is possible to get them specific financial service. In figure 6.5 is shown a fragment of this ontology that models the concept Bank and others concepts related to it. Next to refer this ontology we call it *MontePaschi* ontology. The evaluation of the ability of an automated system to natural language produce

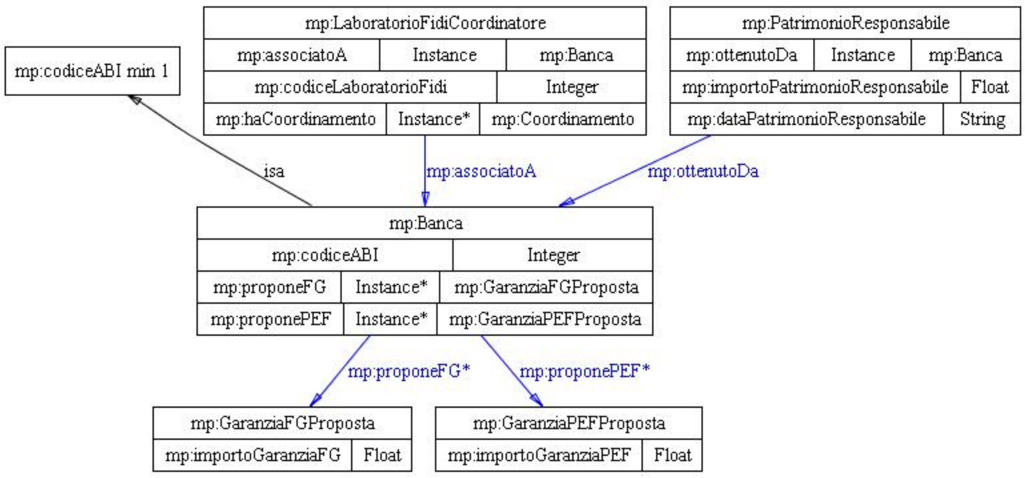


Figure 6.5: A fragment of Groups of Connected Clients ontology provided by Operative Consortium of Monte Paschi of Siena.

an utterance is often bound to the evaluation of the goodness of the lexicon that system can use to speack considering problems as synonyms and homonyms. Moreover, system should have a background of grammatical structures of the language to more understandable produce for humans.

Fou our aim the completness of a sentence is not a foundamental problem; that means, when we want to evaluate if the system is able to speack about the ontology, we consider if it produces something that is understandable and above all true, that is we evaluate if the domain is correctly described and comprehensible without worry if the sentences are grammatically severely rights. It agrees to the *fluidity* principle formuled by Steel [Steels and van Trijp (2011)] that we have already shown in the chapther "Construction Grammar and Fluidity" but that we recall here for clarity: *"when speakers need to express something that was not yet conventionally expressible in the language, hearers are assumed to be intelligent* *enough to figure out what was meant".*

Instead, we had to keep in account problems related to the lexicon that is built automatically by system. The notion of *semiotic graph* allow us to define measures that can be used to evaluate some of these problems that are shown next.
To introduce the semiotic graph notion we have to model the "spaces" involved in the ontology verbalization scenario, that concern with the real world and how it is described respectively. During production, a speaker performs a mapping from her concepts space to utterance space. In turn, parsing performs a mapping from utterance space to concepts space.

Hence, a *semiotic graph* can be defined that consists of a set of nodes from the concepts space (the *meaning nodes*), a set of nodes from the utterance space (the *form nodes*), and a set of *weighted directed links* between them. Every meaning node represents a different semantic description, while every form node represents an utterance. An arrow going from a meaning node to an form one represents a production (i.e. verbalization); an arrow going in the opposite direction represents a parsing action (i.e. understanding).

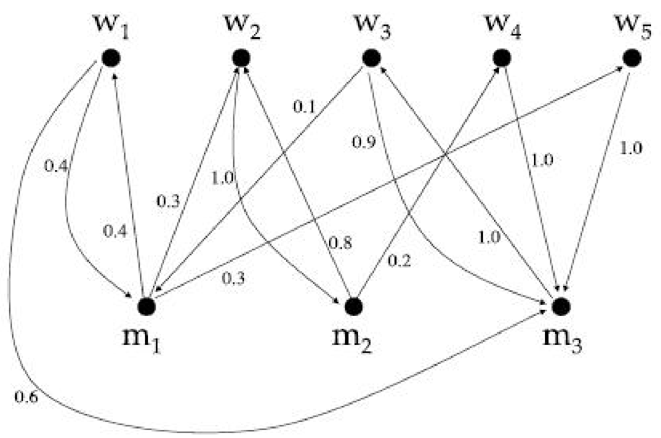


Figure 6.6: An example of a semiotic graph. Nodes marked as m_i represent a meaning (a conjunction of predicates), while the ones marked as w_i represent utterances. Weights on the edges refer to probabilities to obtain a certain production or parsing.

We can build the semiotic graph for the ontology subgraph of which we verbalize; in particular, the construction of the semiotic graph for an ontology subgraph follows the rules for semantics and syntactic categories definition in our model, so building a semiotic graph is equal to define the poles of constructions that must be to incorpote in the FCG; in particular, each meanings we extract from the ontology (represented as the FOL predicates obtained after our axiomatization) is represented through meaning nodes (such as m_i in the figure) and the words or sentences we associate to them computing by way-of-say rules are represented through form node.
Edges in the semiotic graphs have weights whose meaning depends on how the graph itself was built.
In this baseline experiments weights represent two things:

- the probabilities of a certain meaning being verbalized using a certain utterance;
- the information related to the number of synonyms the system knows for a specific meaning.

In the first assumption the weight in a link from meaning node to form one is the probability that the form is used to express the meaning so it depends by the frequency of usage of the form to express the meaning itself: in the previous section we have demonstrated, using the Zpfian distributon of words or sentences from gazetteers in a large amount of free documents, that our rules compute the correct order in the usage of the terms or the sentences to express something so they give us the useful information to compute this probability. Because here we evaluate the production, we formalize only the computation of the weights of the links from a meaning node to more form ones and we consider the ranks in the gazetteers of each words or sentences correspondents to form nodes.
Because the gazetteers have always less than 100 elements, we scale the Gauss's formula with the ranks of the elements to compute these weights; formally, *the weight w_i of a link* i *from meaning node* m *to form one* f *that represents the word or sentence with rank* i *in the gazetteer of* m *is computed as:*

$$w_i = \frac{2 * (n + i - 1)}{n * (n + 1)}$$

where n is the number of elements in the gazetteer.
Applying this formula, the links with lower ranks have the higher weights; it means that former words or sentences in the gazetteer are the most useful to express the meaning. For example, in figure 6.7 a semiotic graph correspondent to concept *Banca* from the MontePaschi ontology is shown. There is one meaning node, that is the meaning of the *Banca* concept modeled by our FOL predicates, and more form nodes, that represent the different syntaxs to express this concept. Weights are computed applying our formula and considering the order of the elements in the gazetteer computed for *Banca* concept.

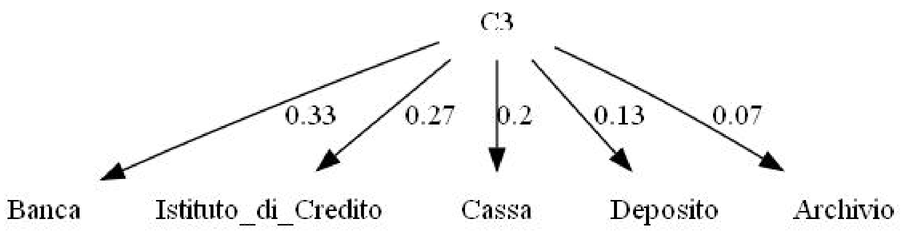


Figure 6.7: The semiotic graph for the concept *Banca* in production.

Considering latter point in the weights interpretation, weights can be used to keep information about synonymy and homonyms.
Synonyms exists when a meaning node is associated with many form nodes. Strictly speaking this is called synonymy only when the involved points in utterance space represent single words. Figure 6.8 illustrates this case. Possible sources of synonymy include:

- wrong guess by the hearer of the meaning of a word, resulting in the adoption of an additional word for a concept which was already associated with another word;
- different words proposed by the system for the same concept.

Regarding homonyms there are at least two mechanisms that introduce them:

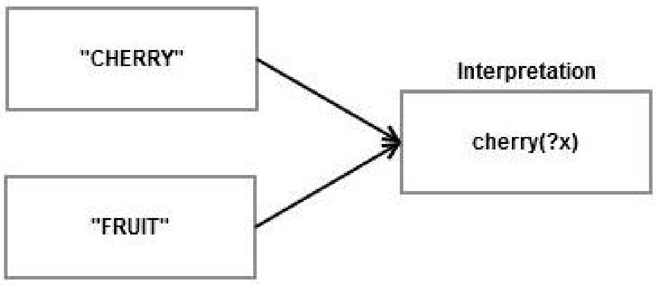


Figure 6.8: Synonymy as represented in a semiotic graph: both the words "cherry" and "fruit" are shown to map to the same concept meaning.

