
*University of Coimbra
Faculty of Sciences and Technology
Department of Informatics Engineering*



Probabilistic Reasoning in the Semantic Web using Markov Logic

MSc Thesis

Pedro Carvalho de Oliveira
Candidate

Paulo Jorge de Sousa Gomes
Supervisor

July 2009

*Knowledge and Intelligent Systems Laboratory
Cognitive and Media Systems Group
Centre for Informatics and Systems of the University of Coimbra*



Abstract

The Semantic Web envisions a world where agents share and transfer structured knowledge in an open and semi-automatic way. In most of the cases, this knowledge is characterized by uncertainty. However, Semantic Web languages do not provide any means of dealing with this uncertainty; they are mainly based on crisp logics, unable of dealing with partial and incomplete knowledge. Reasoning in the Semantic Web resigns to a deterministic process of verifying if statements are true or false.

In the last years, some efforts have been made in representing and reasoning with uncertainty in the Semantic Web. These works are mainly focused on how to extend the logics behind Semantic Web languages to the probabilistic/possibilistic/fuzzy logics, or on how to combine these languages with probabilistic formalisms like Bayesian Networks. In all of these approaches, this is achieved by annotating the ontologies with some kind of uncertainty information about its axioms, using this information to perform uncertainty reasoning. Nevertheless, several questions arise: how can reasoning be done efficiently with this uncertainty information? Where to get this uncertainty information?

In this thesis, we present solutions for both questions. The solution for the first question is Markov logic, a new promising approach to reasoning with uncertainty. In this type of logic, there is no right and wrong world; there are multiple worlds with different degrees of probability. This is done by combining logic and probability in the same representation, and then using efficient learning and inference algorithms. For the second question, several solutions were developed:

- If the ontologies are annotated with some kind of uncertainty information, like probabilities, Markov logic can be used to reasoning about this information;
- If the ontology contains individuals, those individuals can be used to automatically learn the uncertainty of the ontology;
- If the ontology does not comprise uncertainty information or individuals, both resources can be automatically learned by analyzing textual resources and web search engines.

We developed a system, *Incerto*, which explores the capabilities of Markov logic for the Semantic Web. This system was applied in several interesting tasks, like reasoning about automatically learned ontologies and social networks analysis.

The main contributions of this thesis are:

- The application of Markov logic for learning and reasoning about uncertainty in the Semantic Web;
- The development of several techniques for learning automatically the uncertainty of ontologies;
- The development of a new technique to parameterize Markov logic networks with probabilities;
- The development of a new technique to learn the probability of ontology axioms by using web search engines;
- The development of *Incerto*, and its application to several Semantic Web domains.

Contents

1.	Introduction	1
2.	State of the Art.....	3
2.1.	Semantic Web.....	3
2.2.	Markov Logic.....	3
2.3.	Related Work	5
2.3.1.	Uncertainty in the Semantic Web.....	5
2.3.2.	Vagueness in the Semantic Web	7
2.3.3.	Conclusions	8
3.	Learning and Reasoning about Uncertainty in the Semantic Web.....	9
3.1.	Probabilistic Reasoning in Uncertainty-annotated Ontologies.....	11
3.1.1.	Probabilities instead of Weights.....	12
3.1.2.	Experimentation	14
3.2.	Probabilistic Reasoning using Ontology Individuals	17
3.2.1.	Experimentation	18
3.3.	Probabilistic Reasoning by Learning Individuals/Probabilities	25
3.3.1.	Learning Individuals	25
3.3.2.	Learning Probabilities	30
3.4.	Incerto – A Probabilistic Reasoner for the Semantic Web.....	35
3.4.1.	Scalability Tests.....	35
3.5.	Final Remarks.....	38
4.	Conclusions	39
4.1.	Future Work.....	40
4.1.1.	General Ideas.....	40
4.1.2.	Probabilistic Reasoning in Uncertainty-annotated Ontologies	41
4.1.3.	Probabilistic Reasoning by Learning Individuals/Probabilities	41
4.1.4.	System.....	41
	References.....	42

1. Introduction

The Semantic Web (Berners-Lee, J. Hendler, and Lassila 2001) envisions a world where agents share and transfer structured knowledge in an open and semi-automatic way. This knowledge, in most of the cases, is characterized by uncertainty, i.e., there is a lack of certainty in assigning a true/false value to a certain knowledge statement. However, the Semantic Web does not provide any means of dealing with knowledge uncertainty. Its languages, like RDF and OWL, are mainly based on crisp logic, being incapacitated of dealing with partial and incomplete knowledge. Reasoning in the Semantic Web resigns to a deterministic process of verifying if statements are true or false.

One field that has been trying to tackle the problem of knowledge uncertainty is the field of probabilistic reasoning (Pearl 1988). This field studies methodologies to represent and reason about uncertain knowledge through the use of probability theory. Some of the developed models, like Bayesian and Markov networks (Roller et al. 2007), are considered the state of the art on dealing with uncertainty. In fact, based on the success of probabilistic reasoning and other similar areas, some efforts have been made on representing and reasoning with uncertainty in the Semantic Web (Thomas Lukasiewicz and Umberto Straccia 2008). These works mainly focused on extending the logics behind Semantic Web languages with probabilistic/possibilistic/fuzzy concepts, or on combining these languages with probabilistic formalisms like Bayesian networks. However, most of these approaches have some problems of applicability in real domains, mainly because its complexity and domain restrictiveness.

Recently, a new area of research, called statistical relational learning (SRL) (Lise Getoor and Ben Taskar 2007), has arisen. SRL tries to expand probabilistic reasoning to complex relational domains, like the Semantic Web. This is achieved by combining representation formalisms, like logic and frame-based systems, with probabilistic models. Some of its more expressive and complete approaches, like Bayesian logic (Milch et al. 2007) and Markov logic (Pedro Domingos, Stanley Kok, et al. 2008), have proven to provide interesting capabilities on learning and reasoning about uncertainty in many real world domains.

The objective of this thesis is to study mechanisms to perform probabilistic reasoning in the Semantic Web. For this purpose, we use Markov logic, a novel representation formalism that combines first-order logic with probabilistic graphical models.

In Markov logic, unlike first-order logic, worlds that violate formulas are not impossible, but only less probable. This is achieved by attaching weights to first-order formulas: higher the weight, bigger the difference between a world that satisfies the formula and one that does not, other things being equal. These weighted formulas represent a Markov logic network, which can be seen as a template to construct Markov networks from given sets of constants: each ground atom is a variable, logical connectives are the edges between variables, and each grounded formula is a feature. The resulting Markov network gives a probability distribution over the possible worlds, being used to answer any probabilistic query about the domain.

Therefore, to apply Markov logic in the Semantic Web, two objects are needed: first-order formulas and their weights. *Formulas* can be acquired by interpreting the semantics of Semantic Web languages as sets of first-order formulas. Since most of these languages represent ontologies based on Description Logics (Baader et al. 2007), they follow a model-theoretic semantics, having a direct correspondence with formulas in first-order logic. *Weights* can be acquired by several ways: if ontologies are annotated with some kind of

uncertainty values, like probabilities, we can interpret these values as weights and perform reasoning; if ontologies contain individuals, these individuals can be used to learn the weights; if we do not have uncertainty annotations or individuals, both can be learned through the analysis of textual resources and web search engines.

To demonstrate the feasibility of our approach, we developed *Incerto*, a system that provides a Semantic Web interface to Markov logic reasoning and learning capabilities. This system can be accessed visually and programmatically, and was used in many interesting Semantic Web tasks, like ontology learning (Maedche 2002) and social network analysis (Mika 2007).

The main contributions of this thesis are:

- The application of Markov logic for learning and reasoning about uncertainty in the Semantic Web;
- The development of several techniques for learning automatically the uncertainty of ontologies;
- The development of a new technique to parameterize Markov logic networks with probabilities;
- The development of a new technique to learn the probability of ontology axioms by using web search engines;
- The development of *Incerto*, and its application to several Semantic Web domains.

The organization of this thesis is as follows:

- *Chapter 2* reviews the main concepts used in this thesis. A description of the related work is also provided;
- *Chapter 3* describes our proposed approach, with several application examples, and presents the developed system;
- *Chapter 4* gives general conclusions and future directions of this work.

2. State of the Art

In this chapter, we present the two main concepts used in this thesis, Semantic Web and Markov logic, followed by a description of the most relevant related work.

2.1. Semantic Web

In the current web, while it is easy to a human infer the meaning of objects in a web page, to a machine this task is not so easy, being only possible to interpret the keywords and links of those objects. The Semantic Web (Berners-Lee, J. Hendler, and Lassila 2001) tries to fill this knowledge gap between human and machines by adding background knowledge to the web, allowing machines to infer the real meaning of objects. This background knowledge is usually expressed by ontologies (James Hendler 2001), i.e., sets of knowledge terms for some particular topic, including the vocabulary, semantic interconnections, and rules of logic/inference of those terms.

The most prominent markup language proposed by the W3C to model ontologies in the Semantic Web is the *Web Ontology Language*¹ (OWL). OWL provides an expressive shared vocabulary to represent knowledge in the Semantic Web. This vocabulary allows expressing axioms about classes, properties, and individuals of the domain. In this paper, we will focus on OWL2² (Grau et al. 2008)(see Appendix II for its functional syntax), the new version of OWL proposed by the W3C, which subsumes the decidable subsets of the original OWL (OWL DL and OWL Lite).

OWL2 is based on the Description logic *SROIQ(D)* (Grau et al. 2008). Description logics (Baader et al. 2007) are a family of logical languages specially designed to model terminological domains. Formulas in Description Logics are composed by two symbols: *concepts* (i.e., sets of individuals) and *roles* (i.e., relationships between individuals). A relevant feature of Description Logics is their separation of knowledge bases in two distinct parts: the intensional knowledge in the form of a terminology, called *Terminological Box* (TBox), and the extensional knowledge, called *Assertional Box* (ABox). The TBox provides the vocabulary, in terms of concepts and rules, of the knowledge base. This is usually done by defining concepts using the logical equivalence constructor (e.g., $Woman \equiv Person \sqcap Female$). The ABox uses the TBox vocabulary to make assertions about individuals (e.g. $Woman(ANNA)$).

2.2. Markov Logic

Markov Logic (Pedro Domingos, Stanley Kok, et al. 2008) combines first-order logic and probabilistic graphical models (Markov networks (Roller et al. 2007)) in the same representation. The main idea behind Markov Logic is that, unlike first-order logic, a world that violates a formula is not invalid, but only less probable. This is done by attaching weights to first-order logic formulas: the higher the weight, the bigger is the difference between a world that satisfies the formula and one that does not, other things been equal. These sets of weighted formulas are called Markov Logic networks (MLNs). Given a set of

¹ <http://www.w3.org/2004/OWL/>

² <http://www.w3.org/TR/owl2-quick-reference/>

constants (i.e., individuals) of the domain and an interpretation, the groundings of the formulas in an MLN can generate a Markov network by adding a variable for each ground atom, an edge if two ground atoms appear in the same formula, and a feature for each grounded formula. The probability distribution of the network is defined as

$$P(X = x) = \frac{1}{Z} \exp\left(\sum_{i=1}^F w_i n_i(x)\right), \quad 2.1$$

where F is the number of formulas in the MLN, $n_i(x)$ is the (binary) number of true groundings of F_i in the world x , w_i is the weight of F_i , and Z is a normalizing constant.

There are two relevant tasks of Markov Logic for this work: weight learning and inference.

Weight Learning

Given an MLN without weights and a set of example data composed by individuals of the domain, weights can be learned generatively by maximizing the *pseudo-log-likelihood* (Besag 1975) of that data. Basically, it is an iterative process where if the model predicts that a formula is true less often than it really is in the data, the weight is increased; otherwise, it is decreased. The pseudo-log-likelihood of world x given weight w is defined as

$$\log P_w^*(X = x) = \sum_i \log P_w(X_i = xi | N_x(X_i)), \quad 2.2$$

where xi is the truth value of variable i , and $N_x(X_i)$ is the truth values of the neighbors of i .

Inference

The most interesting inference task in Markov Logic is to find the marginal and conditional probabilities of a formula given an MLN and possibly other formulas as evidence. Since exact inference can be too difficult in large domains, approximate inference algorithms, like those based on randomized sampling (e.g., Markov Chain Monte Carlo (Roller et al. 2007) (MCMC)), are usually used. However, MCMC is not efficient in domains where formulas with deterministic or near-deterministic dependencies exist (e.g., formulas with infinite weight) because these areas of the search space can be very difficult to traverse by simple flipping the value of the non-evidence variables. To solve this problem, we can use MC-SAT (H. Poon and P. Domingos 2006), a combination of MCMC and the SampleSAT satisfiability solver (Wei, Erenrich, and Selman 2004). MC-SAT uses slice sampling to help capturing the dependencies between variables, allowing jumping from these difficult areas.

Algorithm 1. Given a set of grounded clauses C with their respective weights W , and the maximum number of samples N , the MC-SAT algorithm (H. Poon and P. Domingos 2006) is as follows:

1. $x^{(0)} \leftarrow \text{Satisfy}(\text{hard } C)$
 2. For $i \leftarrow 1$ to N
 - a. $M \leftarrow \emptyset$
 - b. For all $c_k \in C$ satisfied by $x^{(i-1)}$
 - i. With probability $1 - e^{-w_k}$ add c_k to M
 - c. Sample $x^{(i)} \sim \mathcal{U}_{\text{SAT}(M)}$
-

In each iteration of step 2, all the ground clauses satisfied by the current state of the world are added to a subset, M , with probability $1 - e^{-w_k}$. This subset must be satisfied by the next sampled state of the world, which is composed by a uniform sample, \mathcal{U} , from the set of states $SAT(M)$ that satisfy M using the SampleSAT algorithm. The initial state (step 1) is found by applying a satisfiability solver to all the hard clauses in C , i.e., all the clauses with infinite weights. In step 2, these clauses always have probability of 1, and therefore all the sample states need to satisfy them.

2.3. Related Work

This section is dedicated to related work in the field of reasoning in the Semantic Web. There are two main research themes relevant to this topic: *uncertainty in the Semantic Web*, where probabilistic reasoning is applied to the Semantic Web; and *vagueness in the Semantic Web*, where Semantic Web languages are extended with fuzzy logic concepts. At the end of this section, a summary with the most relevant conclusions about the related work is provided.

2.3.1. Uncertainty in the Semantic Web

Some domains are uncertain by nature. Since Semantic Web languages, like OWL and RDF, are based on crisp logic, it is very difficult, if not impossible, to represent those domains in the Semantic Web. Most of the times, this uncertainty comes from our incapacity of asserting the veracity or falsity of a statement. This uncertainty is usually represented by a probability, i.e., a quantity representing our uncertainty.

There are currently two distinct approaches (Thomas Lukasiewicz and Umberto Straccia 2008) to add probabilistic knowledge in the Semantic Web: *Probabilistic Description Logics* tries to expand and modify the logic behind Description Logics with probabilistic knowledge; and *Probabilistic Semantic Web Languages* which tries to combine Semantic Web languages with probabilistic formalisms like Bayesian networks.

Probabilistic Description Logics

The most expressive Description Logic with probabilistic knowledge is $P\text{-}\mathit{SHOIN}(\mathcal{D})$ (T. Lukasiewicz 2008) (Klinov and Bijan Parsia 2008). $P\text{-}\mathit{SHOIN}(\mathcal{D})$ is a probabilistic extension to $\mathit{SHOIN}(\mathcal{D})$, the Description Logic behind OWL DL. Probabilities are represented by a new kind of axiom, called conditional constraints. *Conditional constraints* are composed by expressions of the form $(D|C)[l,u]$, where D is the evidence, C the conclusion, and $[l,u]$ is a probability interval. There are two types of conditional constraints: *generic constraints*, representing probabilistic relations between classes; and *individual constraints*, representing probabilistic information about the belonging of individuals to certain classes. Currently, there are two reasoners that implement this Description Logic: *Pronto*³ (Klinov 2008), a probabilistic OWL reasoner developed by the Pellet⁴ team, and *ContraBovemRufum*⁵ (Nath and Moller 2008), a simple OWL probabilistic reasoner built on top of Racer⁶. Since OWL Lite

³ <http://pellet.owldl.com/pronto>

⁴ <http://pellet.owldl.com/>

⁵ <http://www.sts.tu-harburg.de/~t.naeth/#Software>

⁶ <http://www.racer-systems.com/>

is a subset of OWL DL, its Description Logic, $\mathcal{SHIF}(\mathcal{D})$, can also be extended with probabilistic knowledge, $\mathcal{P}\text{-}\mathcal{SHIF}(\mathcal{D})$ (T. Lukasiewicz 2008).

