# Ontology-based Error Detection in Text

Fernando Gutierrez

Towards Partial Completion of the Comprehensive Area Exam

Department of Computer and Information Science
University of Oregon

Committee:
Dr. Dejing Dou, Chair
Dr. Stephen Fickas
Dr. Daniel Lowd

February 7, 2014

## Abstract

In general, research related to text analysis assumes that the information contained in text form, although ambiguous, is correct with respect to the domain to which that text belongs to. This assumption comes in part from the fact that text analysis has historically been done over scientific documents. As the trend of taking text understanding to broader domains, such as Internet, we need to consider the presence of incorrect text in our data set. By incorrect text, we refer to a natural language text statement which is either false or contradicts the knowledge of the domain.

We propose the use of Ontology-based Information Extraction (OBIE) to identify incorrect statement in a text. OBIE, a subfield of Information Extraction (IE), uses the formal and explicit specification provided by an ontology to guide the Information Extraction process. OBIE can capture the semantic elements of the text through its IE component, and it can determine if these semantics contradicts the domain through its ontology component, concluding if the text is correct or incorrect.

In the present work, we review the most important topics of Ontology Inconsistency that can be relevant for the task of identifying and explaining incorrect statements, and we also review of the most relevant Information Extraction research. We believe that research in the detection of logic contradiction in ontologies (i.e., Ontology Inconsistency) can provide us with useful insight into identifying incorrect text and determining the specific elements (e.g., axioms) that participate in the contradiction. On the other hand, research in Information Extraction (and OBIE) can provide us awareness about the complexity of the analysis that can be performed on the text, given the semantics that can be extracted from it.

# Contents

## 1 Introduction

In general, research related to text analysis assumes that the information contained in text form, although ambiguous, is correct with respect to the domain to which that text belongs to. This assumption comes in part from the fact that text analysis has historically been done over scientific documents [34, 66, 68, 69]. Since most of these analysis try to understand text through the discovery of relationships between instances and concepts, it is reasonable to focus on reliable data for the process. Under the previously described scenario, the incorporation of mechanisms to deal with incorrectness in text appear to be unnecessary. However, the correctness of scientific documents cannot be guaranteed even in the presence of peer reviewed research [46]. Furthermore, as the trend of taking text understanding to broader domains, such as Internet [3, 17, 50], we need to consider the presence of incorrect text in our data set. By incorrect text, we refer to a natural language text statement which is either false or contradicts the knowledge of the domain.

Most of current research regarding text correctness comes from the educational domain in the form of assistance tools that can provide automatic evaluation over student writings. Summaries and essays provide an important mechanism to judge a student's understanding of a topic. Because they enhance long-term retention of information [40], it is an ideal tool when compared with other evaluation mechanisms (e.g., fill-in the blanks) [74]. However, automatic understanding of written text such as summaries and essays is a complex task given the underlying ambiguity of natural language. The advances in Natural Language Processing (NLP) have allowed major improvements in automatic text grading, with several commercial applications [7, 24, 25, 83].

Three main approaches can be identified for automatic text grading. The first approach is based on the identification of coincident words and n-grams. It includes machine translation techniques [56] and n-gram co-occurrence methods [44]. *N-gram co-occurrence* methods have shown to be adequate for automatic text grading, especially when evaluating characteristics such as the fluency of the text. The second approach uses *Natural Language Processing* (NLP) techniques, with the most popular being Latent Semantic Analysis (LSA) [24, 25]. LSA treats the text as a matrix of word frequencies and applies Singular Value Decomposition (SVD) to the matrix to find an underlying semantic space. Based on the distance between the vector representations of the student's document and a *golden standard* (i.e., correct document) in this semantic space, the similarity of the documents is estimated, which can then be transformed into a grade. LSA has shown to be quite accurate when compared with human grading. Finally, the third approach is based on Information Extraction (IE) [7, 51], which intends to capture the underlying semantics of the text. Because automatic text grading needs to determine the meaning (i.e., semantics) behind the student's writing, IE is a natural choice for analyzing text. It uses NLP tools to identify relevant elements from the text, such as concepts, individuals and relationships. These semantic elements can provide a structural representation of the information in the text.