- system could independently propose the same word for a different concept or, equivalently, it could propose the same word for two different concepts;
- the meaning of an unknown word could be uncertain and hence system would be used in a way that is different from its original meaning.

Figure 6.9 illustrates homonymy. Obviously, one speaks of homonyms only when the points in utterance space represent single words.

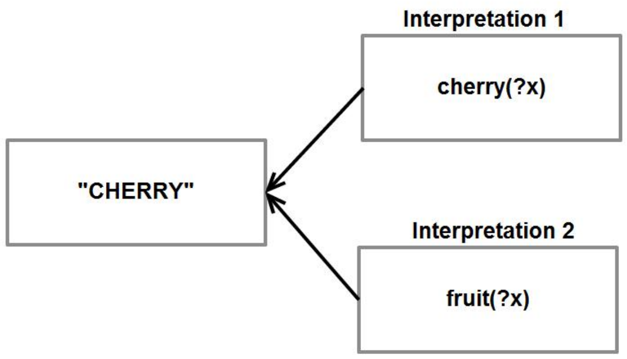


Figure 6.9: Homonymy as represented in the semiotic landscape.

To give a measure to these aspects of production the notion of a node's *out-degree* is crucial. The effective out-degree of a node is related to the number of its outgoing edges but takes into account the weights of the edges.
Generally it is assumed that the weights of the outgoing edges are normalized

such that their sum equals 1, that is true for our case. It means that if there are k edges each with an equal probability $\frac{1}{k}$, then the effective out-degree equals k. If however the k edges have differing probabilities (weights), then the effective out-degree should be smaller. The effective out-degree $k^{'}$ is not generally integer. Therefore the *effective out-degree of a node is the number of edges with equal probability needed for a hypothetical node to have the same associated Shannon information as the original node* (Beule et al.). More precisely, if a node has k outgoing edges with weights x_i, $1 <= i <= k$, its Shannon information is given by

$$\sum_{i=1}^{k} -x_i \log x_i$$

A node with $k^{'}$ outgoing edges with equal probability $\frac{1}{k^{'}}$ thus has a Shannon information of $\log(k^{'})$. By definition this information should be equal to the information associated with the original node, from which it follows that

$$k^{'} = \exp\left(\sum_{i=1}^{k} -x_i \log x_i\right)$$

As an example, the effective out-degree of m_1 in figure 6.6 is 2.97 and of w_3 it is 1.38.

In Beule et al. autors compute the out-degree respect synonyms and homonymys as follow:

- The synonymy of a semiotic graph is defined as the average of the effective out-degree of its meaning nodes minus 1. The synonymy of the graph in figure 6.6 is thus 0.87.
- The homonymy of a semiotic graph is defined as the average of the effective out-degrees of the word nodes minus 1. The homonymy of the graph in figure 6.6 is thus 0.27.

We can assert that *a gazetteer correspondent to an ontology entity is the list of synonyms to express the entity itself* and *the set of gazetteers that corresponds to a same entity contains homonyms to express this entity.* The first assumption is almost obvious, infact when we create the list of the way-of-say for the entity, we compute terms to express the meaning of the entity in different manner and hence we consider these terms to be synonyms. For the second assumption we note that, for example, a class in the ontology can have more relations or properties, and for each of them the class may play different semantic roles and so have different gazetteers. The words or the sentences from these different gazetteers that are syntactically equals between them are homonyms because they represent with the same form different meaning.

We have automatized the procedure for computing the degree of synonyms and homonyms and obvioulsy we have observed that they depend from how the domain is modeled. These measures tell us something about the compositionality of the lexicon. When there are high synonyms and homonyms out-degrees there is high probability that system can produce more utterances for a same ontology event and so system could become ambiguous. So, evaluating these measure we can expect more or less unambiguous utterances.
In particular, for the MontePaschi ontology we have observed that there are not many problems related to the synonyms and homonyms management, because the terms contained in the ontology are more technical and the list of synonyms and homonyms is ristrictly. Moreover we do not include external documents in the production. How to implement strategies that basing on these measures take actions to improve the quality of the production are future works.

The representation through the semiotic graph can also help us to evaluate the compositionality of the lexicon by system during production, that is the way on which system combines lexical constructions using grammatical ones to produce something. To that end we have to consider the problem of the *correlations* between the predicates as they occur in a description to verbalize.
We define the compositionality of a language as *the average number of predicates covered per word*. We know that every lexical entry associates a word with a number of predicates in its meaning according to a certain probability.
For atomic entries, the meaning pole contains only a single predicate. The meaning poles of not-atomic entries contain three predicates. If we define *holistic entries* as the conjunction of more topics that are the combination of atomic and not-atomic

entries, the meaning poles of these entries contain several predicates. Hence, we would like to define the *degree of compositionality* of a lexical entry as the reciprocal of the number of predicates in its meaning.

Words are only associated with a meaning according to the probability we have computed. Therefore, the compositionality of a word is defined as the reciprocal of the weighted sum of the number of predicates in all it's associated meanings, with weights the probability of the word having the particular meaning.
For example, if the word *w** is associated with the meanings m_1, m_2 and m_3 according to the probabilities p_1, p_2 and p_3 respectively (with $p_1 + p_2 + p_3 = 1$), and if the number of predicates in meaning m_i is given by $|m_i|$ ($i \in \{1, 2, 3\}$), then the compositionality *comp(w*)* is given by:

$$comp(w*) = \frac{1}{p_1 * |m_1| + p_2 * |m_2| + p_3 * |m_3|}$$

Next, we define the compositionality of system's lexicon as the average compositionality of all the words it contains. Summarily we can tell that the production by the system is more verbose and correct if the size of the graph and the number of lexicon are high.

In figure 6.10 is represented the precision of the production, defined in this case as follow:

$$P = \frac{CorrectProduction}{CorrectProduction + SpuriosProduction}$$

where Correct Production are the number of sentences produced by system that are true, that is they are sentences that express real factsl in the domain (as for example that a specific Bank has its ABI Code, that is a fact modeled in the ontology). Spurios Production are the number of sentences that are not real. How these relevations depend from previous observations are future works too.

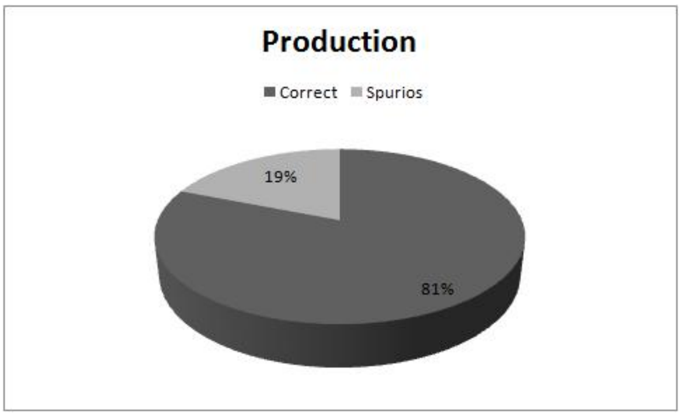


Figure 6.10: Precision of production in our relevations.

Chapter 7

Conclusions

In this thesis an innovative model for Natural Language Interfaces is proposed along with its implementation. The main idea behind the work is achieving the integration between a formal knowledge representation like OWL and a grammatical formalism to enable both understanding and production of natural language sentences. Ontology statements can be regarded as a set of Subject-Verb-Object (SVO) triples. This inherent semantics has been defined as the *semantic seed* of a statement. Semantic seeds can be put into correspondence with the emerging syntactic structures defined by the ontology elements. Construction Grammar (CxG) has proved to be a suitable theory for the purposes of the presented work because it allows modeling such a correspondence via a proper axiomatization of the RDF language. The Fluid Construction Grammar (FCG) has been used in this work due to its implementation flexibility.

The main motivations of this work fall in the scenario of Semantic Web. Many textual unstructured contents are shared between users on the Web and often these documents are not understandable for machines. Using ontologies to model the topics dealt with in document corpora enables structuring, and improves their retrieval by automated systems. Moreover, one of the main goals of Semantic Web is the ontological representation as a way to contents standardization. Nevertheless, ontologies are written using formal languages that are impervious for unskilled users, which need to interact with such a representation for many purposes.

Integrating ontology with FCG allows to define a natural language interface to formal knowledge for external users, which can understand the domain regardless of the formal languages used to model it. On the other hand, systems are able to verbalize about the represented domain. Verbalization capability, and retrieval of external data enable the definition of an *expanded* natural language interface, because systems are able to produce utterances not only about the events coded into the knowledge base but also about external related contents.