Very similar to the previous work is $\mathcal{P}\text{-}\mathcal{SHOQ}(\mathcal{D})$ (Giugno and Thomas Lukasiewicz 2002). This Description Logic is based on $\mathcal{SHOQ}(\mathcal{D})$, a Description Logic very similar to $\mathcal{SHOIN}(\mathcal{D})$. The only difference between them is that $\mathcal{SHOQ}(\mathcal{D})$ does not allow inverse properties.

There are also other Description Logics augmented with probabilistic knowledge, but none of those have the needed expressivity to comport the expressiveness of Semantic Web languages. An overview can be found on (Thomas Lukasiewicz and Umberto Straccia 2008).

Probabilistic Semantic Web Languages

Since the emergence of the first Semantic Web languages, some work has been done in representing probabilistic knowledge in those languages. Most of this work tries to add probabilistic capabilities to those languages without changing their logical foundations or their syntax, by combining them with known probabilistic formalisms.

In (Fukushige 2005) is proposed a simple vocabulary to represent probabilistic knowledge in RDF. This vocabulary is composed by a set of classes and properties representing elements of Bayesian networks, making possible to link RDF statements to those elements. This way, marginal and conditional probabilities about statements can be easily represented and reasoned using Bayesian networks. *pRDF* (Udrea, Subrahmanian, and Majkic 2006) is a formal probabilistic extension to a subset of RDF/S, allowing probabilistic knowledge about classes and properties of individuals. Unlike the previous work, *pRDF* implements its own probabilistic logic, making possible to reason over assertional knowledge in acyclic RDF graphs. (Holi and Hyvönen 2006) presented a framework for representing uncertainty in simple RDFS taxonomies. They were particular interested in computing the degrees of *subsumption*, i.e., overlap, between concepts. By attaching weights, called *masses*, to concepts, a Bayesian network is built. Using the Bayesian network prior and conditional probabilities, an overlap table between concepts is easily built by using *evidence propagation* algorithms.

*PR-OWL*⁷ (Costa and Laskey 2005) is a probabilistic generalization of OWL based on *multi-entity Bayesian networks* (MEBNs) (Laskey and Costa 2005). MEBN logic combines Bayesian probability theory with first order logic, constructing Bayesian networks from parameterized fragments representing the probabilistic knowledge about a collection of related hypotheses. This way, probability distributions can be encoded in first-order theories. The main contribute of PR-OWL is the definition of an upper ontology that guides the development of probabilistic ontologies in OWL using MEBNs.

(Ding, Peng, and Rong Pan 2006) proposed *BayesOWL*, a framework to represent and reason about OWL uncertain knowledge using Bayesian networks. They provide a set of rules and procedures to translate OWL DL concept taxonomies (i.e., class axioms and logical relations between classes) into Bayesian networks. The main idea in this translation is to transform all the classes in variables and all the predicates in arcs between the respective classes. Special variables are inserted to facilitate the modeling of relations between concepts, and to avoid cycles. A simple approach to annotate OWL DL statements with conditional and marginal probabilities is also provided, as methods to automatically construct and refine conditional probability tables. The resulting Bayesian network preserves

⁷ <http://www.pr-owl.org/>

the semantics of the original ontology, and supports ontology reasoning both within and across ontologies. This framework was successfully applied in ontology mapping tasks (Rong Pan et al. 2005).

Similar to the previous work is *OntoBayes* (Yang and Calmet 2005), which combines OWL and Bayesian networks. They provide a simple method to annotate OWL with conditional, marginal, and full disjoint probabilities. Unlike the previous work, the Bayesian network is not automatically built. Users must annotate in OWL the dependencies between variables. This way, users are not restricted to a subset of OWL, like *BayesOWL*, but have the burden of annotate more elements. Given the OWL ontology, the probabilities and dependency annotations, a Bayesian network is easily built, and the inference is made on it. (Gu et al. 2004) also propose a similar approach to the last one, being more focused on reasoning over uncertain contexts represented in OWL.

(Henrik and Norbert 2006) describe work on probabilistic reasoning in two subsets of OWL Lite. They translate restricted OWL Lite ontologies into *Datalog*, a subset of first-order logic, and use a probabilistic extension of Datalog, *pDatalog*, to do probabilistic inference over the ontology. This approach was successfully used in automatic ontology matching tasks (Nottelmann and Umberto Straccia 2006). In (Predoiu and Stuckenschmidt 2007), a probabilistic framework for information integration and retrieval on the Semantic Web is proposed. Their approach uses *Bayesian Description Logic Programs*, a formalism that joins *Description Logic Programs* (DLP), a subset of *Datalog* without negation and without equality, with a fragment of *Bayesian Logic Programs*. In this representation, statements are translated to DLP rules with an attached probability. This way, a Bayesian network can be built from those annotated rules, providing a complete specification of the desired probability distribution.

2.3.2. Vagueness in the Semantic Web

Sometimes, real world domains are composed by imprecise or vague information. Again, Semantic Web languages are not ready to deal with this type of information. One way of dealing with vagueness is by using *fuzzy set theory* and *fuzzy logic* (Klir and Yuan 1995). Instead of having a true/false value, statements with vague concepts have a truth value, usually between $[0, 1]$, where 0 and 1 represents binary false and true, respectively. In the last years, many approaches have been proposed to extend Description Logics with fuzzy set theory. Next, the most relevant approaches to this work are described. For a more complete overview see (Sanchez 2006) and (Thomas Lukasiewicz and Umberto Straccia 2008).

(Stoilos et al. 2005a) propose *f-OWL*, a fuzzy extension to OWL DL. In this extension, degrees of truthiness are added to OWL facts, representing the fuzziness of those facts. When a fact does not have any degree, it is interpreted as a binary true fact (i.e., degree of 1). In this approach, OWL syntax must be changed to cope with the addition of the membership degree. The reasoning is made by a new logic, called *f-SHOIN*. This logic extends *SHOIN* (without datatypes) to deal with fuzzy set theory. (Stoilos et al. 2005b) also proposed *f-SHIN*, a simpler fuzzy description logic that is base of *FIRE*⁸, a fuzzy reasoner for the Semantic Web. A more complete fuzzy extension of *SHOIN* was proposed by (U. Straccia 2006a). Although the principles are very similar, this approach subsumes the previous one, being capable of supporting all the OWL DL language.

⁸ <http://www.image.ece.ntua.gr/~nsimou/FIRE/>

Another fuzzy Description Logic reasoner is *fuzzyDL*⁹ (Bobillo and U. Straccia 2008), a fuzzy extension to *SHIF(D)*, the Description Logic behind OWL Lite. This reasoner supports various membership functions and distinct fuzzy logics, like *Zadeh semantics* and *Lukasiewicz logic*. This reasoner also supports classical Description Logic reasoning, being not restricted to fuzzy reasoning. In (U. Straccia 2006b), a fuzzy extension to a restricted subset of OWL DL, called *DL-Lite*, is proposed. This extension, called *f-DL-Lite*, supports fuzzy queries over fuzzy knowledge bases. (J. Z. Pan et al. 2007) extended the previous work, proposing new query answer languages to query fuzzy knowledge bases, by extending SPARQL to support membership degrees. The work is implemented in *ONTOSEARCH2*¹⁰, a search and query engine for the Semantic Web.

2.3.3. Conclusions

Through the analysis of the previous related work, some general conclusions can be made:

- *There is little work on dealing with uncertainty and vagueness in the Semantic Web.* Even if these two areas are of the most importance to the future of the Semantic Web, only recently the scientific community has started to research them (the first publications about the subjects are from 2005). In the same year, a dedicated workshop about those subjects (*International Workshop on Uncertainty Reasoning for the Semantic Web*¹¹) was created;
- *Probabilistic Semantic Web languages are, usually, more computationally efficient than the other uncertain and vague approaches.* The main reason is that most of them use formalisms like probabilistic graphical models, which are well known and provide efficient reasoning mechanisms. Probabilistic and fuzzy Description Logics usually lack of efficient reasoning mechanisms, and their applications to real world domains are also not well studied;
- *All the works on uncertainty and vagueness in the Semantic Web rely on the principle that the uncertainty or vagueness of the ontology is already asserted.* To our knowledge, there is no work on extracting automatically this information from the ontology, or from other knowledge representations. However, there are many ontologies that are uncertain or vague by nature, but do not have any type of information denoting that fact. This fact leads to the need of develop efficient mechanisms to learn this information.
- *All the works on Probabilistic Semantic Web languages rely on probabilistic formalisms, like Bayesian networks, which do not allow cycles.* However, in many domains, knowledge is cyclic (e.g., the relations *FatherOf* and *SonOf* are cyclic). This fact limits the usability and expressiveness of these approaches in real world domains.

These conclusions identify the main problems and debilities of the related work. This information will be used in the definition of our proposed approach (Chapter 3).

⁹ <http://gaia.isti.cnr.it/~straccia/software/fuzzyDL/fuzzyDL.html>

¹⁰ <http://www.ontosearch.org/>

¹¹ <http://c4i.gmu.edu/ursw/>

3. Learning and Reasoning about Uncertainty in the Semantic Web

The objective of this thesis is to study mechanisms to perform probabilistic reasoning in the Semantic Web (Berners-Lee, J. Hendler, and Lassila 2001). For this purpose, we use Markov logic (Pedro Domingos, Stanley Kok, et al. 2008), a novel representation formalism that combines logic and probabilistic graphical models, and ontologies represented in OWL2 (Grau et al. 2008), the Web Ontology language proposed by the W3C. Before explaining our approach, we first need to clarify why we are using Markov logic and OWL2.

Why Markov Logic

As noted in Section 2.1, the most relevant Semantic Web languages to describe ontologies are based on Description Logics (Baader et al. 2007). These Description Logics follow a model-theoretic semantics, and therefore can be usually interpreted as a set of formulas in first-order logic. To perform probabilistic reasoning over these languages, we need formalisms that allow first-order logic and probabilities. These formalisms must be also prepared to cope with domains with a high relational complexity, as the ones presented in the Semantic Web vision (Berners-Lee, J. Hendler, and Lassila 2001). These requirements lead us to the field of statistical relation learning (Lise Getoor and Ben Taskar 2007). In this field, there are two main approaches that combine the full power of first-order logic and probabilistic graphical models: Markov logic and Bayesian logic (BLOG) (Milch et al. 2007). Between those two, Markov logic was chosen for a set of reasons:

- BLOG models are based on Bayesian networks, and therefore do not allow cycles. Since the Semantic Web domain is characterized by cyclic relations between entities (e.g., the relations *FatherOf* and *SonOf* are cyclic), this fact can restrict the use of BLOG in some ontologies. It is studied (Ding, Peng, and Rong Pan 2006) that these cycles can be removed by the introduction of auxiliary variables. However, this is a difficult task that increases the model complexity, and can be only used on a small subset of some Semantic Web languages;
- Even if BLOG allows first-order knowledge, models are procedurally defined by programs, an unnatural way of defining first-order knowledge. To model a simple fact, complex constructors like *guaranteed object statements* and *dependency statements* must be declared, while in Markov logic it suffices to declare the first-order logic formulas of the domain, and their weights;
- Several algorithms for learning and reasoning in Markov logic were studied (e.g., (P. Singla and P. Domingos 2005) (S. Kok and P. Domingos 2005) (P. Singla and P. Domingos 2006a) (H. Poon and P. Domingos 2006)). In BLOG only reasoning was deeply studied (Milch and Russell 2006) (Milch et al. 2008), and there is no work on learning the parameters or the structure of BLOG models.

Why OWL2

The W3C proposed four main Web Ontology language versions: OWL Lite, OWL DL, OWL Full, and OWL2. The last one was chosen for a set of reasons:

- Just like OWL Lite and OWL DL, OWL2 is decidable. This fact is determinant in the search of efficient reasoning mechanisms;
- OWL2 is a very expressive language, which subsumes OWL Lite and OWL DL;
- OWL2 provides improved annotation mechanisms, making the annotation of axioms easier. Annotations are relevant in our domain, since it is the preferred way to attach uncertainty information to axioms;
- Some of the most used Semantic Web tools, like Protégé¹² and OWLAPI¹³, already support OWL2 and are encouraging the Semantic Web community to use this new language.

From OWL to Markov Logic

The first step in use Markov logic capabilities to reason about uncertainty in the Semantic Web is to transform Semantic Web's representation languages, in our case OWL2, into Markov Logic Networks (MLNs). As seen in Section 2.2, a MLN is composed by a set of weighted first-order logic formulas. So, we must define where these formulas and weights come from.

Formulas

OWL2 is based on the Description Logic $SROIQ(\mathcal{D})$ (Grau et al. 2008). One characteristic of Description Logic languages is that they follow a *model-theoretic* semantics (Baader et al. 2007), and therefore can (in most of the cases) be interpreted as formulas in first-order logic. The main idea behind this interpretation is that concepts correspond to unary predicates, roles to binary predicates, and individuals correspond to constants. In our case, $SROIQ(\mathcal{D})$ can be easily interpreted as first order formulas. Table 1 provides some of these interpretations (see Appendix I for the full interpretation).

OWL2 Axiom	First-order logic formula
$SubClassOf(CE_1, CE_2)$	$\forall x : CE_1(x) \Rightarrow CE_2(x)$
$TransitiveProperty(OPE)$	$\forall x, y, z : OPE(x, y) \wedge OPE(y, z) \Rightarrow OPE(x, z)$
$ClassAssertion(CE, a)$	$CE(a)$

Table 1. Examples of first-order logic interpretations of OWL2 axioms.

Weights

In the next sections, we explore several sources for acquiring weights (Figure 1). First, we explore the cases when the ontologies are already annotated with some kind of uncertainty values that can be interpreted as weights. If those values are already weights, the interpretation is straightforward, and no posterior processing is needed (Section 3.1). However, in the cases where those values are probabilities, we must transform them in weights (Section 3.1.1).

Second, we explore the cases where ontologies do not have any type of uncertainty annotation available. If the ontology contains individuals, we can use those individuals to learn the weights using the weight learning capabilities of Markov logic (Section 3.2). In the

¹² <http://protege.stanford.edu/>

¹³ <http://owlapi.sourceforge.net/>

cases where the ontologies do not have individuals, resources like textual corpus and web search engines can be used to learn individuals or derive automatically the probability of axioms (Section 3.3).

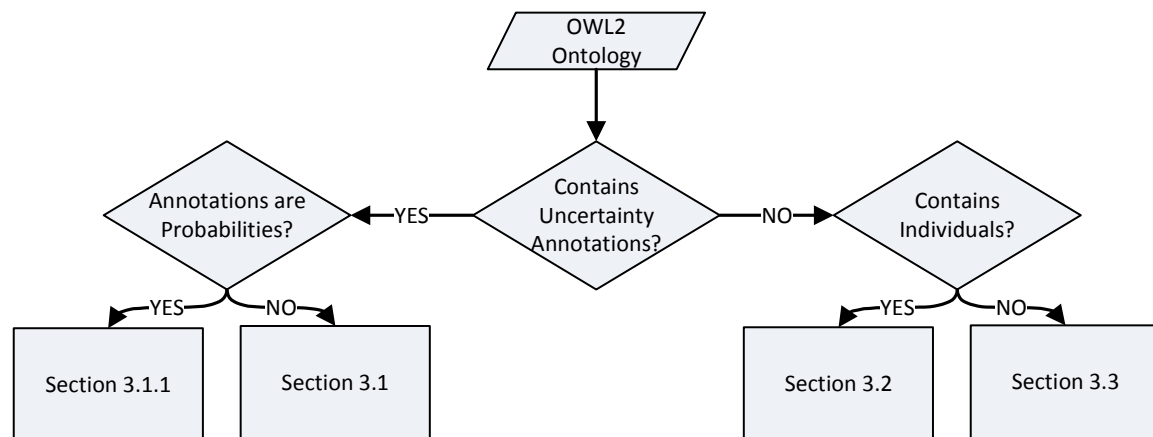


Figure 1. Explored ontology weight sources.

3.1. Probabilistic Reasoning in Uncertainty-annotated Ontologies

Ontology axioms can be annotated with a value representing its uncertainty (usually a weight or a probability) (e.g., a certain class has $X\%$ probability of being subclass of another). This allows ontology engineers to build uncertain ontologies with their own knowledge about the domain. However, this kind of reasoning is not only interesting in this case. There are already domains with ontologies with some kind of uncertainty associated:

- In the field of *ontology learning* from text corpus (Philipp Cimiano 2006), the resulting ontologies usually have a probability associated with the confidence on the asserted relation (e.g., based on the corpus, there is $X\%$ probability that two concepts are related);
- In *ontology mapping, alignment, and matching* tasks (Euzenat and Shvaiko 2007), most of the relations found between distinct ontologies are probabilistic (e.g., two concepts in two different ontologies are the same with $X\%$ probability);
- Ontologies are being used to express *user context* when using a certain application (e.g., (Kersten and Murphy 2006)). For example, an application that recommends actions to the user can use ontologies to assert what are the most interesting concepts to the current context. This is usually done by assigning weights or probabilities to the ontology concepts and relations (e.g., in a word processor, when the user is writing about dogs, the concept *Dog* is $X\%$ more important than the concept *Cat*).

Example

Suppose we have a simple hand-made ontology (Table 2) that models a domain about birds and their capability to fly, composed by 3 classes (*Bird*, *FlyingAnimal*, and *Penguin*) and 2

individuals (*Tim* and *Tweety*). Given the translation to first-order logic in Table 3, using Markov logic we can query for the conditional probabilities of the ontology individuals fly:

$$P(\text{FlyingAnimal}(\text{Tim})) = 0.87$$

$$P(\text{FlyingAnimal}(\text{Tweety})) = 0$$

Axiom	Weight
<i>SubClassOf</i> (<i>Bird</i> , <i>FlyingAnimal</i>)	1.8
<i>SubClassOf</i> (<i>Penguin</i> , <i>ComplementOf</i> (<i>FlyingAnimal</i>))	10
<i>SubClassOf</i> (<i>Penguin</i> , <i>Bird</i>)	10
<i>ClassAssertion</i> (<i>Tim</i> , <i>Bird</i>)	
<i>ClassAssertion</i> (<i>Tweety</i> , <i>Penguin</i>)	

Table 2. Flying Animals ontology.

First-order logic formulas	Weight
$\forall x : \text{Bird}(x) \Rightarrow \text{FlyingAnimal}(x)$	1.8
$\forall x : \text{Penguin}(x) \Rightarrow \neg \text{FlyingAnimal}(x)$	10
$\forall x : \text{Penguin}(x) \Rightarrow \text{Bird}(x)$	10
<i>Bird</i> (<i>Tim</i>)	
<i>Penguin</i> (<i>Tweety</i>)	

Table 3. Flying Animals ontology in first-order logic.

3.1.1. Probabilities instead of Weights

In most of the previously referred cases, the uncertainty is represented as a probability. In Markov logic, weights have a direct correspondence with probabilities if they are interpreted as log odds:

$$w_i = \log \frac{p_i}{1 - p_i}, \quad 3.1$$

Here, p_i is the probability of the formula F_i , and w_i its corresponding weight. However, if F_i shares variables with other formulas, as typically is the case, this correspondence cease to hold, since the weight of F_i is influenced not only by its probability, but also by the other formulas that share the same variable. In this case, the probabilities of all formulas collectively determine all the weights. One solution to this problem is to treat formulas' probabilities as *empirical frequencies* and learn their weights using the algorithms of Section 2.2. For example, if we have the formula $\forall x : \text{Bird}(x) \Rightarrow \text{FlyingAnimal}(x)$ and we know that it is true in 99% of the cases, we create 99 individuals that are both *Birds* and *FlyingAnimals*, and 1 that is a *Bird* but not a *FlyingAnimal*, and then learn its weight with those individuals. However, this solution is unfeasible in large and complex domains for a set of reasons:

- We have to create many individuals, especially if the domain has many types of individuals (we have to have distinct individuals for each one of the types) and/or we want to have a good approximation of the desired probability (e.g., in the previous example, if we had the probability 99.1%, we needed 1000 *Birds*, 991 of them *FlyingAnimals*). The more individuals we have, more difficult is the weight learning;
- We can have very difficult and complex formulas, making the translation to empirical frequencies very difficult to achieve (e.g., $\exists x \forall y : A(x) \wedge (B(x) \wedge \neg C(x)) \vee P(x, y) \Rightarrow x \neq y$);

- It can be impossible to create a proper empirical frequency in cases when formulas contradict other formulas. For example, if we have $\forall x : A(x) \wedge B(x) \Rightarrow C(x)$ and $\forall x : A(x) \Rightarrow \neg C(x)$ both with 100% probability, we cannot create a correct empirical frequency, because an individual of class A cannot be at the same time both C and $\neg C$. These types of formulas could arise in tasks like entity matching, where there could be contradictions between formulas and individuals.

A better solution can be achieved by analyzing one of the weight learning algorithms used in Markov logic. The *discriminative weight learning* algorithm (P. Singla and P. Domingos 2005) maximizes the conditional likelihood of some query predicates taking only in account the evidence atoms X and query atoms Y .

The gradient of the conditional likelihood with respect to the weights is defined by

$$\frac{\partial}{\partial w_i} \log P_w(y|x) = n_i(x, y) - E_w[n_i(x, y)], \quad 3.2$$

where $n_i(x, y)$ is the number of true groundings of clause i in data, and $E_w[n_i(x, y)]$ is the expected number of true groundings of clause i according to the model, value that is approximated by the counts of the most probable state of y given x . The number of true groundings is acquired by counting the number of groundings of the clause that are true in the data in respect to the model. However, if we have the probability of the clause¹⁴, this value can be easily calculated as

$$n_i(x, y) = \text{count}(i) * p_i, \quad 3.3$$

where $\text{count}(i)$ is the total number of groundings of clause i , and p_i the probability of the clause. This way, instead of relying on the individuals to acquire the number of true groundings of the clause, we automatically calculate that value with the desired probability.

This solution is more feasible than the previous one, since it needs fewer individuals (we only need one individual for each individual type) and can be applied in any type of first-order logic formula.

Correctness Study

In this section, we study the behavior of the proposed approach in comparison with the empirical frequency approach. The main objective is to check if the proposed solution derives the same results as the training by empirical frequencies. For this purpose, we created four example MLNs (Table 4), with 10 individuals each, where the formulas were annotated with probabilities. Next, we tested all the possible combinations of probabilities for those formulas, with increases of 0.1, and compared the weights generated by both approaches. As we can see in Table 5, both solutions provide very similar weights, verifying the correctness of the proposed approach.

¹⁴ In Markov logic, formulas are usually transformed in clausal form, generating a set of clauses. In this case, the probability of the formula is equally divided by its clauses.

MLN	First-order logic formulas
1	$\forall x : A(x) \Rightarrow B(x)$
2	$\forall x : A(x) \wedge B(x) \Rightarrow C(x)$
3	$\forall x : A(x) \Rightarrow B(x)$ $\forall x : A(x) \Rightarrow C(x)$
4	$\forall x : A(x) \wedge B(x) \Rightarrow C(x)$ $\forall x : D(x) \Rightarrow \neg C(x)$ $\forall x : B(x) \Rightarrow D(x)$

Table 4. Correctness study MLNs.

MLN	Mean Absolute Weight Difference
1	0.0575
2	0.025
3	0.0625
4	0.05

Table 5. Correctness study results. Mean absolute weight difference is the arithmetic mean of the absolute difference between the weights generated by both solutions with the MLNs of Table 4. A small value indicates that both solutions are similar.

3.1.2. Experimentation

In this section, we present two probabilistic domains that can exploit the previously defined capabilities. First, we analyze a domain about gesture-based affective information recognition, where the probabilities were asserted by field experts. Next, we explore the applicability of Markov logic to reason with automatic learned ontologies.

The Body Gesture Experiment

One of the most interesting tasks in *Affective Computing* (Picard 1997) is to predict the affective state of a person based on its gestures. (Rehman Abbasi, V. Afzulpurkar, and Uno 2008) recorded the unintentional movements of students during a lecture, and manually labeled the movements in 7 gestures (*Head Scratch*, *Nose Itch*, *Lip Touch*, *Eye Rub*, *Chin Rest*, *Lip Zip*, and *Ear Scratch*), corresponding to 6 distinct affective states (*Recalling*, *Satisfied*, *Thinking*, *Tired*, *Bored*, and *Concentrating*). Based on their results, we developed a simple probabilistic ontology (Table 6).

Axiom	Probability
<i>SubClassOf(HeadScratch, Recalling)</i>	1
<i>SubClassOf(ChinRest, Thinking)</i>	0.9
<i>SubClassOf(EyeRub, Tired)</i>	0.81
<i>SubClassOf(LipTouch, Thinking)</i>	0.8875
<i>SubClassOf(NoseItch, Satisfied)</i>	0.775
<i>SubClassOf(LipZip, Bored)</i>	1
<i>SubClassOf(EarScratch, Concentrating)</i>	0.8333

Table 6. Affective state prediction probabilistic ontology.

Next, we defined several sets of individuals and used MC-SAT to perform nine probabilistic queries (Table 7). Some remarks about the results:

- In the first six results, we can see that the probabilities are similar to those expressed in the ontology. The residual differences are derived by the fact that both learning and reasoning are made using approximate algorithms;
- Result number 2 gives a probability 0.07 less than the desired probability of 0.9. This is not only derived by the use of approximate algorithms, but also by the fact that there is other axiom that leads to the conclusion *Thinking* (axiom number 4). Since we do not know if this individual also belongs to the class *LipTouch*, there is a small probability that axiom 4 is also true, and since that axiom contains a smaller weight, the final probability decreases. The inverse occurs in result number 4, where the probability increases in comparison to the expected probability.
- As expected, the probability of result number 8 is greater than the ones with its assertions alone (results 2 and 4).

Set Number	Assertions	Queries and Results
1	<i>ClassAssertion(A, HeadScratch)</i>	<i>Recalling(A) = 0.96</i>
2	<i>ClassAssertion(A, ChinRest)</i>	<i>Thinking(A) = 0.83</i>
3	<i>ClassAssertion(A, EyeRub)</i>	<i>Tired(A) = 0.81</i>
4	<i>ClassAssertion(A, LipTouch)</i>	<i>Thinking(A) = 0.8</i>
5	<i>ClassAssertion(A, LipZip)</i>	<i>Bored(A) = 0.94</i>
6	<i>ClassAssertion(A, EarScratch)</i>	<i>Concentrating(A) = 0.79</i>
7	<i>ClassAssertion(A, EarScratch)</i> <i>ClassAssertion(A, ChinRest)</i>	<i>Thinking(A) = 0.84</i> <i>Concentrating(A) = 0.83</i>
8	<i>ClassAssertion(A, LipTouch)</i> <i>ClassAssertion(A, ChinRest)</i>	<i>Thinking(A) = 0.89</i>

Table 7. Affective state prediction results.

The Ontology Learning Experiment

As previously referred, one of the fields that already produces probabilistic ontologies is the field of ontology learning. In this experiment, we reason about taxonomies automatically learned from web search engines.

Using the *lexico-syntactic* patterns defined by (Hearst 1992) (for more information about those patterns, see Section 3.3.1), we developed a simple system that receives the root of the taxonomy and uses a web search engine to infer its descendants until a pre-defined depth. The subsumption relations receive a confidence value using the following metric (McDowell and Cafarella 2008):

$$Score(i, c) = \frac{count(i, c)}{count(i)}. \quad 3.4$$

Here, $count(i, c)$ is the number of times that class i appears subsumed by c , and $count(i)$ is the total number of times that class i appears subsumed by any class. This metric gives a value in the interval $[0,1]$ and can be roughly interpreted as a probability (i.e., the probability of choosing the class c as the subsumer of i). Using the Yahoo Boss API (see Table 26 for more information about this API), we created a taxonomy with the class *Animal* as root, with a tree-depth of 3 levels. A graphical representation of an excerpt of this

taxonomy, with the respective probabilities, is seen on Figure 2. This taxonomy is easily represented in OWL2 using the *SubClassOf* relation.

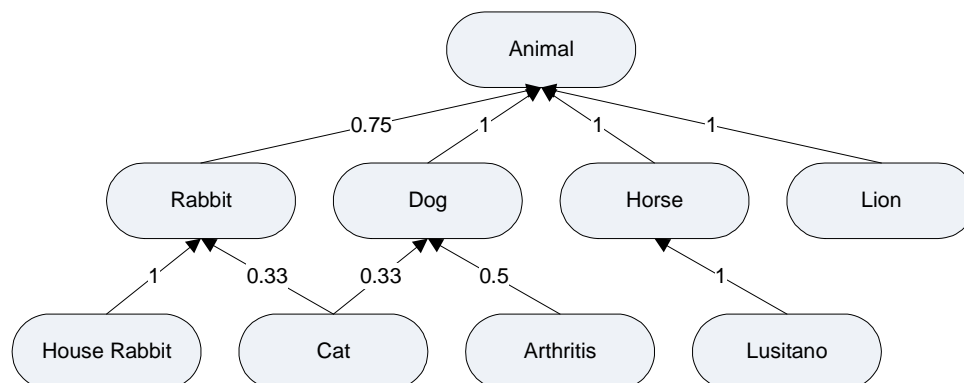


Figure 2. Automatically learned probabilistic taxonomy about animals. Directed edges represent the *SubClassOf* relation (e.g., *SubClassOf(Lusitano,Horse)*).

Using this taxonomy and some sets of example individuals, we can use MC-SAT to query for the conditional probabilities of some classes. The most interesting results are in the next table.

Set Number	Assertions	Queries and Results
1	<i>ClassAssertion(A, Lusitano)</i>	$Horse(A) = 0.9$ $Animal(A) = 0.87$ $HouseRabbit(A) = 0.17$
2	<i>ClassAssertion(A, Cat)</i>	$Dog(A) = 0.24$ $Rabbit(A) = 0.29$ $Animal(A) = 0.76$
3	<i>ClassAssertion(A, HouseRabbit)</i>	$Rabbit(A) = 0.81$ $Animal(A) = 0.58$

Table 8. Most interesting reasoning results of Figure 2 taxonomy.

Using the same approach, we can reason about other automatically extracted taxonomies (Figure 3, Figure 4, and Figure 5). Some example queries and their results can be seen on Table 9.

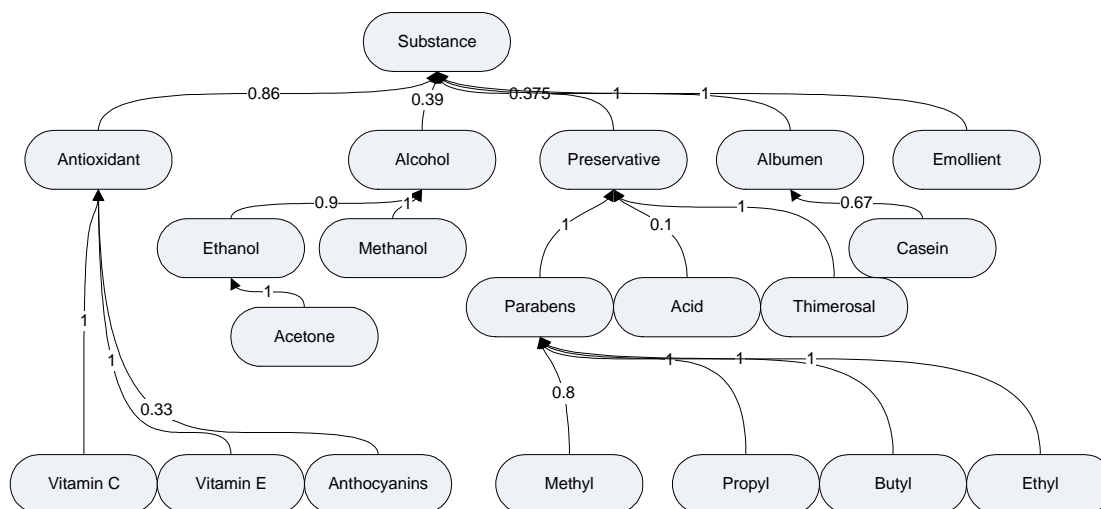


Figure 3. Automatically learned probabilistic taxonomy about substances. Directed edges represent the *SubClassOf* relation.