The development of the presented system experienced different phases. The very first component of the whole system was extraction of the semantic events represented in the ontology. Semantic events are descriptions of some situations in the domain under investigation. Suitable rules for computing different ways of saying to express the same semantic situation have been developed. Many concepts are syntactically expressed in different ways depending on the context they appear, and the identification the *semantic seed* helps to disambiguate the context and to select the right syntax.

Next, understanding and production functions were developed to make system able to interface with users. Here the use of the FCG was crucial. FCG performs both these tasks using the same set of constructions. This characteristic fits well the purposes of this research. As a consequence, the axiomatization of RDF statements that models the semantic seed of a statement in the ontology has been embedded in the FCG framework to enable the composition of lexical and grammatical constructions describing the domain under investigation. The implemented system is able to produce and to parse using such constructions.

The experiments reported in this thesis demonstrate that the system performs good. It is able to understand information from free textual documents and moreover it is able to produce correct sentences about the contents. Here the term “correct” is to be intended as “reporting actually events described in the ontology”. Even if sentences can be not grammatically complete, they are perfectly understandable by humans.

Parsing is precise, in the sense that it creates right correspondences between semantic events represented into the ontology and the meaning of the textual input.

This has many side effects: the system can perform semantic annotation of the text, and it can retrieve new instances to put in the ontology. It is obvious that the quality of parsing depends on the ontology model: the more ontology is complete (i.e. it models each aspect of the domain) the more parsing achieves good results (i.e. it annotates and/or retrieves more contents).

An important aspect of the proposed methodology is that it is both ontology and language independent. Axiomatization rules depend merely on the formalism used to model the domain. The spoken language used to name the ontology element is a-priori fixed, and it forms the base lexicon used in syntactic constructions for production and parsing. In this sense the system is portable.

Summarily, a system designed according to the proposed methodology:

- is able to extract the semantic events modeled into an ontology representation actually expressed in OWL/RDF syntax. It means that the system can infer from the syntax of the representation the semantic aspects of the modeled domain;
- is able to compute the most used ways of saying to express such semantic aspects;
- is able to annotate plain text related to the domain under investigation using the rules computed in the previous item;
- is able to infer new instances from the text to enrich the ontology;
- is able to produce correct statements (in the sense outlined above) about all the events it knows.

The system is far from being complete, and many theoretical and implementation issues have to be faced to reach this goal. There is no direct interaction with users apart from answering to SPARQL queries that map onto the domain ontology. The first activity will be enabling direct parsing of the users' sentences to produce appropriate answers. This is the core aspect in a NLI. There are some issues in this respect. Direct interaction with humans implies understanding of incomplete grammatical structures that include co-references and anaphors. Such a statement cannot be mapped directly onto the knowledge representation formalism.

To this purpose, the main idea is to investigate a way of modeling anaphors in the FCG through proper grammatical constructions. Such constructions have to depend tightly on the lexical ones that are in charge of describing the concepts referred to in anaphors.

Finally, other structured knowledge sources are under investigation. The main interest is on rules for converting an entity-relation diagram (ERD) in OWL ontology to allow the system to verbalize about database contents in response to conventional SQL queries.

Appendix A

From Wikipedia to Semantic MediaWiki

A.1 Introduction

In this appendix we illustrate the scenario in which we moved first steps. Initially we aimed only to solve the problem of information retrieval from unstructured sources shared on the Web to expand the knowledge base of a system modeled through an ontology. Important resources for these purposes were represented by wikis, on which anyone shares his knowledge editing plain text, without constraints to follow. The first wiki in the world is Wikipedia, that is known as the biggest free Web encyclopedia.

Result of this phase of work was a semantic annotator, that using the ontology of the system, keep as input a free text related to the represented domain and outputted the same text enriched by tags that express the meaning of understood words.
Annotations are usefull to structure the contents and to include them in the knowledge of the system. System not only enriched its base, but showdown it was able to understand the text through the semantics of the ontology. In this chapter we describe our approach to annotate semantically Wikipedia.

Our technique can be regarded as an instance of a general framework aimed to the extraction of semantic information from a document corpus provided that a structured machine-readable description of the domain dealt with is at disposal, that is presented in the next paragraph.

A.2 The Role of Wikis in the Semantic Web

Sharing information and exchanging knowledge are the primary goals for popular wikis. Communities of editors use wikis for their purposes of collaborative data creation and retrieval on the Web. The success of wikis is mainly due to the *wiki-way* and *user-friendliness* principles. According to the first principle, an editor is not necessarily an expert, but he can write freely his contents using plain text and few markup elements (*wiki-text*); so a wiki system is simple to use and approachable to all. Moreover, user-friendliness point out the collaborative aspect of the system; editors cooperate between them to modify, update and lace existed information.

Although these aspects are been widely accepted by the web community, they lead to a big limitation to external tools aimed to use the wiki information. Data are not structured and not machine-readable; a computer does not understand the meaning of the information from a wiki that is useless. A way to obtain formalized and structured data can be semantic annotation of documents. Such technique adds textual information to the document via a tagging system. A set of tags associated to a word is used to specify its semantic meaning. Many attempts have been made in the semantic community to extract meaning from existing wiki systems: they are mainly based on documents organization in the wiki, as we'll discuss next.

We use the arrangement of a Wikipedia page to set up a hierarchical strategy aimed to save computational effort. At first, we analyze the TOC of the page turning it in an OWL fragment to be aligned with the domain ontology. The alignment enables our approach to devise only a small region in the ontology to be used to analyze the full text of each section. This phase is the core of our approach. For each ontology entity involved in a section, a set of rules describes all possible roles of its name if we use it as a word in a sentence dealing with that class/property. Rules are coded in OWL as suitable properties whose values are the description of an entity name as a part of speech in Penn Treebank notation.

They represent patterns to be searched for after POS tagging the section's text. The final step is the automatic generation of the semantic wiki code to be published on line. The particular syntax used to write both plain wiki pages and semantic wiki ones allows also replacing the original Wikipedia page with the new one where new hyperlinks have been added automatically. This hierarchical approach is merely applicable to whatever document corpus with a rigid structural layout. This is not a strong limitation because we use the structure only to reduce the number of rules to be tested in text processing.

A.3 Wikipedia in Action

In 2001 Jimbo Wales and Larry Sanger created a wiki that would have become in a few years the most popular on line encyclopedia in the world: Wikipedia. Wikipedia is an authoring and content management system, which contains thousands of hyperlinked documents written by editors from different countries in more than two hundred languages as English, German, French and Japanese. Since 2005, documents in Wikipedia have grown exponentially (Wales (2005)), pointing out the great participation of authors with different cultures, experiences, and interests. Nowadays, the encyclopedia contains more than 10 million articles, as reported in (Giles (2005)).

Wikipedia is based on MediaWiki, available at *http://www.mediawiki.org*, a free software which was developed initially by the Wikipedia community for exclusive use. MediaWiki is now a generalpurpose platform, which is used in several wiki sites as well. MediaWiki defines an environment where authenticated users can write contents either creating new pages or updating existing ones, while external users can search for information and read articles without limitations. The primary method for entering information into a wiki based on MediaWiki is a *wiki-window*. Here, editors write their articles using a simple markup language that in turn is transformed into XHTML pages. Accordingly, the window already provides many facilities to format pages and to structure documents as the definition of hyperlinks between two pages. Hyperlinks are fundamental for navigating a wiki: they define a generic relation between pages but do not specify the kind of such relation. A hyperlink can be regarded as a some relation in this respect.

MediaWiki uses a rigid internal structure to organize the documents using a system of namespaces: a namespace is a prefix in the name of the page. In the case of Wikipedia, articles are categorized through namespaces that distinguish different kinds of pages according to their aim. Examples of namespace are "User:" followed by a page name for user homepages, "Help:" for documentation pages, ". . . talk:" for various kinds of discussion pages, and so on. Editors cannot define namespaces freely. Indeed, they depend on the configuration settings of Wikipedia.

The idea of Wikipedia is to allow everyone to edit and extend the encyclopedic content, according to the wiki-way principle. Problems of this principle are synonymy and homonymy in the titles: several different pages for the same subject may emerge in a decentralized editing process. MediaWiki has a redirect mechanism by which a page can be caused to forward all requests directly to another page. Although the wiki's language is very simple, and editors are encouraged by its ease of use to insert novel contents, the wiki-way mode does not directly enforce any restrictions and there are many ambiguities in the interpretation of information by automatic applications. Searching knowledge in Wikipedia means reading the pages and choosing the documents that satisfy the need of the researcher. An external automatic tool has not the ability to interpret data correctly as humans have.