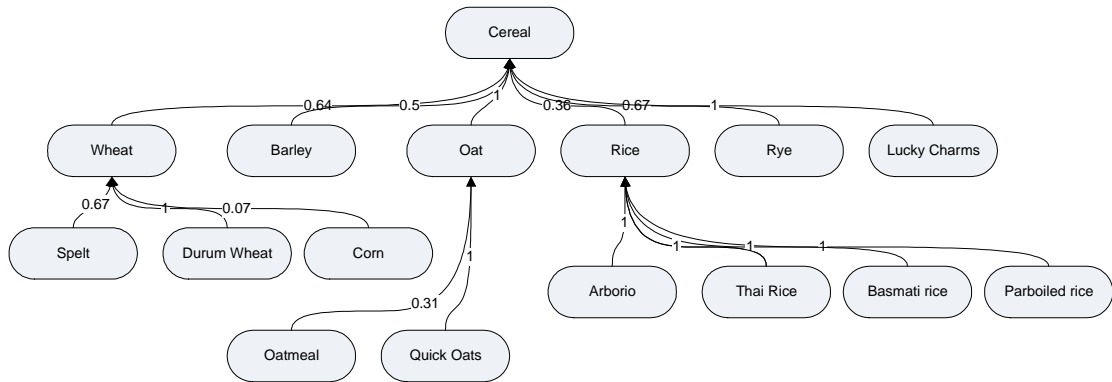


Figure 4. Automatically learned probabilistic taxonomy about cereals. Directed edges represent the *SubClassOf* relation.

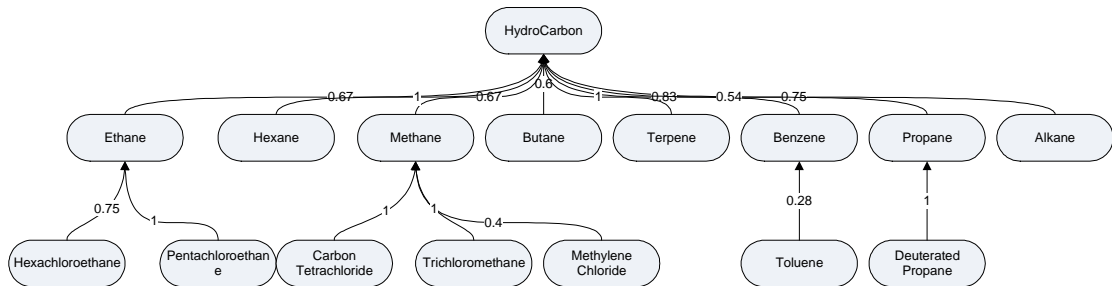


Figure 5. Automatically learned probabilistic taxonomy about hydrocarbons. Directed edges represent the *SubClassOf* relation.

Set Number	Assertions	Queries and Results
1	<i>ClassAssertion(A, Acetone)</i>	<i>Ethanol(A) = 0.91</i> <i>Alcohol(A) = 0.81</i> <i>Substance(A) = 0.58</i>
2	<i>ClassAssertion(A, OatMeal)</i> <i>ClassAssertion(B, BasmatiRice)</i> <i>ClassAssertion(C, Spelt)</i>	<i>Oat(A) = 0.11</i> <i>Cereal(A) = 0.24</i> <i>Rice(B) = 0.95</i> <i>Cereal(B) = 0.33</i> <i>Wheat(C) = 0.72</i> <i>Cereal(C) = 0.1</i>
3	<i>ClassAssertion(A, Pentachloroethane)</i> <i>ClassAssertion(B, Hexane)</i>	<i>Ethane(A) = 0.9</i> <i>HydroCarbon(A) = 0.48</i> <i>HydroCarbon(B) = 0.86</i>

Table 9. Most interesting reasoning results of Figure 2, Figure 3, and Figure 4 taxonomies.

3.2. Probabilistic Reasoning using Ontology Individuals

In the last section, we adopted the principle that ontologies were somehow annotated with some kind of uncertainty information. However, these situations only occur in restricted domains, mainly those where these ontologies were built automatically by machines. In fact, it is studied (Tversky and Kahneman 1974) that humans are not good at either producing or perceiving concepts related to uncertainty, like probabilities. And even if humans were good at perceiving these types of concepts, creating and maintaining large uncertainty annotated

ontologies can be a cumbersome and difficult task, invalidating all the gains that could arise from the annotation. These facts raise the importance of developing mechanisms to learn this uncertainty automatically. This can be useful not only to help users when creating uncertain ontologies, but also to gain access to the vast number of non-uncertainty annotated ontologies already available.

As noted in Section 2.2, in Markov logic, formulas' weights can be learned generatively through example data. This example data usually is composed by individuals of the domain and their relations. In OWL2, individuals correspond to the ABox of the ontology, and therefore they can be used to learn the formulas weights by interpreting them as ground atoms.

Example

Assume a simple ontology about birds (Table 10), with several example birds. Using Markov logic's generative learning, we learned the weights present in Table 11. With that information, using MC-SAT, some probabilistic queries can be made (Table 12).

Axiom
<i>SubClassOf(Bird, FlyingAnimal)</i>
<i>SubClassOf(Penguin, ComplementOf(FlyingAnimal))</i>
<i>SubClassOf(Penguin, Bird)</i>
<i>ClassAssertion(Tim, Bird)</i>
<i>ClassAssertion(Tom, Bird)</i>
<i>ClassAssertion(Tweety, Bird)</i>
<i>ClassAssertion(Tweety, Penguin)</i>
<i>ClassAssertion(Tim, FlyingAnimal)</i>
<i>ClassAssertion(Tom, FlyingAnimal)</i>

Table 10. Flying Animals ontology, with several example birds.

Axiom	Weight
<i>SubClassOf(Bird, FlyingAnimal)</i>	0.88
<i>SubClassOf(Penguin, ComplementOf(FlyingAnimal))</i>	1.45
<i>SubClassOf(Penguin, Bird)</i>	0

Table 11. Learned weights for Table 10 ontology.

Assertion	Query and Result
<i>ClassAssertion(A, Penguin)</i>	<i>FlyingAnimal(A)</i> = 0.17
<i>ClassAssertion(B, Bird)</i>	<i>FlyingAnimal(B)</i> = 0.7

Table 12. Most interesting reasoning results of Table 10 ontology.

3.2.1. Experimentation

In this section, we explore several domains where individuals can be used to learn the uncertainty of the ontology: a financial ontology about a bank and its operations, a web based social network, and two machine learning datasets transformed into ontologies.

The Financial Experiment

Evaluation Procedure and Data Set. Uncertainty reasoning is very important in discovering hidden knowledge in *risk assessment* domains. In this experiment, we use a financial ontology, GoldDLP¹⁵, to assess the risk of certain financial operations. In this ontology, there is information about a bank that offers services like loans and credit cards to private persons. The ontology contains 116 class and property axioms and 297 individuals, mainly distributed between accounts, clients, credit cards, and loans. One of the most interesting tasks in this domain is to determine if a given loan is a problematic loan. There is an OWL class responsible for that information, named *ProblemLoan*, and some axioms about that class (e.g., *ProblemLoan* is the complement of *OkLoan*). The main task in this experiment is to determine each loan's probability of being a *ProblemLoan*.

Experimental Results. Using generative learning and MC-SAT, we found that nine loans have a probability >90% of satisfying the conditions necessary for being a *ProblemLoan*. If we compare the results (Table 13) with a non-probabilistic reasoner, like *Pellet*¹⁶ (Sirin et al. 2007), these are the same nine individuals identified deterministically by it. However, our approach returns some more interesting results that were not identified by *Pellet*. All the other loans have a probability between 45-48% of satisfying the conditions of *ProblemLoan*. This information is valuable because, roughly speaking, it demonstrates that any loan has an associated probability of being a problematic loan. This kind of results cannot be achieved using non-probabilistic reasoning, and therefore demonstrates the necessity of probabilistic reasoning to have a more profound understanding about the domain. However, if we use an existent Semantic Web probabilistic reasoner (e.g., *Pronto*¹⁷ (Klinov 2008)), its results are the same of a non-probabilistic one, since the ontology does not contain any information about the uncertainty of its axioms.

Individual	Probability	Pellet
loan5148	0.48	False
loan5363	0.93	True
loan5549	0.48	False
loan5582	0.45	False
loan5868	0.97	True
loan6007	0.46	False
loan6202	0.98	True
loan6227	0.95	True
loan6297	0.97	True
loan6585	0.45	False
loan6599	0.95	True
loan6995	0.97	True
loan7130	0.97	True
loan7137	0.97	True
loan7154	0.47	False
loan7171	0.47	False

Table 13. Financial experiment results comparison between the proposed approach (*Property* column) and a deterministic reasoner (*Pellet*).

¹⁵ <http://www.cs.put.poznan.pl/alawrynowicz/semintec.htm>

¹⁶ <http://pellet.owldl.com/>

¹⁷ <http://pellet.owldl.com/pronto>

The Social Network Experiment

One of the most used Semantic Web vocabularies is the *Friend of a Friend*¹⁸ (FOAF) vocabulary. This vocabulary allows describing social network data (i.e., persons and their relations) in OWL, with special incentive in linking users from different social networks. There are several web-based social networks that provide information about their users in FOAF (see *Mindswap*¹⁹ for a comprehensive list), and some projects are already exploiting that information (e.g., Google Social Graph API²⁰).

The objective of this experiment is to use Markov logic to explore the relational structure of FOAF networks. As data set, we choose *Advogato*²¹, a social network of free software developers. Advogato provides three interesting FOAF properties for our analysis: *foaf:knows(x,y)*, meaning that user *x* knows user *y*; *foaf:currentProject(x,y)*, meaning that user *x* is currently working in project *y*; and *foaf:member(x,y)*, meaning that user *x* is member of the group *y*. After gathering and processing all the available FOAF profiles, we had a total of 6688 individuals, representing 4198 users, 2487 projects, and 3 groups. Based on the *Link Mining* literature (Lise Getoor and Diehl 2005)(Lise Getoor 2003), we identified three interesting tasks to our experiment: link prediction, link-based classification, and link-based cluster analysis.

Link Prediction

Link prediction (Lise Getoor and Diehl 2005) is the problem of predicting the existence of a link between two objects based on the relations of the object with other objects. In our domain, we are particularly interested in predicting the acquaintance between users, i.e., the *foaf:knows* property. For this purpose, based on our common sense about the domain, we defined three simple rules to perform this task:

Weight	Formula
0.09	$\forall x, y, z : \text{knows}(x, y) \wedge \text{knows}(y, z) \Rightarrow \text{knows}(x, z)$
2.70	$\forall x, y : \text{knows}(x, y) \Leftrightarrow \text{knows}(y, x)$
1.11	$\forall x, y, z : \text{currentProject}(x, z) \wedge \text{currentProject}(y, z) \Rightarrow \text{knows}(x, y)$

Table 14. Link prediction rules.

The first two rules define *knows* as a *transitive* and *symmetric* property, respectively, while the last rule states that if two persons work on the same project, they probably know each other. Weights were learned generatively with all the individuals available. To better describe the results of the link prediction, we developed a simple artificial example composed by 9 users and 3 projects (Figure 6). Next, using MC-SAT, we queried for the conditional probabilities of the *foaf:knows* property for all those users. Results can be seen on Table 15.

¹⁸ <http://www.foaf-project.org/>

¹⁹ <http://trust.mindswap.org>

²⁰ <http://code.google.com/apis/socialgraph/>

²¹ <http://advogato.org/>

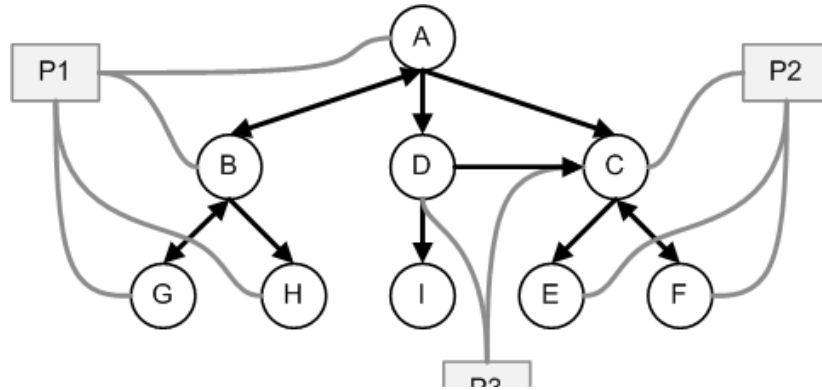


Figure 6. Graphical representation of the artificial example. Users are represented by circles (A-I) and projects by squares (P1-P3). Black directed edges represent the *foaf:knows* relation, while gray undirected edges represent the *foaf:currentProject* relation.

	A	B	C	D	E	F	G	H	I
A	0.83	0.97	0.95	0.93	0.48	0.50	0.91	0.82	0.47
B	1.00	0.97	0.48	0.50	0.44	0.43	1.00	0.98	0.48
C	1.00	0.49	0.93	1.00	0.97	1.00	0.44	0.46	0.47
D	1.00	0.50	0.98	0.84	0.47	0.48	0.50	0.50	0.92
E	0.46	0.44	1.00	0.48	0.86	0.86	0.45	0.43	0.41
F	0.48	0.43	1.00	0.48	0.85	0.86	0.43	0.43	0.41
G	0.90	1.00	0.46	0.49	0.45	0.43	0.84	0.89	0.42
H	0.83	1.00	0.49	0.51	0.43	0.45	0.89	0.83	0.45
I	0.49	0.45	0.46	1.00	0.40	0.42	0.42	0.44	0.59

Table 15. *foaf:knows(x,y)* results of the previous example. Columns represent the x, lines the y (e.g., $P(\text{foaf:knows}(A,G)) = 0.90$).

Some interesting results can be seen in this example:

- $\text{knows}(A,G)$ is greater than $\text{knows}(A,F)$, even if both users are at the same distance from A. The only difference between them is that G works in the same project than A, getting a bigger probability;
- $\text{knows}(D,A)$, $\text{knows}(C,A)$, and $\text{knows}(C,D)$ have big probabilities, mostly because the symmetry of *knows*. However, the probability of $\text{knows}(C,D)$ is the greatest, since both users also work in the same project, P3;
- Since H and F does not share any direct connection, the probability of $\text{knows}(H,F)$ is low, but not null.

Link-based Classification

The main task in *link-based classification* (Lise Getoor 2003) is to predict the category of an object based on the relations of that object with other objects. In our domain, there are three groups of users related to the experience of the user in the community: *Apprentice*, *Journeyer*, and *Master*. These groups are expressed through the *foaf:member* property. The objective of this experiment is to predict each user's group based on their connections to other users. For this purpose, we defined another simple rule (Table 16) that uses the relationship between users expressed on the three rules of Table 14.

Weight	Formula
0.19	$\forall x, y, z : \text{knows}(x, y) \wedge \text{member}(x, z) \Rightarrow \text{member}(y, z)$

Table 16. Link-based classification rule.

This rule states that the group of a user is influenced by the groups of the users that he knows. The weight of the rule was learned generatively in conjunction with the three rules of the previous experiment (their weights remained very similar). Next, we extracted a sub-network composed by 172 users (11 Apprentices, 55 Journeymen, 93 Masters) and 54 projects and randomly removed the group information to 27% of the users (i.e., 47 users). With the rules of Table 14 and Table 16 and the sub-network individuals, we used MC-SAT to predict the membership of the missing group users. The results, using metrics 3.5, 3.6, 3.7, and 3.8, can be seen in the next table.

$$\text{Specificity} = \frac{\text{True Negatives}}{\text{True Negatives} + \text{False Positives}} \quad 3.5$$

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}} \quad 3.6$$

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}} \quad 3.7$$

$$F - \text{Measure} = \frac{2 * (\text{Precision} * \text{Recall})}{\text{Precision} + \text{Recall}} \quad 3.8$$

Group	Specificity	Precision	Recall	F-measure
Apprentice (4)	0.98	0	0	0
Journeyman (15)	0.97	0.83	0.33	0.48
Master (28)	0.37	0.7	1	0.82
Weighted Avg	0.61	0.68	0.70	0.64

Table 17. Link-based classification results. Between brackets is the number of individuals of the group.