Information in Wikipedia is not processable by a machine and it must be formalized suitably to this aim. In recent years semantic technologies community has been interested to this problem and many attempts have been made to extract semantics from existing wiki sources of unstructured information. A straightforward approach should be rewriting manually contents following the conventions about semantic annotation imposed from the Wikipedia Community. Such a technique requires a very huge effort and has been never considered.

A.4 Semantic Wikipedia

The first attempt to give semantics to Wikipedia contents was made by its founders, which imposed editors to classify their documents using categories. Unlike MediaWiki namespace organization, an editor can create new categories. A Wikipedia category contains articles about a specific topic: the *"Movies Category"* includes all articles about films or movies. Each category may have one or more subcategories that specialize articles classification. *"1960 Movies Category"* includes all films in 1960 and *"Italian Movies Category"* is the set of articles about Italian films; both are subcategories of *"Movies Category"*. Authors must classify their pages inserting them in a specific category or subcategory, and each category is associated with a page by the "Category:" namespace. Category classification is a taxonomical one, because each subcategory is an element of its super-category. Also articles in the categories are linked between them, according to the MediaWiki facilities for structuring pages: two articles are linked if the former refers to the other in its plain text.

Hyperlinks in the documents weave a net of relations between articles on the taxonomical category classification, and attempts were made in many works to extract a semantic structure from such a network, as in Kozlova (2005) or Nastase and Strube (2008).
The first approach fills an ontology creating classes, instances and relations between them. Classes are the categories, instances are the articles (an instance for each article) for the corresponding class/category and relations are the hyperlinks between articles and categories.
In the second paper, four types of categories are defined: explicit relation categories, partly explicit relation categories, implicit relation category and class attribute categories. Authors use these category types to identify the dominant constituent, and to extract relations and classes with their attributes.

All these approaches assume that the semantic meaning of wiki documents depends only on the category classification and hyperlinks structure, so data are organized in ontology to be used by external automatic tools. Such approaches suffer from the inherent non-static nature of wiki articles: the ontology has to be

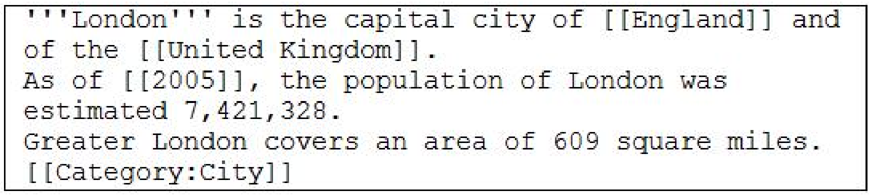
updated repeatedly. Moreover, many information items are not inserted in the ontology because they are not hyperlinked leading to loss of data.
Other important approaches add further structure to MediaWiki by means of textual annotation of the wiki's content. These approaches define a new research field in the semantic technology leading to Semantic MediaWiki (SMW). SMW extends the popular MediaWiki engine to build semantic wikis.

A.4.1 Semantic MediaWiki Annotation Rules

SMW is free software acting as an extension of the MediaWiki engine: SMW registers for specific events or requests, and MediaWiki calls SMW procedures when needed.
SMW does not overwrite any content created using MediaWiki, so it can be added to existing wikis without computational load for data relocation. The goal of SMW is to enrich data with semantics during the insertion of text in the wiki-window via special tags annotations; it is a novel type of wiki known as semantic wiki. If Sematic MediaWiki is integrated in an existing wiki, semantics is associated only to novel contents, and it does not structure existing data. An interesting problem is to add semantics to existing information.

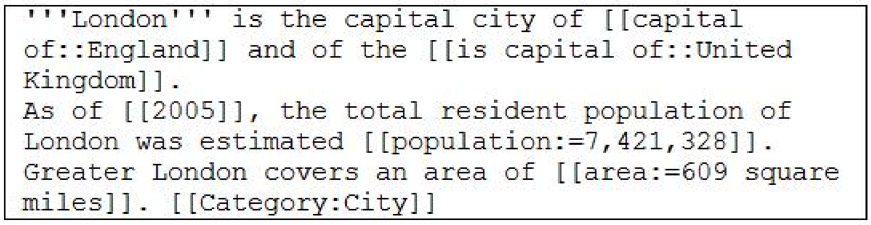
SMW's annotation mechanism is based on standard Semantic Web formalisms. As a consequence, binary properties are the central expressive mechanism. A binary property links an entity with other one or with a data value. Binary properties in SMW are implemented adopting a page-centric perspective where properties augment the page's contents in a structured way by adding proper tags to represent them. Semantic MediaWiki has not a standard set of annotation rules. In our work we used the most widely accepted set of rules that are reported in Völkel et al. (2006). In the cited work, authors propose integration in Wikipedia to allow specifying types for relations between articles and for data inside each article in an easy-to-use manner, using the Semantic MediaWiki engine. In the article an example shows a semantic interpretation of the Wikipedia page about London city. We import the same example and the related figures.

Figure A.1 (the figure 3 in that paper) presents a fragment of the source text



```
'''London''' is the capital city of [[England]] and
of the [[United Kingdom]].
As of [[2005]], the population of London was
estimated 7,421,328.
Greater London covers an area of 609 square miles.
[[Category:City]]
```

Figure A.1: Source of page about London city in Wikipedia using MediaWiki engine.



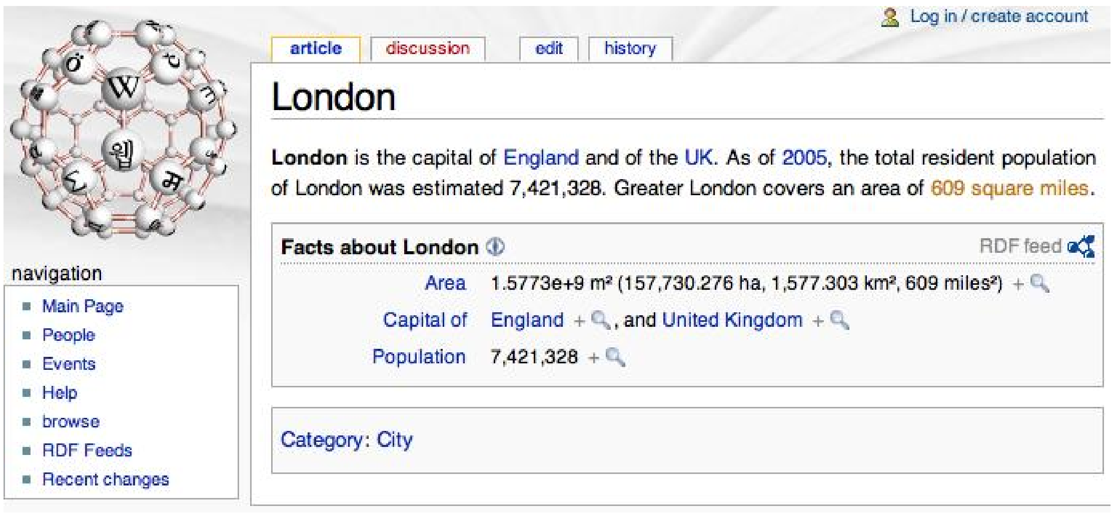
```
'''London''' is the capital city of [[capital
of::England]] and of the [[is capital of::United
Kingdom]].
As of [[2005]], the total resident population of
London was estimated [[population:=7,421,328]].
Greater London covers an area of [[area:=609 square
miles]]. [[Category:City]]
```

Figure A.2: Source of page about London city in Wikipedia using Semantic MediaWiki engine.

that describes London city using MediaWiki (traditional Wikipedia). In MediaWiki quotes are used to bold the text between it; double square brackets define a hyperlink between the page and the other wiki page whose name is inside them, without specification of the semantic meaning for the link. At the end of the page, the [[Category:City]] square brackets link the page with the City Category, according to namespaces mechanism described above. Semantics is associated to such a text via the annotation rules that are shown in the example of Figure A.2 (figure 4 in Völkel et al. (2006)).

Markup elements have been extended. In the example the link [[England]] becomes [[capital of::England]] to assert that "London" concept has a binary property called "capital of" with the "England" concept. Also there is an interesting annotation that involves data values that are not hyperlinks, as the population number. Annotation rules impose using the same markup elements in both cases. To avoid that a data value is formatted as hyperlink, authors decide that users must first declare the property and specify its type. In the case of property "population" user must declare that it has a numerical type. If a property is not declared yet, then

SMW assumes to default that its value denotes wiki pages, so that annotations will become hyperlinks. The resulting page is displayed in Figure A.3 (figure 2 in Völkel et al. (2006)).



London

London is the capital of England and of the UK. As of 2005, the total resident population of London was estimated 7,421,328. Greater London covers an area of 609 square miles.

Facts about London RDF feed

Area	1.5773e+9 m² (157,730.276 ha, 1,577.303 km², 609 miles²) +
Capital of	England +, and United Kingdom +
Population	7,421,328 +

Category: City

Figure A.3: Semantic MediaWiki page about London city.

A.5 Semantic Annotation Strategy

Our framework for semantic annotation of plain text is a three step hierarchical approach:

- Document structure analysis to devise structure;
- Paragraph Text analysis through linguistic rules;
- Generation of semantic annotation.

This hierarchical approach is used to have a reduced set of rules to be applied in full text analysis and to reduce the computational effort. The first step looks at the document layout and tries to devise a rough list of the main concepts the document deals with. In this way, we select a limited portion of the OWL domain ontology to be used in the next step.

The second step is the core of the whole framework and it represents the true semantic information extraction phase. Here, complete binary properties are extracted from plain text (relation name and property data values). In this approach, the name of the class and of the relations of the domain ontology is used as a reference for a suitable set of rules for text analysis that are plugged directly into the ontology. Rules use the Penn Treebank notation to describe the possible roles played by the name of the class/relation in a statement dealing with it. Knowledge about the structure of such sentences is not needed. Such rules are simply patterns to be searched for when the text has been processed with a POS tagger.
The third step is a mere generation of the annotated text according to a suitable tagging system. We finalized the general framework to semantic annotation of Wikipedia pages with the following three-step procedure:

- inferring the semantic meaning of the Wikipedia sections using all the domain ontology;
- inferring the semantic meaning of the text in a single Wikipedia section using only the concepts and relations obtained in the first step;
- annotation of contents and rewriting in the online semantic wiki.

A.5.1 Inferring the Semantic Meaning of the Wikipedia Sections

MediaWiki engine allows organizing the contents of a page in sections and subsections that could be defined as the traditional paragraphs of a chapter in a book. The focus of a section is a particular argument that is part of the page's topic as the Biography Section in a Person page. Sections are be indexed in a table, that is a template for the page: the *Table of Content* (TOC).

Each Wikipedia page has a TOC, which is not imposed from MediaWiki. Most editors use it to organize their content pages. TOC is a tree structure where the root is the category of the page and each node is a section. Nodes are organized as taxonomy according to the section structure. We use the TOC to partition the text to be searched for commonly used sentence patterns. Each section is processed using only the most appropriate patterns with respect to its title, which is assumed to be the topic dealt with in the section itself. The TOC provides us also with information about the way concepts in the domain ontology are treated in the page.

A suitable crawler extracts the TOC from Wikipedia page and then it's transformed in OWL fragment (TOC ontology). TOC ontology is then mapped to the domain ontology to extract relevant concepts that correspond to the page's sections. Mapping can be achieved by applying alignment methods like QOM Ehrig and Staab (2004) or PROMPT Noy and Musen (2003). The base idea of these methods is the definition of similarity measures, which express how two entities from different ontology structures are similar. Similarity can be expressed both in terms of the meaning of the two entities and comparing the tree topology surrounding them.
More formally, an alignment between ontology O and O' is the quadruple *<e, e', s, n>* where e is an element in *O*, *e'* is an element in *O'*, *s* is the relation between elements that defines the similarity, and $n \in [0..1]$ is the level of confidence of such relation.

The framework we use to map ontology is FOAM (Framework for Ontology Alignment and Mapping) Ehrig (2005). FOAM takes two ontology structures as its inputs and returns a list of concept couples with the corresponding *n* value. This number is obtained by iterating a process that computes and aggregates many similarity measures. The output list by FOAM is filtered according to a suitable threshold defining the minimum acceptable similarity. At the end of the alignment process, the system knows which concepts of the domain ontology are dealt in the page and the semantic sense of each section has been inferred.

A.5.2 Inferring the Semantic Meaning of the Text in a single Wikipedia Section

For each section, the system can infer semantic information from the text in each section using the name of the relations in the ontology directly. For each concept extracted at the end of the first step, it takes the name of object properties (properties between classes) and data-type properties (relations between classes and data values). Then the system searches for all these properties in the text directly; the instances of the relations are inferred through their role in the sentences. As a consequence, the system must know which part of speech is used to describe entities of each relation, regardless they are either classes or a data values.

As we will show in the next paragraph, we designed an application that implements the framework in the Arts domain that is described using OWL. If we want to know who is the mother of an artist, in the appropriate section we can find the word mother (which is the name of the relation between Artist and Mother concepts in the domain ontology) and we can obtain the actual name searching for a proper name in the same sentences. If we want to know a birth date, we can search the word birth in the appropriate section and we can infer the date by searching for a set of possible patterns regarding dates in the rest of the sentence. This strategy requires the use of a POS tagger to enrich the ontology with all the roles of an entity name used as a word in the sentence patterns. In our framework we have used the Stanford NLP POS tagger that returns the phrase structure as a tree. In turn, the tree can be used to classify the main components according to the Penn Treebank English POS tag set (Marcus et al. (1993)).

Object and data-type properties of the domain ontology are both annotated with tag patterns that specify the role of the relation's entities in the text. In our framework, the process is iterated to validate the results. In some cases the system uses the Text Runner Search Engine Banko and Etzioni (2008) to verify that the triple (entity, relation, entity) is true. Text Runner Search Engine is an online service that extracts an entity in a relation from open information sources, given the relation itself and the second entity. It uses a methodology based on the analysis of the structure of English sentences. We use it only to disambiguate the output of the system and to verify its correctness. However, our system is able to return more relations than the ones returned by this service.

A.5.3 Automatic Annotation of Wikipedia Page

The last stage is the production of the semantic annotated Wikipedia page to be published on a semantic wiki. At the end of the two previous phases the system has discovered in the text a list of instances of the OWL properties that are used to describe the domain ontology. Moreover, the ontology connects such entities through properly named relations that can be coded in the resulting semantic wiki page. The innovation in this work is that such relations are derived independently from the original hyperlink structure. In fact, the system extracts semantic information also from non-hyperlinked text.

Provided that properties that have not been suitably declared in a semantic wiki default to hyperlinks, the same output code can be reused to replace the original Wikipedia page adding all the annotations as hyperlinks. In this scenario, if an editor does not create a connection between two Wikipedia pages that are related in a conceptual way, our framework allows to create it automatically. In our framework implementation, a suitable web application interfaces with the online semantic wiki http://semanticweb.org/ and writes in it the same text of the original Wikipedia page enriched with the annotations. Detailed architectural design of the implementation is reported in the description of the application scenario.

A.6 Application Scenario

Figure A.4 shows the architectural view of the proposed framework finalized to semantic sense extraction from Wikipedia pages. The three stages of our approach are implemented as follows.

Step one is accomplished by the Crawler component plus the FOAM engine, the Text Analyzer coupled with a POS tagger are in charge of step two, and the Semantic Annotator carries out the third step. We implemented the Crawler, the

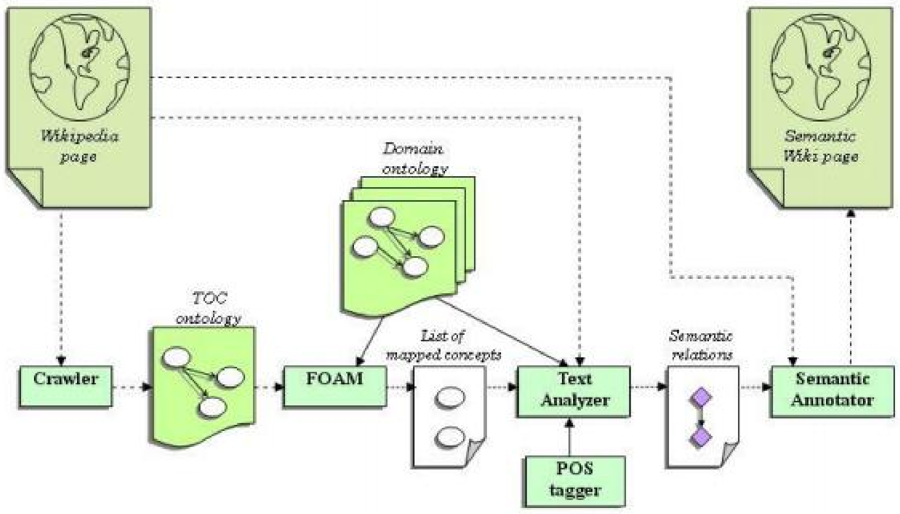


Figure A.4: Architectural design of the framework implementation.

Text Analyzer, and the Semantic Annotator as original components. The Crawler extracts the contents in a Wikipedia page and it is used to device the TOC. Relevant concepts to be used as a reference in the second step are obtained by the FOAM tool via alignment of the TOC ontology and the domain one.