Good results can be achieved on predicting user's groups taking only in account the relational structure of the network. The bad results on predicting the *Apprentice* group are probably derived from the small number of elements of that group in the test network. The results could be probably improved if other non-relational information about users was provided (e.g., nationality, age, sex).

Link-based cluster analysis

In the last experiment, we had seen how to classify users in a set of predefined groups. However, in some cases, the information about groups is not available and we still need to segment the users. The goal of *link-based cluster analysis* (Lise Getoor 2003) is to cluster objects into groups that show similar relational characteristics. In our domain, it is interesting to cluster users given their acquaintances with other users. For this task, we can use the three rules presented in the link prediction task (Table 14), since they can give us a relational matrix of the *foaf:knows* property for all the users (i.e., the probability of all the users know each other). Using the same sub-network of the last task (172 users and 54 projects), we used MC-SAT with the previously referred rules to predict the *foaf:knows* property for all the 172 users. With those results, we applied two distinct clustering techniques: the general purpose *k-means* clustering algorithm (Marques de Sá 2001), and

the *Markov Cluster Algorithm*²² (MCA) (Van Dongen 2000), an unsupervised graph clustering algorithm.

After some initial experimentation, we defined the number of desired clusters in the k-means algorithm to 3, and the *inflation* property of the MCA to 1.6 (which also produces 3 clusters). Since the initialization of *cluster centroids* in k-means is random, the algorithm was run 100 times and the best solution is the one presented. Table 18 provides the cluster sizes and the number of shared members between solutions.

		K-means		
		K1 (114)	K2 (47)	K3 (11)
MCA	C1 (135)	102	22	11
	C2 (30)	5	25	0
	C3 (7)	7	0	0

Table 18. Link-based clustering analysis results. The table represents the number of shared members between the clusters of the two algorithms (e.g., cluster C2 and K2 share 25 individuals). Between brackets is the size of each cluster.

Even if the underlying techniques are conceptually distinct, both solutions provide similar clusters, both in size and composition. The biggest clusters from both solutions (*C1* and *K1*) are very similar, as well the second biggest clusters (*C2* and *K2*).

The Non-Relational Experiment

In the last experiments, we have seen how to classify and cluster relational domains. However, there are some domains that are “flat” by nature, i.e., they do not provide any meaningful relation between objects of the same kind. In this experiment, we study two of these domains: the Mushrooms dataset and the Titanic dataset.

Mushrooms Dataset

Based on the Mushrooms Dataset²³, we created a small ontology modelling the domain. The ontology is composed by 2 classes of mushrooms (*Edible* and *Poisonous*) and 6 properties (*hasCapColor*, *hasHabitat*, *hasOdor*, *hasSporePrintColor*, *hasStalkColorAboveRing*, and *hasStalkSurfaceBelowRing*). There are 8124 distinct mushrooms, and the task is to predict the class of the mushrooms given their properties. We split the data in two disjoint sets, each one with 4062 mushrooms, and rotated their usage on the training and test of these rules:

Formula

$$\forall x : !(hasOdor(x, Almond) \vee hasOdor(x, Anis) \vee hasOdor(x, None)) \Rightarrow Poisonous(x)$$

$$\forall x : hasSporePrintColor(x, Green) \Rightarrow Poisonous(x)$$

$$\forall x : hasOdor(x, None) \wedge hasStalkSurfaceBelowRing(x, Scaly) \wedge ! hasStalkColorAboveRing(x, Brown) \Rightarrow Poisonous(x)$$

$$\forall x : hasHabitat(x, Leaves) \wedge hasCapColor(x, White) \Rightarrow Poisonous(x)$$

Table 19. Mushrooms dataset rules as provided in the dataset file. Rules for the *Edible* class are the negation of these four rules. Weights are not demonstrated since they vary with the training set.

²² <http://micans.org/mcl/>

²³ <http://archive.ics.uci.edu/ml/datasets/Mushroom>

Weights were learned generatively, and MC-SAT inference was performed. The (averaged) results can be seen on the next table.

Class	Specificity	Precision	Recall	F-measure
Edible (4208)	0.99	0.98	0.90	0.94
Poisonous (3916)	0.88	0.91	0.98	0.94
Weighted Avg	0.94	0.94	0.94	0.94

Table 20. Mushroom dataset classification results. Between brackets is the number of individuals in each class.

Since the rules were made by experts in the domain, they gave good results in the classification process.

Titanic Dataset

The Titanic Dataset²⁴ is composed by information about the passengers of the RMS Titanic ship that sunked in April 1912. There is information about the class (*First, Second, Third, or Crew*), age (*Adult or Child*), and sex of the passengers. The main task is to predict if a certain passenger survived to the accident given their properties. Using the dataset information, we created a simple ontology modelling the domain, with a total of 2201 passengers. Based on the famous *women-and-children-first* protocol that became famous in the Titanic disaster, we created three simple rules (Table 21).

Formula
$\forall x : hasSex(x, Female) \Rightarrow Survivor(x)$
$\forall x : hasAge(x, Child) \Rightarrow Survivor(x)$
$\forall x : inClass(x, First) \Rightarrow Survivor(x)$

Table 21. Titanic rules. *Non-Survivor* rules are the negation of these three. Weights are not demonstrated since they vary with the training set.

The first two rules state that women and children could have more probability of being survivors, while the last rule states that passengers in first-class also could have more probabilities than the others of being saved. Using generative learning and MC-SAT, the data was split 50:50 in two disjoint sets, being the role of testing and training set swapped and the results averaged:

Class	Specificity	Precision	Recall	F-measure
Survivor (711)	0.81	0.61	0.60	0.61
NonSurvivor (1490)	0.60	0.81	0.81	0.81
Weighted Avg	0.67	0.75	0.75	0.75

Table 22. Titanic dataset classification results. Between brackets is the number of individuals in each class.

This results show that even with simple and easily comprehensible rules it is possible to achieve good classification results with Markov logic.

²⁴ <http://stats.math.uni-augsburg.de/Mondrian/Data/Titanic.txt>

3.3. Probabilistic Reasoning by Learning Individuals/Probabilities

In the last section, we have explored the use of ontology individuals to automatically learn the uncertainty of the ontology axioms and perform inference with that information. This feature proved to be useful in domains where there was no information about that uncertainty, or in complex domains where this uncertainty is hard to infer, specially for humans. However, there are domains that are uncertain but do not have any type of information that could help us infer its uncertainty.

These domains are not uncertainty annotated, and do not have a sufficient number of individuals that allows learning the weights with some confidence in the results. In fact, in most of the cases, these domains do not have any individuals at all. In a preliminary study (Table 23) with 216 ontologies from the TONES ontology repository²⁵, we found that approximately 75% of the ontologies do not have any type of individuals, and only about 7% had more individuals than formulas, fact that indicates a high probability of having a good number of individuals.

Number of ontologies	Number of ontologies with individuals	Number of ontologies with more individuals than formulas
216	54 (25%)	15 (7%)

Table 23. Preliminary study on ontology individuals.

This problem is mainly due to the fact that a large number of Semantic Web ontologies currently available were made to model pure terminological domains, with the main objective of answer questions about concepts and not individuals. In these ontologies, we have to find other ways of gathering information to learn the uncertainty of the axioms. In this thesis, we explored two approaches to tackle this problem: learn individuals and learn probabilities.

3.3.1. Learning Individuals

Due to the enormous quantity of textual resources currently available, specially those present in the World Wide Web and available through web search engines, extracting ontology individuals from those sources has become a task of growing interest. This is the task studied in the field of ontology population.

Ontology Population

The objective of *ontology population* techniques is to, given a source ontology and a corpus, extract individuals of that ontology from the corpus, with their class and property assertions. There are two main types of methods to perform this task: *supervised* and *semi-supervised* methods (e.g., (Tanev and Magnini 2006)), using machine learning classifiers to learn individuals; and *unsupervised methods*, mainly using pre-defined *lexico-syntactic patterns*. In this thesis, we explored unsupervised methods, mainly due to its unsupervised and domain-free applicability.

²⁵ <http://owl.cs.manchester.ac.uk/repository/>

(Hearst 1992) defined 6 simple patterns to extract hyponymy relations from text. These patterns extract hyponyms by analysing expressions like “Animals such as dogs and cats” and “Dogs, cats, and other animals”. As noted by several other works (e.g., (Evans and Street 2004) and (McDowell and Cafarella 2008)), these patterns can be also used to extract class assertions, since an hyponymy relation can be also seen as an assertion that a certain class or individual is a member of (i.e., a hyponym) and is subsumed by other class. Several other works expanded those patterns, by improving them or by proposing new patterns (e.g., (Etzioni et al. 2005) and (P. Cimiano, Ladwig, and Staab 2005)). (McDowell and Cafarella 2008) also propose several metrics to evaluate the ontology population results, and to choose the best class for an individual.

Proposed Approach

After exploring the results of the previously presented works, we implemented the following ontology individual lexico-syntactic extraction patterns:

Type	Pattern
Class Assertion	CLASS {,} such as {NP,*} {(and or)} NP
	such CLASS as {NP,*} {(and or)} NP
	NP {,NP}* {,} (and or) {(all every)} other CLASS
	CLASS {,} (including {e}specially) {NP,*} {(or and)} NP
	CLASS like {NP,*} {(and or)} NP
Property Assertion	NP (is are) {(a an the)} CLASS
	NP (is are) {(a an the)} RELATION NP
	NP {,} RELATION NP

Table 24. Individual extraction patterns. CLASS and RELATION represent the class or relation that is to be populated, and NP represents a Noun Phrase. Optional elements are between braces, disjoint elements between parentheses, separated by a vertical bar. Asterisks indicate that optional elements can appear 0 or more times.

For example, if we want to populate the class “Animal”, we look for expressions in the text like “Animals such as dogs, cats, and rabbits”, “Lions are animals”, “animals, specially chimpanzees and apes”. For properties, for example “son of”, we look for expressions like “Bob is the son of John” and “Bob, the son of John”. *Noun Phrase* detection is performed with the following patterns:

Pattern
NP = {DT} {CD} {AP} NOMINAL
AP = {ADVERB} ADJECTIVE+
NOMINAL = NOUN+

Table 25. Noun phrase detection patterns. DT represents a determiner, CD a cardinal number, and AP an adjective phrase. Optional elements are between braces, and the plus indicates that the element must appear 1 or more times.

The ontology population procedure (Algorithm 2) was implemented in GATE²⁶, a framework for natural language processing that already provides a tokenizer, sentence detector, part-of-speech tagger, and morphological analyzer. Patterns were implemented in JAPE, a finite state transducer engine for annotations based on regular expressions, available in GATE.

²⁶ <http://gate.ac.uk>

Algorithm 2. Given an ontology O and a set of knowledge sources KS , the ontology population process is as follows:

1. Build Corpus
 - a. If KS contains files
 - i. Add files to corpus
 - b. If KS contains search engines
 - i. Issue search engine queries using patterns of Table 24, using the classes and properties of O
 - ii. Add results title and snippet to corpus
2. Process Corpus
 - a. Tokenization
 - b. Sentence Splitting
 - c. Part of Speech Tagging
 - d. Morphological Analysis
3. Parse Results
 - a. Apply Table 24 patterns to the processed corpus
 - b. Match the found assertions with the properties and classes of O
 - c. Add correct assertions to O

As knowledge sources, we can use electronic documents (e.g., *Microsoft Word* and *Adobe PDF* documents) or web search engines. We implemented the access to the three most used web search engines (*Google*, *Yahoo*, and *Live Search*). However, each of these web search engines provides distinct limitations (Table 26), which can influence the quality of the extracted individuals. Due to the maximum number of queries restriction of Live Search, in this work we mainly use Google and Yahoo search engines capabilities.

Web Search Engine	Max. Number of Queries	N. Results per query	Registration
Google Search API ²⁷	Unlimited*	64 (8 per time)	Optional
Yahoo BOSS ²⁸	Unlimited*	100	Mandatory
Live Search API 2.0 ²⁹	7/second	1000 (50 per time)	Mandatory

Table 26. Web search engines APIs and their features. The unlimited number of queries of Google and Yahoo is theoretical.

Example

Suppose we want to populate a simple ontology about the characters of the animation television series *Dragon Ball*³⁰ and their fights (Figure 7). Using the extraction rules previously defined, and using the Google Search API as the knowledge source with 8 results per query, we found the assertions of Table 27.

²⁷ <http://code.google.com/apis/ajaxsearch/>

²⁸ <http://developer.yahoo.com/search/boss/>

²⁹ <http://dev.live.com/livesearch/>

³⁰ http://en.wikipedia.org/wiki/Dragon_Ball

From the 11 Dragon Ball characters found, only one, “character” is wrong. This is due to the phrase “its *characters are dragon ball characters*”. Some of them, like *Kid Chi* and *Tien* are abbreviations of the full names (*Kid Chi-Chi* and *Tien Shinhan*, respectively).

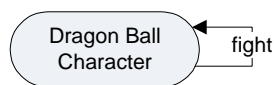


Figure 7. Dragon Ball ontology. Directed edge represents a property, rounded square a class.

Class/Property	Assertion	Count
Dragon Ball Character	goku	4
	piccolo	3
	frieza	2
	emperor pilaf	1
	character	1
	kid chi	1
	majin buu	1
	tien	1
	vegeta	1
	king piccolo	1
	chiaotzu	1
fight	(goku,piccolo)	1
	(piccolo,frieza)	1
	(goku,tien)	1
	(vegeta,frieza)	1

Table 27. Dragon Ball ontology population individuals.

Experimentation

In this experiment, we automatically learn an ontology (Figure 8) about diseases and their symptoms and perform clustering on it.

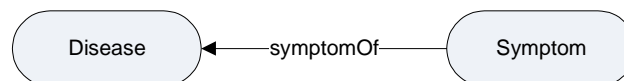


Figure 8. Diseases ontology. Directed edge represents a property, rounded squares classes.

The first step is to learn the individuals of the class *Disease* using the class assertion patterns of Table 24. As corpus, we used both Google Search and Yahoo BOSS APIs with the maximum results possible per query (64 for Google, 100 for Yahoo). As we can see in Table 28, both search engines give similar precision results. Yahoo BOSS gives a bigger total number of individuals due to the bigger number of results per query. Precision results could be improved if we ignored the diseases that appeared only once in the corpus. However, their number is largely reduced (to about one third in both search engines). After joining all the correct diseases found by both search engines, and removing a small number of non-human diseases like powdery mildew, a total of 271 distinct diseases were found.

Web Search Engine	Individuals	Total	Correct	Precision
Yahoo BOSS	All	330	231	0.70
	More than 1 count	117	92	0.79
Google Search API	All	193	128	0.66
	More than 1 count	67	51	0.76

Table 28. Disease ontology class population results.

The next step is to learn the symptoms of those diseases. Using the property assertions patterns of Table 24, we queried both search engines with two textual variations of the *symptomOf* property: “symptom of” and “sign of”. Since we already know which diseases we want to find symptoms, the last part of the patterns can be easily substituted by the desired disease, i.e., we can query for “* is a symptom of Malaria” and “* is a sign of Malaria” instead of the more general queries “* is a symptom of *” and “* is a signal of *”. The results (Table 29) indicate similar precision values for both search engines.

After joining all the correct symptom assertions produced by both search engines, and removing the diseases without symptoms, we had an ontology composed by 140 diseases, 459 symptoms, and 671 symptoms assertions. One interesting fact is that there are individuals that are represented as both diseases and symptoms. For example, *Bronchitis* is defined as a disease, but also as a symptom for other diseases, like *Lung Cancer*. This fact occurs with 21 of the 140 diseases.

Web Search Engine	Individuals	Total	Correct	Precision
Yahoo BOSS	All	618	389	0.63
	More than 1 count	228	144	0.63
Google Search API	All	661	419	0.63
	More than 1 count	187	133	0.71

Table 29. *SymptomOf* ontology property population results.