The Text Analyzer searches in the appropriate sections of the Wikipedia page the names of all the relations that in the domain ontology are linked to concepts obtained in step one. Also the Text Analyzer matches the Penn Treebank patterns used to specify the part-of-speech roles played by the entities of the relations. This component outputs a list of couples formed by the name of the relation and the name of the inferred entity, as it will be clear later.

Finally, the Semantic Annotator writes the same Wikipedia page enriched with semantic annotations using the output of the Text Analyzer. In the figure, dotted arrows are data paths, while plain arrows define the use of an internal resource/module. We will describe in detail the use of the components mentioned above by means of a specific application scenario. To test and demonstrate our methodology we used the Art domain. All concepts in the ontology, their properties and the relations between them are formalized in an OWL structure.

Let's suppose that we want to semantically annotate the Wikipedia page about Gustav Klimt, the Austrian symbolist painter. Wikipedia page of the artist is at *http://en.wikipedia.org/wiki/Klimt*, that is the input of the framework. The Crawler component extracts from the page the TOC and formalizes it in the TOC ontology. TOC ontology contains only classes and subclasses, without object or data-type properties for them; there is only the is-a property to define the subsection relation. Next, system invokes FOAM to map the TOC ontology in the art domain ontology extracting the concepts of the art domain that correspond to the sections in the table.

As an example, the system infers that concept Life of the ontology corresponds to the sections "Early life and education" and "Later life and posthumous success" in the Wikipedia page. In the mapping phase, the system prioritizes the subsections rather than sections because the TOC is a taxonomical classification, so a subsection is the super-section too. Now system knows what arguments could be dealt in the sections of the page, that are the relations linked to the mapped concepts of the domain. For the concept Life an interesting OWL object property is the following:

```
<owl:ObjectProperty rdf:ID="father">
<rdfs:range rdf:resource="#Father"/>
<rdfs:domain rdf:resource="#Life"/>
<rdfs:comment="NNP"/>
</owl:ObjectProperty>
```

Such property will enable us to know who was Klimt's father. In the fourth line of the previous OWL source code, a RDF Schema comment specifics the part-of-speech role played by the co-domain entities of the relation: a singular proper noun. We plug the role of each entity in the ontology directly, to have the POS tags at disposal when an object property is selected to be searched in the text. The use of comments has the advantage to maintain the OWL-DL profile of the ontology.

Just let's suppose that in our database we do not have any information about the name of Klimt's father. The system uses fragments of OWL source code to infer this information. It searches for the relation name (father) only in the sections devised in the preceding phase. This technique avoids analyzing the full text of the page.

In the sub-section "Early life and education" the Text Analyzer searches for all the possible relations related to concept Life. As regards the Klimt's father the following statement is retrieved:

Gustav Klimt was born in Baumgarten, near Vienna, the second of seven children, three boys and four girls. All three sons displayed artistic talent early on. His father, Ernst Klimt, formerly from Bohemia, was a gold engraver.

The Text Analyzer finds father in the text, discards all other sentences and maintains only the phrase:

His father, Ernst Klimt, formerly from Bohemia, was a gold engraver.

To infer the correct co-domain entity of the relation, the Text Analyzer component invokes the POS tagger with the sentence above. The output of the process is:

His_PRP$ father_NN ,_, Ernst_NNP Klimt_NNP ,_, formerly_RB from_IN Bohemia_NNP ,_, was_VBD a_DT gold_NN engraver_NN ._.

By using the RDF Schema comment in the OWL father relation, the Text Analyzer knows that co-domain entity of the father relation must be a proper noun (a single NNP tag or a sequence of NNP tags) so Ernst Klimt is the name we were searching for. Text Analyzer outputs a list of couples each of them formed by the name of a relation and the name of the corresponding inferred instance (father, Ernst Klimt couple in this case). This list is next used by the Semantic Annotator to rewrite the same Wikipedia page enriched with semantic annotations according to semantic annotation rules presented in Völkel et al. (2006). Semantic annotator uses the name of the relation to add semantics to the correspondent instance's name. In the proposed example, it rewrites the sentence as:

His father, [[father::Ernst Klimt]], formerly from Bohemia, was a gold engrave.

Bibliography

Grigoris Antoniou and Frank van Harmelen. *A Semantic Web Primer, 2nd Edition (Cooperative Information Systems)*. The MIT Press, 2 edition, 2008. ISBN 0262012421, 9780262012423.

Michele Banko and Oren Etzioni. The tradeoffs between open and traditional relation extraction. In *Proceedings of ACL-08: HLT*, pages 28–36, Columbus, Ohio, June 2008. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/P/P08/P08-1004`.

J. Batali. The negotiation and acquisition of recursive grammars as a result of competition among exemplars. In Ted Briscoe, editor, *Linguistic Evolution through Language Acquisition: Formal and Computational Models*, chapter 5. Cambridge University Press, 2002. URL `http://www.isrl.uiuc.edu/~amag/langev/paper/batali02theNegotiation.html`.

John Batali. The negotiation and acquisition of recursive grammars as a result of competition among exemplars. In *Linguistic Evolution through Language Acquisition: Formal and Computational Models*, pages 111–172. Cambridge University Press, 1999.

Sean Bechhofer, Frank van Harmelen, Jim Hendler, Ian Horrocks, Deborah L. McGuinness, Peter F. Patel-Schneider, and Lynn Andrea Stein. OWL Web Ontology Language Reference. Technical report, W3C, http://www.w3.org/TR/owl-ref/, February 2004.

Benjamin K. Bergen and Nancy Chang. Embodied construction grammar in simulation-based language understanding. Technical report, EDS): CON-

STRUCTION GRAMMAR(S): COGNITIVE AND CROSS-LANGUAGE DIMENSIONS. JOHN BENJAMIN PUBL CY, 2003.

Joachim De Beule and Luc Steels. Hierarchy in fluid construction grammar. In Furbach U., editor, *Proceedings of KI-2005*, number 3698 in Lecture Notes in AI, pages 1–15, Berlin, 2005. Springer-Verlag. doi: 10.1007/11551263_1. URL `http://www.isrl.uiuc.edu/~amag/langev/paper/joachim05hierar_fluid_construc_gramm.html`.

Joachim De Beule, Bart De Vylder, and Tony Belpaeme. A cross-situational learning algorithm for damping homonymy in the guessing game. In *Logic and Machines*.

Harold Boley. The rule markup language: Rdf-xml data model, xml schema hierarchy, and xsl transformations. In Oskar Bartenstein, Ulrich Geske, Markus Hannebauer, and Osama Yoshie, editors, *INAP (LNCS Volume)*, volume 2543 of *Lecture Notes in Computer Science*, pages 5–22. Springer, 2001. ISBN 3-540-00680-X.

Dan Brickley and Libby Miller. FOAF Vocabulary Specification 0.97. Namespace document, January 2010. URL `http://xmlns.com/foaf/spec/20100101.html`.

P. Buitelaar, P. Cimiano, P. Haase, and M. Sintek. Towards Linguistically Grounded Ontologies. In Lora Aroyo, Paolo Traverso, Fabio Ciravegna, Philipp Cimiano, Tom Heath, Eero Hyv'onen, Riichiro Mizoguchi, and Eyal Oren and, editors, *Proceedings of the 6th European Semantic Web Conference (ESWC)*, pages 111–125. Springer, Berlin, 2009. ISBN 978-3-642-02120-6.

Angelo Cangelosi and Domenico Parisi. Computer simulation: A new scientific approach to the study of language evolution. In Angelo Cangelosi and Domenico Parisi, editors, *Simulating the Evolution of Language*, chapter 1, pages 3–28. Springer Verlag, London, 2002. URL `http://www.isrl.uiuc.edu/~amag/langev/paper/cangelosi01computerSimulation.html`.

Philipp Cimiano, Peter Haase, and Jörg Heizmann. ORAKEL: a portable natural language interface to knowledge bases. Technical report, Institute AIFB, University of Karlsruhe, 2007. URL `http://www.aifb.uni-karlsruhe.de/Publikationen/showPublikation?publ_id=1439`.

Peter Clark, Phil Harrison, Tom Jenkins, John Thompson, and Rick Wojcik. Acquiring and using world knowledge using a restricted subset of english. In *In FLAIRS 2005*, pages 506–511, 2005.

Anne Cregan, Rolf Schwitter, and Thomas Meyer. Sydney owl syntax - towards a controlled natural language syntax for owl 1.1. In Christine Golbreich, Aditya Kalyanpur, and Bijan Parsia, editors, *OWLED*, volume 258 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2007.

W. Croft. *Syntactic categories and grammatical relations: the cognitive organization of information*. University of Chicago Press, 1991. ISBN 9780226120904. URL `http://books.google.it/books?id=h1gLdOkH1GgC`.

William Croft and Alan D. Cruse. Polysemy: the construal of sense boundaries. In *Cognitive Linguistics*, chapter 5, pages 109–140. Cambridge University Press, 2004.

David Crystal. *The Cambridge Encyclopedia of Language*. Cambridge University Press, Cambridge, 2. edition, 1998.

Hamish Cunningham, Diana Maynard, Kalina Bontcheva, Valentin Tablan, Niraj Aswani, Ian Roberts, Genevieve Gorrell, Adam Funk, Angus Roberts, Danica Damljanovic, Thomas Heitz, Mark A. Greenwood, Horacio Saggion, Johann Petrak, Yaoyong Li, and Wim Peters. *Text Processing with GATE (Version 6)*. 2011. ISBN 978-0956599315. URL `http://tinyurl.com/gatebook`.

Danica Damljanovic, Milan Agatonovic, and Hamish Cunningham. FREyA: an Interactive Way of Querying Linked Data using Natural Language. In *Proceedings of 1st Workshop on Question Answering over Linked Data (QALD-1), Collocated with the 8th Extended Semantic Web Conference (ESWC 2011)*, Heraklion, Greece, June 2011. URL `http://www.sc.cit-ec.uni-bielefeld.de/sites/www.sc.cit-ec.uni-bielefeld.de/files/proceedings.pdf`.

Marc Ehrig. Y.: Foam - framework for ontology alignment and mapping; results of the ontology alignment initiative. In *Proceedings of the Workshop on Integrating Ontologies. Volume 156., CEUR-WS.org (2005) 72-76*, pages 72–76, 2005.

Marc Ehrig and Steffen Staab. Qom - quick ontology mapping. In Sheila A. McIlraith, Dimitris Plexousakis, and Frank van Harmelen, editors, *International Semantic Web Conference*, volume 3298 of *Lecture Notes in Computer Science*, pages 683–697. Springer, 2004. ISBN 3-540-23798-4. URL `http://dblp.uni-trier.de/db/conf/semweb/iswc2004.html#EhrigS04`.

Christiane Fellbaum, editor. *WordNet An Electronic Lexical Database.* The MIT Press, Cambridge, MA ; London, May 1998. ISBN 978-0-262-06197-1. URL `http://mitpress.mit.edu/catalog/item/default.asp?ttype=2&tid=8106`.

Charles Fillmore, Paul Kay, and Catherine O'Connor. Regularity and idiomaticity in grammatical constructions: The case of let alone. *Language*, 64:501–538, 1988.

Charles J. Fillmore. Syntactic intrusions and the notion of grammatical construction. In Mary Niepokuj, Mary VanClay, Vassiliki Nikiforidou, and Deborah Feder, editors, *Proceedings of the Eleventh Annual Meeting of the Berkeley Linguistics Society*, University of California, Berkeley, 1985. Berkeley Linguistics Society.

Jenny Rose Finkel and Christopher D. Manning. Joint parsing and named entity recognition. In *Proceedings of the North American Association of Computational Linguistics (NAACL 2009)*, 2009. URL `pubs/joint-parse-ner.pdf`.

Jenny Rose Finkel, Trond Grenager, and Christopher Manning. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, ACL '05, pages 363–370, Stroudsburg, PA, USA, 2005. Association for Computational Linguistics. doi: http://dx.doi.org/10.3115/1219840.1219885. URL `http://dx.doi.org/10.3115/1219840.1219885`.

Günther Fliedl, Christian Kop, and Jürgen Vöhringer. Guideline based evaluation and verbalization of owl class and property labels. *Data Knowl. Eng.*, 69:331–

342, April 2010. ISSN 0169-023X. doi: http://dx.doi.org/10.1016/j.datak.2009.08.004. URL `http://dx.doi.org/10.1016/j.datak.2009.08.004`.

Norbert E. Fuchs, Kaarel Kaljurand, and Tobias Kuhn. Attempto Controlled English for Knowledge Representation. In Cristina Baroglio, Piero A. Bonatti, Jan Małuszyński, Massimo Marchiori, Axel Polleres, and Sebastian Schaffert, editors, *Reasoning Web, Fourth International Summer School 2008*, number 5224 in Lecture Notes in Computer Science, pages 104–124. Springer, 2008.

Jim Giles. Internet encyclopaedias go head to head. *Nature*, 438(7070):900–901, 2005. ISSN 0028-0836. URL `http://dx.doi.org/10.1038/438900a`.

A. E. Goldberg. *Constructions: A construction grammar approach to argument structure*. University of Chicago Press, Chicago, 1995. URL `http://www.isrl.uiuc.edu/~amag/langev/paper/goldberg95constructionsA.html`.

Adele E Goldberg. The nature of generalization in language adele e. goldberg. *The nature of generalization in language*, (Goldberg):1–30, 2006. URL `http://lingo.stanford.edu/sag/papers/ag09.pdf`.

Adele E Goldberg and Devin Casenhiser. *Learning Argument Structure Constructions*, pages 185–204 ST – Learning Argument Structure Construc. Center for the study of Language and Information, 2006.

R. Guha and R. McCool. Tap: An semantic web test-bed. *Web Semantics: Science, Services and Agents on the World Wide Web*, 1(1), 2011. ISSN 1570-8268. URL `http://www.websemanticsjournal.org/index.php/ps/article/view/28`.

T. Hashimoto and T. Ikegami. Emergence of net-grammar in communicating agents. *Biosystems*, 38(1):1–14, 1996. doi: 10.1016/0303-2647(95)01563-9. URL `http://www.isrl.uiuc.edu/~amag/langev/paper/hashimoto96emergenceOf.html`.

Daniel Hewlett, Aditya Kalyanpur, Vladimir Kolovski, and Christian Halaschek-wiener. Effective nl paraphrasing of ontologies on the semantic web (technical report. Technical report.

Willem-Olaf Huijsen. Controlled language - an introduction.

Carlos A. Hurtado, Alexandra Poulovassilis, and Peter T. Wood. Ranking approximate answers to semantic web queries. In *Proceedings of the 6th European Semantic Web Conference on The Semantic Web: Research and Applications*, ESWC 2009 Heraklion, pages 263–277, Berlin, Heidelberg, 2009. Springer-Verlag. ISBN 978-3-642-02120-6. doi: http://dx.doi.org/10.1007/978-3-642-02121-3_22. URL `http://dx.doi.org/10.1007/978-3-642-02121-3_22`.

Eero Hyvönen, Eetu Mäkelä, Mirva Salminen, Arttu Valo, Kim Viljanen, Samppa Saarela, Miikka Junnila, and Suvi Kettula. Museumfinland – finnish museums on the semantic web. *Journal of Web Semantics*, 3(2):25, 2005.

Thomas Wasow Ivan A. Sag and Emily M. Bender. *Syntactic Theory: A Formal Introduction, 2nd edition.* CSLI, 2003.

Ray S. Jackendoff. *Semantic Structures.* Cambridge: MIT Press, 1990.

Mustafa Jarrar, C. Maria, and Keet Paolo Dongilli. Multilingual verbalization of orm conceptual models and axiomatized ontologies. Technical report, 2006.

Kaarel Kaljurand. *Attempto Controlled English as a Semantic Web Language.* PhD thesis, Faculty of Mathematics and Computer Science, University of Tartu, 2007.

Esther Kaufmann and Abraham Bernstein. How useful are natural language interfaces to the semantic web for casual end-users? In *ISWC/ASWC*, pages 281–294, 2007.

Paul Kay and Charles J. Fillmore. Grammatical constructions and linguistic generalizations: The What's X Doing Y? construction. *Language*, 75(1):1–33, 1999.

R. I. Kittredge. *Sublanguages and controlled languages.*

Holger Knublauch, Ray W. Fergerson, Natalya F. Noy, and Mark A. Musen. The protégé owl plugin: An open development environment for semantic web applications. pages 229–243. Springer, 2004.

Natalia Kozlova. *Automatic Ontology Extraction for Document Classification*. PhD thesis, Saarland University, 2005.

Markus Krötzsch, Denny Vrandečić, and Max Völkel. Semantic MediaWiki. In Isabel Cruz, Stefan Decker, Dean Allemang, Chris Preist, Daniel Schwabe, Peter Mika, Mike Uschold, and Lora Aroyo, editors, *The Semantic Web - ISWC 2006*, volume 4273 of *Lecture Notes in Computer Science*, chapter 68, pages 935–942. Springer Berlin / Heidelberg, Berlin, Heidelberg, 2006. ISBN 978-3-540-49029-6. doi: 10.1007/11926078_68. URL `http://dx.doi.org/10.1007/11926078_68`.

G Lakoff. The invariance hypothesis: is abstract reason based on image-schemas? *Cognitive Linguistics*, 1(1):39–74, 1990.

Ronald W. Langacker. *Grammar and Conceptualization*. Walter De Gruyter, 1999. URL `http://www.isrl.uiuc.edu/~amag/langev/paper/langacker99grammarAnd.html`. Cognitive Linguistics Research, 14.