An interesting task to perform with this kind of ontology is to create clusters of similar diseases. For this purpose, we can use a set of similar rules to those used in the clustering experimentation of Section 3.2.1:

Weight	Formula
0.21	$\forall x, y, z : \text{symptomOf}(z, x) \wedge \text{symptomOf}(z, y) \Rightarrow \text{Similar}(x, y)$
7.27	$\forall x, y : \text{Similar}(x, y) \Leftrightarrow \text{Similar}(y, x)$
0	$\forall x, y, z : \text{Similar}(x, y) \wedge \text{Similar}(y, z) \Rightarrow \text{Similar}(x, z)$

Table 30. Diseases clustering rules.

In this case, the first rule defines that two diseases are similar if they share symptoms. The other two rules define *Similar* as a symmetric and transitive property, respectively. Weights were learned generatively using all the individuals available. We used MC-SAT with the previously referred rules to predict the *Similar* property for all the 140 diseases. With those results, we applied two partitional clustering techniques (Zhao and Karypis 2005) available in the *CLUTO*³¹ toolkit: *Repetead bisections* (RB) and *Nearest-Neighbor Graphs* (NNG), both configured to create 5 clusters each:

³¹ <http://glaros.dtc.umn.edu/gkhome/cluto/cluto/overview>

		RB				
		K1 (84)	K2 (15)	K3 (24)	K4 (8)	K5 (9)
NNG	C1 (42)	42	0	0	0	0
	C2 (20)	7	13	0	0	0
	C3 (35)	3	0	23	0	9
	C4 (29)	20	0	1	8	0
	C5 (14)	12	2	0	0	0

Table 31. Link-based clustering analysis results. The table represents the number of shared members between the clusters of the two algorithms (e.g., cluster C2 and K2 share 13 individuals). Between brackets is the size of each cluster.

Both solutions created similar clusters, specially (C2, K2) and (C3, K3). Some of these clusters can be analyzed by their main symptoms. For example, all the diseases in cluster K5 have a common symptom: depression. This cluster includes diseases like migraines, anxiety, alzheimers, and bipolar disorder. Cluster K4 is composed by diseases related to the respiratory system, like lung cancer, tuberculosis, asthma, pneumonia, and diphtheria. These diseases share symptoms like coughing, chest pain, and bronchodilation. In Cluster C5, most of the diseases, like cholera, salmonella, and colon cancer, share diarrhea as a common symptom.

3.3.2. Learning Probabilities

An OWL2 ontology is composed by a set of axioms that model the semantic relations between entities of the domain. In the last two sections, we have seen how to use individuals to learn automatically the uncertainty of those axioms. However, these approaches can have some problems:

- We need a representative number of individuals to perform the learning process with some confidence in the results. This arise problems when: we do not have the desired number of individuals; or the final number of individuals is too large to perform weight learning efficiently.
- Weights are hard for humans to understand, not only because they are not normalized, but also because in Markov logic they do not have a direct correspondence to probabilities, and therefore their values can be misleading.

An interesting approach would be to automatically learn the probability of axioms, instead of weights, and without the need for learning individuals. In this thesis, we explore the use of semantic similarity techniques to perform this task.

Semantic Similarity

Semantic similarity (Bollegala, Matsuo, and Ishizuka 2007) is the process of finding the similarity between two words or entities. This is usually done by studying the *co-occurrence* between those words or entities in a textual corpus: if they appear together many times, this could be the indication that they are somehow semantically related. The most used techniques use the redundancy and size of a huge corpus, like the World Wide Web, and the results of search engines to measure that similarity. This is usually done by counting the number of results returned by those search engines in specific queries related to the words or entities whose similarity we want to assert.

In this thesis, we explored 6 distinct metrics to perform this task (Bollegala, Matsuo, and Ishizuka 2007) (Church and Hanks 1990) (Magnini et al. 2002) (Cilibrasi and Vitanyi 2004):

$$\text{WebJaccard}(P, Q) = \frac{H(P, Q)}{H(P) + H(Q) - H(P, Q)} \quad 3.9$$

$$\text{WebOverlap}(P, Q) = \frac{H(P, Q)}{\min(H(P), H(Q))} \quad 3.10$$

$$\text{WebDice}(P, Q) = \frac{2H(P, Q)}{H(P) + H(Q)} \quad 3.11$$

$$\text{PointwiseMutualInformation}(P, Q) = \log_2 \left(\frac{H(P, Q)}{H(P)H(Q)} * N \right) \quad 3.12$$

$$\text{CorrectedConditionalProbability}(P, Q) = \frac{H(P, Q)}{H(P)H(Q)^{\frac{2}{3}}} * N^{\frac{2}{3}} \quad 3.13$$

$$\text{NormalizedWebDistance}(P, Q) = \frac{\max(\log H(P), \log H(Q)) - \log H(P, Q)}{\log N - \min(\log H(P), \log H(Q))} \quad 3.14$$

Here

- $H(P)$ is the number of search engine results for search phrase P
- $H(Q)$ is the number of search engine results for search phrase Q
- $H(P, Q)$ is the number of search engine results for the tuple of search phrases PQ
- N is the total number of indexed pages by the search engine

When search phrases P or Q are composed by more than one word, they are enclosed by quotation marks. The index size of search engines, N , is usually not known, but it can be approximated by using sampling techniques (e.g. (Bar-Yossef and Gurevich 2008)). Based on the updated results provided by (de Kunder 2009), in this work we estimate the size of the Google index to 40 billion pages, and Yahoo to 20 billion.

WebJaccard, WebOverlap, and WebDice give a value in the interval $[0,1]$, with higher values representing bigger similarities. Pointwise Mutual Information (PMI) and Corrected Conditional Probability (CCP) give a real positive number, with higher values also representing bigger similarities. The Normalized Web Distance (NWD) gives a positive real value, with the values closer to 0 representing the most similarity. However, some of these properties can change in some rare conditions (e.g., when search engines return more results in $H(P, Q)$ than in $H(P)$ and/or $H(Q)$). In these cases, if the result diverges from its range, we can clip the value to the closer value of the range (e.g., if PMI gives a result below 0, we can simply set it to 0).

Example

To assert the semantic similarity between the words “dog” and “pet” using the Google Search API, we perform the following queries:

Query	Result Count
dog	64,900,000
pet	61,200,000
dog pet	53,500,000

Table 32. Google Search API query results count.

The metrics results can be seen on Table 33, accompanied by other examples.

Similarity	WebJaccard	WebOverlap	WebDice	PMI	CCP	NWD
(dog,pet)	0.74	0.87	0.85	9.07	62.08	0.03
(coimbra,portugal)	0.01	0.45	0.02	6.95	18.97	0.48
(google,Pedro)	0.00	0.20	0.01	1.69	0.31	0.83
(good,evil)	0.12	0.83	0.21	6.94	12.27	0.30
(banana, einstein)	0.00	0.01	0.01	4.80	1.65	0.61

Table 33. Examples of semantic similarity metrics results using the Google Search API.

Proposed Approach

The presented techniques assert the semantic similarity between any two entities by giving a confidence value to the likelihood that these entities have any type of semantic relation. However, in our specific case, we do not want to calculate if two entities are connected by any semantic relation: we already know they probably are, we just want to give a confidence value to that relation. For example, if we have the axiom $SubClassOf(Dog, Pet)$, we are not interested in asserting if the entity Dog has any type of semantic relation with Pet : we already know that they probably have one (a subsumption relation). What we want is a confidence value to that specific relation between those two concrete entities. For this purpose, we can use the previously referred metrics with some modifications, namely in the format of the search engine queries.

If we have a semantic relation defined by (P, R, Q) , where P is the subject, Q the object, and R the relation, we redefine the previous query definitions as:

- $H(P)$ is the number of search engine results for search phrase " $PR *$ "
- $H(Q)$ is the number of search engine results for the search phrase " $* RQ$ "
- $H(P, Q)$ is the number of search engine results for the triple of search phrases " PRQ "

Here, $*$ represents a web search engine wildcard that matches any potential word. Note that query phrases are obligatorily enclosed with quotation marks.

Example

If we have the axiom $SubClassOf(Dog, Pet)$, which can be translated to the semantic relation $(Dog, is\ a, Pet)$, we perform the following queries using Google Search API:

Query	Result Count
"dog is a *"	287,000
"* is a pet"	182,000
"dog is a pet"	785

Table 34. Google Search API query results count.

The metrics results, with some more examples, can be seen on the next table.

Relation	WebJaccard	WebOverlap	WebDice	PMI	CCP	NWD
(dog,is a,pet)	0.002	0.004	0.003	9.230	9.949	0.480
(Obama,president of,United States)	0.031	0.135	0.059	16.745	1,175.085	0.220
(Einstein, eat,banana)	0.000	0.000	0.000	0.000	0.000	$+\infty$
(Coimbra,is in,Portugal)	0.000	0.003	0.000	12.432	44.438	0.536
(good,is,evil)	0.000	0.006	0.000	2.891	0.180	0.820

Table 35. Examples of the modified semantic similarity metrics results using the Google Search API.

As we can see, the first 3 metrics give results always near 0. This is mainly due to the fact that $H(P, Q)$ is constantly small compared to $H(P)$ and $H(Q)$, and since in these metrics $H(P, Q)$ is divided by some form of combination between $H(P)$ and $H(Q)$, the resulting value is small. The other metrics take the size of the index, N , into account, and, based on our common sense, give better results.

Given those results, we have metrics that give values in the range $[0,1]$, but in this case they always give values near 0. And we have metrics that work as intended, but give a positive real value. However, we need that those metrics gave us a well distributed value between $[0,1]$ that could be interpreted as our confidence on the veracity of the relation. To this purpose, we need to normalize those values. If we have a set of values $[v_0, \dots, v_n]$, a value v can be normalized into range $[x, y]$ by performing a linear scaling transformation:

$$v' = \frac{v - \min}{\max - \min} * ((y - x) + x). \quad 3.15$$

Since we know that $x = 0$, $y = 1$, and $\min = 0$ in all of the metrics, this formula can be simplified into a simple division by \max :

$$v' = \frac{v}{\max} \quad 3.16$$

In the case of NWD, this value must be also converted into a dissimilarity:

$$v'' = 1 - v' \quad 3.17$$

If we apply the normalization to the previous results, we get the following values:

Relation	WebJaccard	WebOverlap	WebDice	PMI	CCP	NWD
(dog,is a,pet)	0.055	0.032	0.056	0.551	0.008	0.995
(Obama,president of,United States)	1.000	1.000	1.000	1.000	1.000	0.998
(Einstein,eat,banana)	0.000	0.000	0.000	0.000	0.000	0.000
(Coimbra,is in,Portugal)	0.002	0.021	0.002	0.742	0.038	0.995
(good,is,evil)	0.003	0.042	0.004	0.173	0.000	0.992

Table 36. Normalization of the metrics values in Table 35.

In some cases, a semantic relation can have more than one query pattern. For example, the *subClassOf* relation can have all the patterns defined in Table 24, since these patterns were originally designed to infer *subClassOf* relations. In those cases, for one semantic relation, we have several values for each metric. The final metric value can be simply an arithmetic mean of those values, e.g., if $PMI(R) = \{v_1, \dots, v_n\}$, the new PMI is:

$$PMI'(R) = \frac{\sum_i v_i}{\text{count}(\text{WebPMI}(R))}, \quad 3.18$$

where $count(PMI(R))$ is the number of patterns used for semantic relation R .

Experimentation

In Section 3.1.2, we have extracted a taxonomy about animals from a web search engine, and used a simple metric based on the frequency of the extracted entities to assert the confidence value of the relations. In this experiment, we use the presented metrics to assert those confidence values. Using the patterns of Table 24, we issued the queries of Table 37 and used the Google Search API to calculate the various metric values. The results can be seen in Table 38.

Pattern	$H(P)$	$H(Q)$	$H(P, Q)$
1	"* such as P"	"Q such as *"	"Q such as P"
2	"such * as P"	"such Q as *"	"such Q as P"
3	"P (and OR or) (all OR every)? other *"	"* (and OR or) (all OR every)? other Q"	"P (and OR or) (all OR every)? other Q"
4	"* (including OR specially OR especially) P"	"Q (including OR specially OR especially) *"	"Q (including OR specially OR especially) P"
5	"* like P"	"Q like *"	"Q like P"
6	"P (is OR are) (a OR an OR the)? *"	"* (is OR are) (a OR an OR the)? Q"	"P (is OR are) (a OR an OR the)? Q"

Table 37. Search engine queries based on patterns from Table 24. ? indicates an optional pattern, indicating that two queries will be issued: one with that pattern, and one without.

Relation	Desired	WebJaccard	WebOverlap	WebDice	PMI	CCP	NWD
rabbits, animals	1	0.27	0.81	0.28	0.96	0.53	0.99
dogs, animals	1	1.00	1.00	1.00	0.98	0.82	1.00
horses, animals	1	0.63	0.96	0.65	1.00	0.77	1.00
lions, animals	1	0.16	0.60	0.17	0.94	0.55	0.99
house rabbits, rabbits	1	0.00	0.18	0.00	0.28	1.00	0.00
cats, rabbits	0	0.01	0.21	0.01	0.48	0.02	0.66
cats, dogs	0	0.11	0.09	0.11	0.62	0.09	0.83
arthritis, dogs	0	0.00	0.00	0.00	0.19	0.00	0.33
lusitano, horses	1	0.00	0.08	0.00	0.26	0.33	0.00
Correlation	1	0.43	0.62	0.44	0.45	0.85	0.07

Table 38. Modified semantic similarity metrics results based on query results from Table 37.

CCP is the metric with best correlation. This result is mainly derived from the fact that it gives very low values to the wrong assertions (i.e., the ones with 0 as its desired value). The analysis of the results of the appliance of the individuals patterns (Table 39) demonstrate that there is no single pattern that is responsible for the high correlation of CCP: some patterns are good to measure specific assertions (e.g. pattern 4 and 6 to assertion (*House Rabbits, Rabbits*)), while others give average results in all the patterns (e.g. pattern 1). The combination of all the patterns is the key to the good results.

The other metrics also give good results, especially WebOverlap. NWD contains a low correlation since it can only approximate the correct value of the four first assertions.

Relation	Desired	1	2	3	4	5	6
rabbits, animals	1	0.54	0.58	0.26	0.08	1.00	0.39
dogs, animals	1	0.48	0.51	1.00	0.12	0.61	0.33
horses, animals	1	0.46	0.26	0.97	0.14	0.62	1.00
lions, animals	1	0.26	1.00	0.32	0.09	0.81	0.17
house rabbits, rabbits	1	0.00	0.00	0.00	1.00	0.00	0.93
cats, rabbits	0	0.00	0.00	0.05	0.00	0.06	0.05
cats, dogs	0	0.00	0.00	0.21	0.00	0.07	0.11
arthritis, dogs	0	0.00	0.00	0.00	0.00	0.00	0.00
lusitano, horses	1	1.00	0.00	0.00	0.00	0.00	0.84
Correlation	1	0.65	0.54	0.42	0.37	0.58	0.71

Table 39. Results of the modified CCP metric, with separate patterns from Table 37.

3.4. Incerto – A Probabilistic Reasoner for the Semantic Web

Using the ideas of the previous sections, we developed *Incerto*, a probabilistic reasoner for the Semantic Web based on Markov logic. The system was developed in Java, and is freely available through a GNU Lesser General Public License³² (LGPL) at <http://code.google.com/p/incerto>.

The system interacts with four main external components: the *Ontology Processor (OWL API³³)*, responsible for the reading, processing, and writing of OWL2 ontologies; the *Markov Logic Engine (Alchemy³⁴ (S. Kok et al. 2007))*, responsible for the Markov logic reasoning and learning processes; the *Natural Language Processor (GATE³⁵)*, responsible for the processing of natural language resources; and a set of *Additional Libraries*, used in several minor tasks. There are three ways of communicating with the system: programmatically through an API; visually with a GUI; and with a command line interface.

3.4.1. Scalability Tests

In this section, we study the scalability of Markov logic procedures in the Semantic Web domain, as implemented in *Incerto*. Our purpose is to measure the scalability of three distinct procedures:

- *Pre-processing*, composed by the load and transformation of ontologies in MLNs;
- *Weight Learning*, using the generative algorithm (see Section 2.2);
- *Inference*, using the MC-SAT algorithm (see Section 2.2).

For this purpose, those procedures were performed on seven distinct ontologies, each one with a varied number of individuals. This variation was made by randomly populating each ontology with a set of individuals (we tested with 1, 10, 100, 1.000, and 10.000 individuals), using these individuals to make an average of three assertions for each ontology class or property. The following ontologies were used:

³² <http://www.gnu.org/copyleft/lesser.html>

³³ <http://owlapi.sourceforge.net/>

³⁴ <http://alchemy.cs.washington.edu/>

³⁵ <http://gate.ac.uk/>

-
- **Animals** – Automatically learned ontology (Figure 2), composed by 9 classes and a simple taxonomic structure;
 - **Substances** – Automatically learned ontology (Figure 3), composed by 20 classes and a more complex taxonomic structure;
 - **Body Gestures** – Simple ontology about body gestures (Table 6), with 13 classes;
 - **Diseases** – Ontology about diseases and their symptoms (Figure 8). The ontology was augmented with the rules of Table 30, being composed by three classes and one property.
 - **Social Network** – Social network of Section 3.2.1, with three classes and three properties. Augmented with the rules of Table 14 and Table 16.
 - **GoldDLP** – Financial ontology used in Section 3.2.1, with 39 classes and 6 properties.
 - **Wine** – Ontology about wines³⁶, composed by 43 classes and 13 properties.

The averaged results (5 runs for each experiment), using an *Intel Centrino Duo T2300* with 1536 MB of memory and the *Alchemy* engine, can be seen on Figure 9, Figure 10, and Figure 11. Some interesting results were found:

- Pre-processing takes a relatively small time comparing to the weight learning and inference procedures, especially when the number of individuals is high. The *Wine* ontology took more time in this procedure, mainly because it contains more complex axioms, making its interpretation to first-order logic more difficult;
- *Animals*, *Substances*, and *Body Gestures* ontologies take less than six seconds to perform weight learning with 10.000 individuals. However, *Diseases*, *Social Network*, and *GoldDLP* ontologies took around half an hour with 1.000 individuals, and exceeded the predefined maximum time (three hours) with 10.000 individuals. The *Wine* ontology only processed 10 individuals in the maximum time;
- Inference is the most exigent procedure with all the ontologies. In the first three ontologies, inference was made with the maximum individuals in the predefined time. With the *Diseases*, *Social Network*, and *GoldDLP* ontologies, inference was only possible with 100 individuals. With more individuals, the available memory was exhausted. The *Wine* ontology only processed 10 individuals in the maximum time.

The bad inference results in the last four ontologies are mainly due to the fact that the *Wine* ontology contains *cardinality restrictions*, while the other three have *transitivity* properties. Both cardinality restrictions and transitivity generate first-order formulas with more than two free variables (see Appendix I). Since in Markov logic all the possible combinations between these free variables with the individuals of the domain must be grounded, the number of groundings increases exponentially with the number of free variables. This fact raises the complexity of reasoning, both in terms of time and used memory.

In the weight learning, the bad results are also derived from the same problem, because even if inference is not made (only discriminative learning uses inference), the number of groundings for each formula must be counted, and the more complex are the formulas, more intensive is the counting.

³⁶ <http://www.w3.org/TR/2003/CR-owl-guide-20030818/wine>

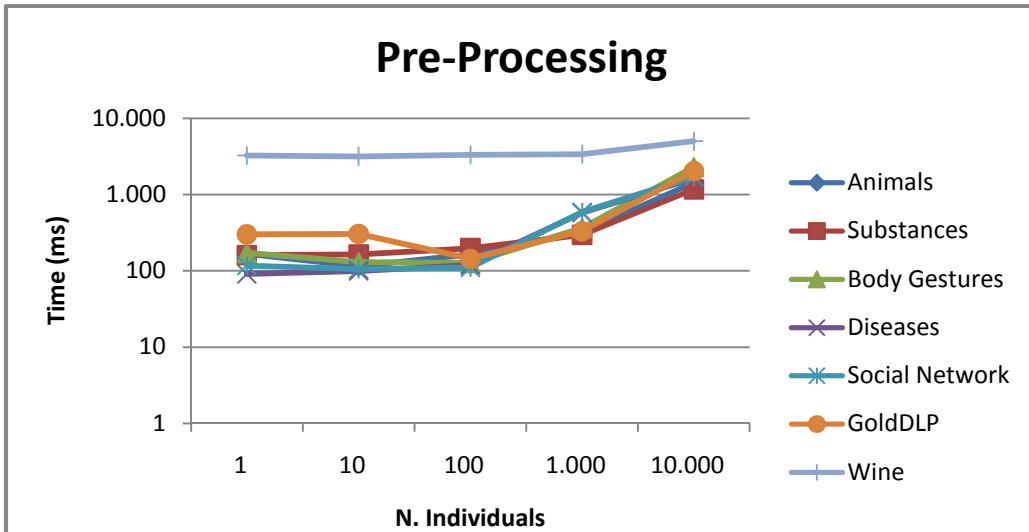


Figure 9. Pre-Processing scalability results (logarithmic scale).

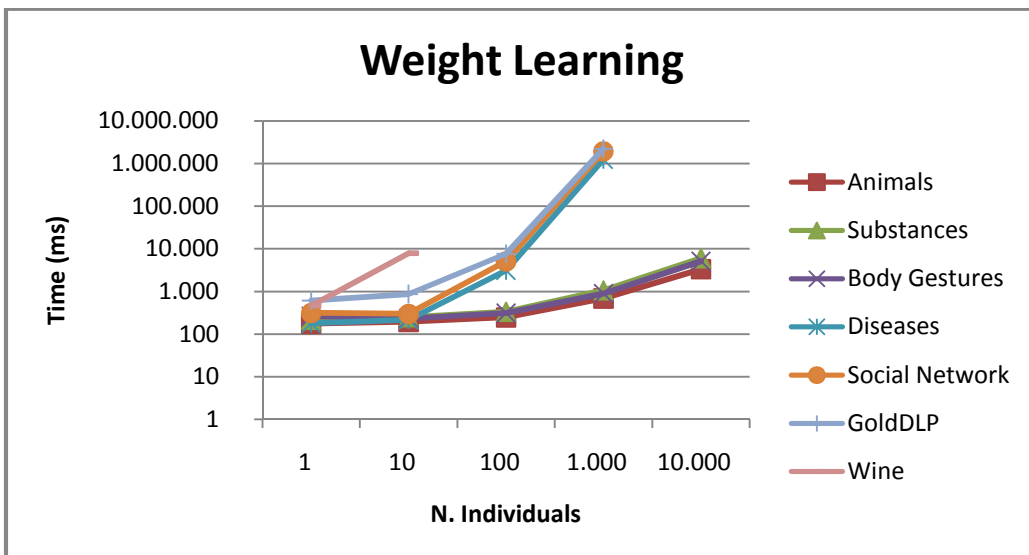


Figure 10. Markov logic weight learning scalability results (logarithmic scale).

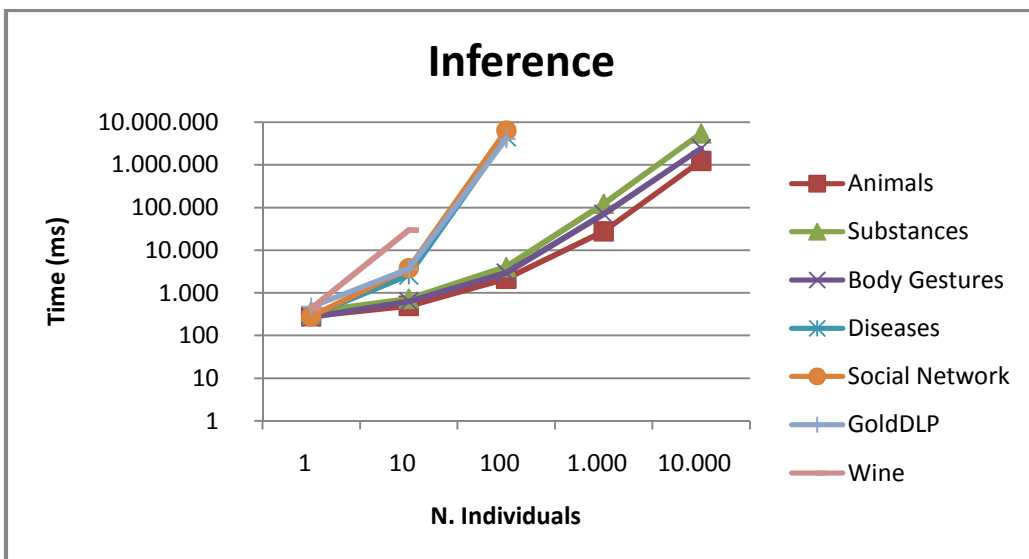


Figure 11. Markov logic inference scalability results (logarithmic scale).

3.5. Final Remarks

The main issue addressed in this chapter is: how can an OWL2 ontology be transformed into a MLN so we can use Markov logic capabilities? As seen, MLNs formulas can be acquired by interpreting ontology's axioms as first-order logic formulas. For the MLN weights, several approaches were proposed: interpret ontology's uncertainty annotations as weights; use ontology individuals to learn the weights; or learn probabilities and individuals of the ontology and use them to learn the weights.

As seen, each of these approaches has its advantages and limitations, being appropriate for special situations. By analyzing them, two conclusions can be drawn. The first is that the more complex they are, the more probable it is that the quality of results declines. For example, if we have an ontology already with individuals, and we learn the weights using those individuals, the results are probably better than by learning the weights using individuals learned by a web search engine. This result leads to the second result, that the less information we have about the ontology, the worst will be the quality of the results. For example, having an ontology with both uncertainty annotations and individuals is better than have the same ontology only with individuals. Both facts must be taken in account when choosing any of the proposed approaches.

4. Conclusions

To realize the Semantic Web vision of a world where knowledge is the most important element, mechanisms must be developed to represent and reason about the uncertainty that this knowledge could arise. In this thesis, we explored the use of Markov logic, a unifying representation of logic and probability, to learn and reason about uncertainty in the Semantic Web. The main contributions of this thesis are:

- Through Markov logic, we applied the ideas of a new field, statistical relational learning, which has been developing mechanisms to learn and reason in large uncertain relational domains. These characteristics are the same of the Semantic Web;
- Unlike other approaches, we studied not only reasoning about uncertainty in the Semantic Web, but also how we can learn that uncertainty from various sources, like ontology individuals, textual corpus, and web search engines;
- Unlike other approaches that also use probabilistic graphical models, our approach is based on Markov networks, which are undirected probabilistic models that allow cyclic knowledge;
- We developed a new method to transform probabilities in Markov logic weights. This method is more efficient than other approaches, and has no restrictions on his application;
- We developed a new method to learn the probabilities of OWL2 axioms using a web search engine. This method uses semantic similarity techniques, and can be applied in several domains where there is no other information about the uncertainty of the ontology;
- Our system, *Incerto*, can be used in many crucial domains for the Semantic Web, like ontology learning, social networks analysis, and ontology individual clustering.

In fact, since Markov logic is a so broad approach, we think that our approach of using Markov logic learning and reasoning capabilities in the Semantic Web can be seen as an introductory step in providing a general Markov logic framework for the Semantic Web (Pedro Domingos, Lowd, et al. 2008), providing services like ontology learning, reasoning, mapping, refining, among others.

However, currently, Markov logic contains some problems that interfere with its use in many Semantic Web domains. The high complexity on reasoning with transitivity and cardinality restrictions largely restricts its appliance in many ontologies. The lack of the definition of uncertainty information about evidential knowledge (i.e., we can only define the uncertainty of relations between classes and properties, and not about individuals belonging to a certain class or property) also reduces its usability in several domains, like ontology population and learning. To realize the vision of Markov logic as a general framework for the Semantic Web, these problems must be solved.

4.1. Future Work

In this section, we identify some directions for future work. Some of these directions have the objective of improving the executed work, while others explore some new interesting ideas and concepts that aroused during this work.

4.1.1. General Ideas

In general, there are several issues that deserve further exploration. The most obvious is the appliance of the presented techniques in more domains. We have applied our approaches in several relevant domains for the Semantic Web, like reasoning about automatically learned ontologies (Section 3.1.2) and web-based social networks (Section 3.2.1). More domains and tasks, like mapping and aligning ontologies (Euzenat and Shvaiko 2007), and automatically learning and augmenting ontologies (e.g., (Philipp Cimiano 2006), (Maedche 2002), (Suchanek, Sozio, and Weikum 2009)) could largely benefit from Markov logic capabilities.

In this thesis, we explored several ways to automatically learn the uncertainty of ontology axioms. However, more ideas could be explored:

- *Other ways of learning individuals.* We explored the use of textual resources to learn ontology individuals. Other way of populating ontologies is through the analysis of structured data, like relational databases or other ontologies. In this case, *mappings* (Euzenat and Shvaiko 2007) must be made between the structured data objects and the entities of the ontology.
- *Learn the uncertainties directly from textual corpus.* This is done by analyzing textual resources for patterns like “70% of A is B” or “Most of the A’s are B’s”. This can be done by using previously trained classifiers or general lexico-syntactic rules.
- *Use the structure of the ontology.* The structure of the ontology can provide interesting information about the uncertainty of its axioms. Some other works (Gomes 2004) (Stuckenschmidt and Klein 2004) (Ramakrishnan et al. 2005) already explored similar approaches in ontologies, however with distinct objectives than ours. The field of *network analysis* (Brandes and Erlebach 2005) can provide us with some interesting concepts that can be potentially transferred to our specific case.
- *Collective learning of weights.* The idea is to learn the weights collectively from multiple ontologies about the same domain. This task can be achieved by exploring techniques from collective learning fields, like *relational reinforcement learning* (Tadepalli, Givan, and Driessens 2004).
- *Trust propagation.* The idea is to use the propagation of *trust metrics* in groups to automatically learn the uncertainty of certain axioms. This idea was already applied in the Markov logic context (M. Richardson 2004).

As seen in Section 3.4.1, there are some OWL2 properties, like *transitivity* and *cardinality restrictions*, which make reasoning in Markov logic very difficult in most of the domains. Some recent works (P. Singla and P. Domingos 2008) (Jain and Beetz 2008) are already exploring new techniques to cope with this kind of problems, and we plan to apply them in the future.

4.1.2. Probabilistic Reasoning in Uncertainty-annotated Ontologies

One of the properties of the proposed approach to interpret probabilities as weights in Markov logic that needs further study is the influence of the number of individuals used in the learning process. During our experimentation, we found that in certain domains the number of individuals used in the learning process largely influences the learned weights. These weights, sometimes, did not reflect the desired probabilities, giving very polarized probabilities during inference. It is studied (Jain, Kirchlechner, and Beetz 2007) that the number of individuals can influence the learned weights in Markov logic, and we plan to study this situation in more depth in the future. We also plan to study the difficulties of convergence of the discriminative learning algorithm in some domains, fact that influences the quality of the learned weights.

4.1.3. Probabilistic Reasoning by Learning Individuals/Probabilities

Learn Individuals

Several improvements can be made to the ontology population process. Improving the linguistic processing of the corpus, for example by adding support to others languages besides English and adding stemmers to help lemmatizers, are some of these improvements. Other tasks related to the mapping of the learned individuals to the classes and individuals already existent in the source ontologies, like *coreference resolution* (P. Singla and P. Domingos 2006b) (Culotta, Wick, Hall, and McCallum 2007), *word-sense disambiguation* (Navigli 2009), and *canonicalization* (Culotta, Wick, Hall, Marzilli, et al. 2007), could also improve the quality of the results. One interesting idea, as proposed by (Suchanek, Sozio, and Weikum 2009), is to use the ontology structure and the already available individuals to guide the learning process. Other improvements, like learning new lexico-syntactic patterns and develop specific extractors to parse lists (Etzioni et al. 2005), could also improve the number of extracted individuals.

Learn Probabilities

In this thesis, we explored 6 semantic similarity metrics to learn axioms probabilities using web search engines. In the future, more metrics (e.g., (Lin 1998) (Bollegala, Matsuo, and Ishizuka 2007)) could be also explored. More complex ways of normalizing metrics results, like the ones that use *Naïve Bayes Classifiers* (Etzioni et al. 2005), even if more complex, usually give better results, and could be an issue for further exploration.