Ronald W. Langacker. *Concept, Image, and Symbol: the Cognitive Basis of Grammar*, volume 1 of *Cognitive Linguistic Research*. Walter de Gruyter, second edition, December 2002. ISBN 3110172801. URL `http://www.amazon.co.uk/exec/obidos/ASIN/3110172801/citeulike00-21`.

Vanessa Lopez, Victoria Uren, Enrico Motta, and Michele Pasin. Aqualog: An ontology-driven question answering system for organizational semantic intranets. *Web Semantics: Science, Services and Agents on the World Wide Web*, 5(2), 2011. ISSN 1570-8268. URL `http://www.websemanticsjournal.org/index.php/ps/article/view/113`.

Frank Manola and Eric Miller. RDF primer. W3C recommendation, W3C, February 2004. URL `http://www.w3.org/TR/rdf-primer`. Published online on February 10th, 2004 at `http://www.w3.org/TR/2004/REC-rdf-primer-20040210/`.

Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. Building a large annotated corpus of english: the penn treebank. *Comput. Lin-*

guist., 19:313–330, June 1993. ISSN 0891-2017. URL `http://portal.acm.org/citation.cfm?id=972470.972475`.

Paul Martin, Douglas E. Appelt, Barbara J. Grosz, and Fernando Pereira. Team: an experimental transportable natural-language interface. In *Proceedings of 1986 ACM Fall joint computer conference*, ACM '86, pages 260–267, Los Alamitos, CA, USA, 1986. IEEE Computer Society Press. ISBN 0-8186-4743-4. URL `http://dl.acm.org/citation.cfm?id=324493.324576`.

B McBride. Jena: a semantic web toolkit, 2002. URL `http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1067737`.

James W. Minett and William S.-Y. Wang, editors. *Language Acquisition, Change and Emergence: Essays in Evolutionary Linguistics*. City University of Hong Kong Press, July 2005. URL `http://www.isrl.uiuc.edu/~amag/langev/paper/minett_wang2004editedBook.html`.

V. Nastase and M. Strube. Decoding wikipedia categories for knowledge acquisition. In *Proceedings of the 23rd AAAI Conference on Artificial Intelligence*, Chicago, 2008.

Natalya F. Noy and Mark A. Musen. The prompt suite: Interactive tools for ontology merging and mapping. *International Journal of Human-Computer Studies*, 59(6)(6):983–1024, 2003.

Eric Nyberg and Teruko Mitamura. The kantoo machine translation environment. In *AMTA*, pages 192–195, 2000.

Emanuele Pianta, Luisa Bentivogli, and Christian Girardi. Multiwordnet: developing an aligned multilingual database. In *Proceedings of the First International Conference on Global WordNet*, January 2002. URL `http://multiwordnet.fbk.eu/paper/MWN-India-published.pdf`.

Roberto Pirrone, Vincenzo Cannella, Orazio Gambino, Arianna Pipitone, and Giuseppe Russo. Wikiart: An ontology-based information retrieval system for arts. In *ISDA'09*, pages 913–918, 2009.

Carl Pollard. *Lectures on the foundations of Hpsg.* Ohio State University, 1997.

Carl Pollard and Ivan Sag. *Head-Driven Phrase Structure Grammar.* Chicago University Press, Chicago, Illinois, 1994.

Borislav Popov, Atanas Kiryakov, Angel Kirilov, Dimitar Manov, Damyan Ognyanoff, and Miroslav Goranov. Kim semantic annotation platform. *Journal of Natural Language Engineering*, 10(3-4):375–392, September 2004. URL `http://www.ontotext.com/publications/KIM_SAP_ISWC168.pdf`.

Eric Prud'hommeaux and Andy Seaborne. SPARQL query language for RDF. Technical report, 1 2008. URL `http://www.w3.org/TR/2008/REC-rdf-sparql-query-20080115/`.

Josef Ruppenhofer, Michael Ellsworth, Miriam R. L. Petruck, Christopher R. Johnson, and Jan Scheffczyk. FrameNet II: Extended theory and practice. Technical report, ICSI, 2005. URL `http://framenet.icsi.berkeley.edu/book/book.pdf`.

Giuseppe Russo, Roberto Pirrone, and Arianna Pipitone. Acquisition of new knowledge in tutorj, 2009. URL `http://aaai.org/ocs/index.php/FSS/FSS09/paper/view/939`.

Rolf Schwitter and Marc Tilbrook. Meaningful web annotations for humans and machines using controlled natural language. *Expert Systems*, 25(3):253–267, 2008.

Andrew D. M. Smith. *Evolving Communication through the Inference of Meaning.* PhD thesis, Theoretical and Applied Linguistics, School of Philosophy, Psychology and Language Sciences, The University of Edinburgh, 2003. URL `http://www.isrl.uiuc.edu/~amag/langev/paper/admsmith03phdthesis.html`.

Kenny Smith. Cognitive linguistics and connectionist models of language acquisition. Master's thesis, Department of Linguistics, University of Edinburgh, 1999. URL `http://www.isrl.uiuc.edu/~amag/langev/paper/smith99cognitiveLinguistics.html`.

Valentin I. Spitkovsky, Hiyan Alshawi, Angel X. Chang, and Daniel Jurafsky. Unsupervised dependency parsing without gold part-of-speech tags. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing (EMNLP 2011)*, 2011. URL `pubs/goldtags.pdf`.

Guy L. Steele, Jr. *Common LISP: the language (2nd ed.).* Digital Press, Newton, MA, USA, 1990. ISBN 1-55558-041-6.

L. Steels. The origins of syntax in visually grounded robotic agents. *Artificial Intelligence*, 103(1-2):133–156, 1998. doi: 10.1016/S0004-3702(98)00066-6. URL `http://www.isrl.uiuc.edu/~amag/langev/paper/steels98theOrigins2.html`.

L. Steels. The emergence and evolution of linguistic structure: From lexical to grammatical communication systems. *Connection Science*, 17(3-4):213–230, Sep-Dec 2005.

Luc Steels, editor. *Design Patterns in Fluid Construction Grammar.* John Benjamins, Amsterdam, 2011.

Luc Steels and Joachim De Beule. Unify and merge in fluid construction grammar. In P. Vogt and et al., editors, *Symbol Grounding and Beyond: Proceedings of the Third International Workshop on the Emergence and Evolution of Linguistic Communication*, Lecture Notes in Computer Science, pages 197–223. Springer, 2006. doi: 10.1007/11880172_16. URL `http://www.isrl.uiuc.edu/~amag/langev/paper/Steels06Unify_merge_FCG.html`.

Luc Steels and Remi van Trijp. How to make construction grammars fluid and robust. In Luc Steels, editor, *Design Patterns in Fluid Construction Grammar*, pages 301–330. John Benjamins, Amsterdam, 2011.

Thanh Tran, Tobias Mathäß, and Peter Haase. Usability of keyword-driven schema-agnostic search. In Lora Aroyo, Grigoris Antoniou, Eero Hyvönen, Annette ten Teije, Heiner Stuckenschmidt, Liliana Cabral, and Tania Tudorache, editors, *ESWC (2)*, volume 6089 of *Lecture Notes in Computer Science*, pages 349–364. Springer, 2010. ISBN 978-3-642-13488-3. URL `http://dblp.uni-trier.de/db/conf/esws/eswc2010-2.html#TranMH10`.

Max Völkel, Markus Krötzsch, Denny Vrandecic, Heiko Haller, and Rudi Studer. Semantic wikipedia. In *WWW '06: Proceedings of the 15th international conference on World Wide Web*, pages 585–594, New York, NY, USA, 2006. ACM. ISBN 1-59593-323-9. doi: 10.1145/1135777.1135863. URL `http://portal.acm.org/citation.cfm?id=1135863`.

Jimmy Wales. Wikipedia in the free culture revolution. In *Companion to the 20th annual ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications*, OOPSLA '05, pages 5–5, New York, NY, USA, 2005. ACM. ISBN 1-59593-193-7. doi: http://doi.acm.org/10.1145/1094855.1094859. URL `http://doi.acm.org/10.1145/1094855.1094859`.

Chong Wang, Miao Xiong, Qi Zhou, and Yong Yu. PANTO: a portable natural language interface to ontologies. In *The Semantic Web: Research and Applications, 4th European Semantic Web Conference, ESWC 2007, Innsbruck, Austria*, pages 473–487. 2007. doi: 10.1007/978-3-540-72667-8_34.

Colin White and Rolf Schwitter. An update on peng light. In *Proceedings of the Australasian Language Technology Association Workshop 2009*, pages 80–88, Sydney, Australia, December 2009. URL `http://www.aclweb.org/anthology/U09-1011`.