4.1.4. System

One of the most interesting future tasks is to develop a domain-specific Java Markov logic reasoning engine that could be easily incorporated in the existing system. This engine would be optimized and specifically developed for the Semantic Web domain, providing only the algorithms that perform well and are needed in this domain. This engine would also solve one of the major bottlenecks of the system, which is the need to communicate with external processes, since the available Markov logic engines were not developed in Java.

References

- Baader, Franz, Diego Calvanese, Deborah L. McGuinness, Daniele Nardi, and Peter F. Patel-Schneider. 2007. *The Description Logic Handbook: Theory, Implementation, and Applications*. 2nd ed. Cambridge University Press.
- Bar-Yossef, Ziv, and Maxim Gurevich. 2008. Random sampling from a search engine's index. *Journal of the ACM* 55, no. 5: 1-74.
- Berners-Lee, T., J. Hendler, and O. Lassila. 2001. The Semantic Web. *Scientific American* 284, no. 5: 28-37.
- Besag, J. 1975. Statistical analysis of non-lattice data. *The Statistician* 24, no. 3: 179-195.
- Bobillo, F., and U. Straccia. 2008. fuzzyDL: An expressive fuzzy description logic reasoner. In *Proceeding of the IEEE International Conference on Fuzzy Systems*, 923-930.
- Bollegala, D, Y Matsuo, and M Ishizuka. 2007. Measuring semantic similarity between words using web search engines. In *Proceedings of the 16th international conference on World Wide Web*, 757-766. Banff, Alberta, Canada: ACM.
- Brandes, Ulrik, and Thomas Erlebach. 2005. *Network Analysis: Methodological Foundations*. 1st ed. Springer, March 24.
- Church, K. W., and P. Hanks. 1990. Word association norms, mutual information, and lexicography. *Computational linguistics* 16, no. 1: 22-29.
- Cilibrasi, R., and P. M. B. Vitanyi. 2004. The google similarity distance. *Arxiv preprint cs/0412098*.
- Cimiano, P., G. Ladwig, and S. Staab. 2005. Gimme' The Context: Context driven Automatic Semantic Annotation with CPANKOW. In *Proceedings of the WWW*.
- Cimiano, Philipp. 2006. *Ontology Learning and Population from Text: Algorithms, Evaluation and Applications*. 1st ed. Springer.
- Costa, P. C. G., and K. J. Laskey. 2005. PR-OWL: A Bayesian Ontology Language for the Semantic Web. In *Proceedings of the 1st Workshop on Uncertainty Reasoning for the Semantic Web (URSW 2005)*, 6-10.
- Culotta, A., M. Wick, R. Hall, M. Marzilli, and A. McCallum. 2007. Canonicalization of database records using adaptive similarity measures. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, 201-209. ACM New York, NY, USA.
- Culotta, A., M. Wick, R. Hall, and A. McCallum. 2007. First-order probabilistic models for coreference resolution. In *Proceedings of NAACL HLT*, 81-88.
- Ding, Zhongli, Yun Peng, and Rong Pan. 2006. BayesOWL: Uncertainty Modeling in Semantic Web Ontologies. In *Soft Computing in Ontologies and Semantic Web*, 3-29.
- Domingos, Pedro, Stanley Kok, Daniel Lowd, Hoifung Poon, Matthew Richardson, and Parag Singla. 2008. Markov Logic. In *Probabilistic Inductive Logic Programming*, 92-117.
- Domingos, Pedro, Daniel Lowd, Stanley Kok, Hoifung Poon, Matthew Richardson, and Parag Singla. 2008. Just Add Weights: Markov Logic for the Semantic Web. In *Uncertainty Reasoning for the Semantic Web I*, 1-25.
- Etzioni, O., M. Cafarella, D. Downey, A. M. Popescu, T. Shaked, S. Soderland, D. S. Weld, and A. Yates. 2005. Unsupervised named-entity extraction from the web: An experimental study. *Artificial Intelligence* 165, no. 1: 91-134.
- Euzenat, Jérôme, and Pavel Shvaiko. 2007. *Ontology Matching*. 1st ed. Springer.
- Evans, R., and S. Street. 2004. A framework for named entity recognition in the open domain. In *Proceedings of Recent Advances in Natural Language Processing (RANLP-2003)*, 137-144. Borovetz, Bulgaria, September.

-
- Fukushige, Y. 2005. Representing Probabilistic Relations in RDF. In *Proceedings of Workshop on Uncertainty Reasoning for the Semantic Web (URSW)*.
- Getoor, Lise. 2003. Link mining: a new data mining challenge. *SIGKDD Explor. Newsl.* 5, no. 1: 84-89.
- Getoor, Lise, and Christopher P. Diehl. 2005. Link mining: a survey. *SIGKDD Explor. Newsl.* 7, no. 2: 3-12.
- Getoor, Lise, and Ben Taskar. 2007. *Introduction to Statistical Relational Learning*. The MIT Press.
- Giugno, Rosalba, and Thomas Lukasiewicz. 2002. P-SHOQ(D): A Probabilistic Extension of SHOQ(D) for Probabilistic Ontologies in the Semantic Web. In *Proceedings of the European Conference on Logics in Artificial Intelligence*, 86-97. Springer-Verlag.
- Gomes, Paulo. 2004. Software Design Retrieval Using Bayesian Networks and WordNet. In *Advances in Case-Based Reasoning*, 184-197.
- Grau, B. C., I. Horrocks, B. Motik, B. Parsia, P. Patel-Schneider, and U. Sattler. 2008. OWL 2: The next step for OWL. *Web Semantics: Science, Services and Agents on the World Wide Web*.
- Gu, T., H. K. Pung, D. Zhang, and G. Kotsis. 2004. A Bayesian approach for dealing with uncertain contexts. In *The Proceeding of the Second International Conference on Pervasive Computing*.
- Hearst, M. A. 1992. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the 14th conference on Computational linguistics-Volume 2*, 539-545. Association for Computational Linguistics Morristown, NJ, USA.
- Hendler, James. 2001. Agents and the Semantic Web. *IEEE Intelligent Systems* 16, no. 2: 30-37.
- Henrik, Nottelmann, and Fuhr Norbert. 2006. Adding Probabilities and Rules to Owl Lite Subsets Based on Probabilistic Datalog. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems* 14, no. 1: 17-42.
- Holi, Markus, and Eero Hyvönen. 2006. Modeling Uncertainty in Semantic Web Taxonomies. In *Soft Computing in Ontologies and Semantic Web*, 31-46.
- Jain, Dominik, and Michael Beetz. 2008. Towards Accurate Models of Joint Probability Distributions in Relational Domains: Cardinality Constraints and Dynamic Parameters. Technical Report. Technische Universität München.
- Jain, Dominik, Bernhard Kirchlechner, and Michael Beetz. 2007. Extending Markov Logic to Model Probability Distributions in Relational Domains. In *KI 2007: Advances in Artificial Intelligence*, 129-143.
- Kersten, M., and G. C. Murphy. 2006. Using task context to improve programmer productivity. In *Proceedings of the 13th ACM SIGSOFT 14th International Symposium on Foundations of Software Engineering*, 1-11. ACM Press New York, NY, USA.
- Klinov, Pavel. 2008. Pronto: A Non-monotonic Probabilistic Description Logic Reasoner. In *The Semantic Web: Research and Applications*, 822-826.
- Klinov, Pavel, and Bijan Parsia. 2008. Probabilistic Modeling and OWL: A User Oriented Introduction to P-SHIQ(D). In *Proc. of the Fifth OWL: Experiences and Directions Workshop 2008 (OWLED'08)*. October 26.
- Klir, George J., and Bo Yuan. 1995. *Fuzzy Sets and Fuzzy Logic: Theory and Applications*. 1st ed. Prentice Hall PTR.
- Kok, S., and P. Domingos. 2005. Learning the structure of Markov logic networks. In *Proceedings of the Twenty-Second International Conference on Machine Learning*, 22:441-448.
- Kok, S., P. Singla, M. Richardson, and P. Domingos. 2007. *The Alchemy system for statistical relational AI*. Technical report, Department of Computer Science and Engineering, University of Washington, Seattle, WA. <http://alchemy.cs.washington.edu>.
- de Kunder, Maurice. 2009. WorldWideWebSize.com | The size of the World Wide Web. <http://www.worldwidewebsite.com/>.

-
- Laskey, K. B., and P. C. G. Costa. 2005. Of Klingons and Starships: Bayesian Logic for the 23rd Century. In *Proceedings of the Twenty-first Conference on Uncertainty in Artificial Intelligence*. AUAI Press.
- Lin, D. 1998. An information-theoretic definition of similarity. In *Proceedings of the Fifteenth International Conference on Machine Learning*, 296-304.
- Lukasiewicz, T. 2008. Expressive probabilistic description logics. *Artificial Intelligence* 172, no. 6-7: 852-883.
- Lukasiewicz, Thomas, and Umberto Straccia. 2008. Managing Uncertainty and Vagueness in Description Logics for the Semantic Web. *Web Semantics Sci Serv Agents World Wide Web*.
- Maedche, Alexander. 2002. *Ontology Learning for the Semantic Web*. 1st ed. Springer.
- Magnini, B., M. Negri, R. Prevete, and H. Tanev. 2002. Is it the right answer? Exploiting web redundancy for answer validation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, 425-432.
- Marques de Sá, J.P. . 2001. *Pattern Recognition: Concepts, Methods and Applications*. 1st ed. Springer.
- McDowell, L. K., and M. Cafarella. 2008. Ontology-driven, unsupervised instance population. *Web Semantics: Science, Services and Agents on the World Wide Web* 6, no. 3: 218-236.
- Mika, Peter. 2007. *Social Networks and the Semantic Web*. 1st ed. Springer.
- Milch, B., B. Marthi, S. Russell, D. Sontag, D. L. Ong, and A. Rolobov. 2007. BLOG: Probabilistic Models with Unknown Objects. In *Introduction to Statistical Relational Learning*, 373-398. The MIT Press.
- Milch, B., and S. Russell. 2006. General-purpose MCMC inference over relational structures. In *Proc. 22nd Conference on Uncertainty in Artificial Intelligence*, 349-358.
- Milch, B., L. S. Zettlemoyer, K. Kersting, M. Haimes, and L. P. Kaelbling. 2008. Lifted Probabilistic Inference with Counting Formulas. In *Proceedings of the 23rd AAAI Conference on Artificial Intelligence*, 1062-1068.
- Nath, T. H., and R. Moller. 2008. ContraBovemRufum: A System for Probabilistic Lexicographic Entailment. *Proceedings of the 21st International Workshop on Description Logics (DL2008)*.
- Navigli, Roberto. 2009. Word sense disambiguation: A survey. *ACM Comput. Surv.* 41, no. 2: 1-69.
- Nottelmann, Henrik, and Umberto Straccia. 2006. A probabilistic, logic-based framework for automated Web directory alignment. *Soft computing in ontologies and the semantic Web. Studies in fuzziness and soft computing*: 47-77.
- Pan, J. Z., G. Stamou, G. Stoilos, and E. Thomas. 2007. Expressive Querying over Fuzzy DL-Lite Ontologies. In *Proceedings of the International Workshop on Description Logics (DL 2007)*.
- Pan, Rong, Zhongli Ding, Yang Yu, and Yun Peng. 2005. A Bayesian Network Approach to Ontology Mapping. *Proceedings of the International Semantic Web Conference 2005*: 563-577.
- Pearl, Judea. 1988. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. 1st ed. Morgan Kaufmann.
- Picard, R. W. 1997. *Affective computing*. MIT press.
- Poon, H., and P. Domingos. 2006. Sound and Efficient Inference with Probabilistic and Deterministic Dependencies. In *Proceedings of the National Conference on Artificial Intelligence*, 21:458-463. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999.
- Predoiu, Livia, and Heiner Stuckenschmidt. 2007. A probabilistic Framework for Information Integration and Retrieval on the Semantic Web. In *Proc. of 3rd International Workshop on Database Interoperability (InterDB) in conjunction with the VLDB conference*, 23-28. Vienna, Austria.

-
- Ramakrishnan, Cartic, William H. Milnor, Matthew Perry, and Amit P. Sheth. 2005. Discovering informative connection subgraphs in multi-relational graphs. *SIGKDD Explor. Newsl.* 7, no. 2: 56-63.
- Rehman Abbasi, Abdul, Nitin V. Afzulpurkar, and Takeaki Uno. 2008. Exploring Un-Intentional Body Gestures for Affective System Design . In *Affective Computing*. Jimmy Or. InTech Education and Publishing , May.
- Richardson, M. 2004. Learning and Inference in Collective Knowledge Bases. PhD Thesis, University of Washington.
- Roller, D., N. Friedman, L. Getoor, and B. Taskar. 2007. Graphical Models in a Nutshell. In *Introduction to Statistical Relational Learning*, 13-55. The MIT Press.
- Sanchez, Elie. 2006. *Fuzzy Logic and the Semantic Web*. 1st ed. Elsevier Science.
- Singla, P., and P. Domingos. 2005. Discriminative Training of Markov Logic Networks. In *Proceedings of the National Conference on Artificial Intelligence*, 20:868. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999.
- . 2006a. Memory-Efficient Inference in Relational Domains. In *Proceedings of the National Conference on Artificial Intelligence*, 21:488. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999.
- . 2006b. Entity resolution with markov logic. In *Proceedings of the Sixth IEEE International Conference on Data Mining*, 572–582.
- . 2008. Lifted first-order belief propagation. In *Proceedings of the Twenty-Third National Conference on Artificial Intelligence*. Chicago, IL: AAAI Press.
- Sirin, E., B. Parsia, B. C. Grau, A. Kalyanpur, and Y. Katz. 2007. Pellet: A practical OWL-DL reasoner. *Web Semantics: Science, Services and Agents on the World Wide Web* 5, no. 2: 51-53.
- Stoilos, G., G. Stamou, V. Tzouvaras, J. Z. Pan, and I. Horrocks. 2005a. Fuzzy OWL: Uncertainty and the Semantic Web. In *Proceedings of the International Workshop on OWL: Experiences and Directions*.
- . 2005b. The fuzzy description logic f-SHIN. In *Proc. of the International Workshop on Uncertainty Reasoning for the Semantic Web*, 67–76.
- Straccia, U. 2006a. A Fuzzy Description Logic for the Semantic Web. *Fuzzy Logic and the Semantic Web*: 73-90.
- . 2006b. Answering vague queries in fuzzy DL-Lite. In *Proceedings of the 11th International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems (IPMU-06)*, 2238–2245.
- Stuckenschmidt, Heiner, and Michel Klein. 2004. Structure-Based Partitioning of Large Concept Hierarchies. In *The Semantic Web – ISWC 2004*, 289-303.
- Suchanek, Fabian M, Mauro Sozio, and Gerhard Weikum. 2009. SOFIE: A Self-Organizing Framework for Information Extraction. In . Madrid, Spain, April.
- Tadepalli, P., R. Givan, and K. Driessens. 2004. Relational Reinforcement Learning: An Overview. In *Proceedings of the ICML'04 Workshop on Relational Reinforcement Learning*, 4:1–9.
- Tanev, H., and B. Magnini. 2006. Weakly supervised approaches for ontology population. *Proceedings of EACL-2006, Trento*: 3-7.
- Tversky, Amos, and Daniel Kahneman. 1974. Judgment under Uncertainty: Heuristics and Biases. *Science* 185, no. 4157: 1124-1131.
- Udrea, O., V. S. Subrahmanian, and Z. Majkic. 2006. Probabilistic RDF. In *IEEE International Conference on Information Reuse and Integration*, 172-177.
- Van Dongen, Stijn. 2000. Graph Clustering by Flow Simulation. PhD Thesis, University of Utrecht, May.

-
- Wei, W., J. Erenrich, and B. Selman. 2004. Towards Efficient Sampling: Exploiting Random Walk Strategies. In *Proceedings of the National Conference on Artificial Intelligence*, 670-676. San Jose, CA: AAAI Press.
- Yang, Y., and J. Calmet. 2005. OntoBayes: An Ontology-Driven Uncertainty Model. In *Proceedings of the International Conference on Computational Intelligence for Modelling, Control and Automation and International Conference on Intelligent Agents, Web Technologies and Internet Commerce*, 1:457-463. IEEE Computer Society Washington, DC, USA.
- Zhao, Y., and G. Karypis. 2005. *Criterion functions for document clustering*. University of Minnesota.