All of the previously mentioned methods share a common characteristic, they do not have an effective way of determining what is incorrect in a text. Methods such as *n-gram co-occurrence* and IE-based can identify specific elements in the text from previously known patterns (e.g., n-gram methods) or expected domain elements (e.g., IE regular expression patterns). So, in order to find incorrect text, these methods would require a *reference of incorrectness*, such as text with incorrect statements or incorrect facts of domain knowledge. However, because the incorrectness of a statement can be originated by many different factors, this *reference of incorrectness* would need to be very large to provide useful coverage of content, which would be impractical. In the case of LSA-based methods, because the correctness of the text is measure regarding its similarity to a *golden standard*, determining if a text is incorrect is even more difficult. It is possible to argue that a low similarity is an indication of incorrectness. Yet, even a correct text can obtain a low similarity with respect to the golden standard, if it is written in an unexpected fashion or contain more information [70].

Because we have defined incorrect text as a false or contradicting statement, it is reasonable to consider logic as a mechanism to identify it. Through logic, the truth value of a statement (i.e., if statement is true or false) can be determine from a set of facts. However, we need to know the truth value of these facts in order verify if the statement from the text is false or not. So, the information contained in the statement itself is not sufficient to conclude if its incorrect. Through logic, we can also determine if a statement is a contradiction. Logic contradiction in text is studied by Contradiction Detection, which tries to identify pairs of sentences that are *very unlikely* to be true at the same time [14]. It can uses syntactic and lexical elements from the text [14], or background statistical knowledge [60] to determine if a pair of sentences contradict each other. However, with only information from the text itself to support the validity of the pair of contradicting statements, Contradiction Detection cannot determine with certainty which of the statements is false.

Domain knowledge can provide us with the required semantics to determine if a text is false or a contradiction. An ideal formal model for representing the semantics of domain knowledge is *ontology*, which is an explicit specification of a conceptualization [27]. It represents knowledge through concepts, relationships between concepts, and constraints. In Artificial Intelligence, ontologies can represent domain knowledge in a formal model that permits logical reasoning. By including the domain ontology in the analysis of a text, we can verify the correctness of the content by determining its logical implication with respect to the domain it belongs to. If a statement is a false logical consequence of the domain knowledge, or if it is a logical contradiction of the domain knowledge, we consider the statement to be incorrect.

However, because the ontology itself does not have the tools to actually analyze text, we need a mechanism to capture the semantics of the text in a *logic friendly way*. Some of the previously mentioned text analysis methods produce mathematical representations of the text, such as spacial representation (e.g., LSA), to study its content. On the other hand, IE transform the text into a set of entities and relationships between entities, which can easily be mapped into a logic based representation (i.e., ontology) in the form of *logic axioms*. This connection has lead to the use of IE with ontologies in the research area known as Ontology-based Information Extraction (OBIE) to identify incorrect statement in a text. OBIE, a subfield of IE, uses the formal and explicit specification provided by an ontology to guide the Information Extraction process [77].

We propose the use of OBIE to identify incorrect statement in a text. OBIE can capture the semantic elements of the text through its IE component, and it determine if these semantics contradicts the domain through its ontology component, determining if the text is correct or incorrect. We call this approach for identifying incorrect text Ontology-based Error Detection. We believe that research in the detection of logic contradiction in ontologies (i.e., Ontology Inconsistency [35, 61]) can provide us with useful insight into identifying incorrect text and determining the specific elements (e.g., axioms) that participate in the contradiction. On the other hand, research in Information Extraction (and OBIE) can provide us awareness about the complexity of the analysis that can be performed on the text [4, 79], given the semantics that can be extracted from it.

The reset of the paper is organized as follows. In Section 2, we identify and review the most important topics of Ontology Inconsistency that can be relevant for the task of identifying and explaining incorrect statements. In Section 3, we present a review of the most relevant Information Extraction research. In Section 4, we offer possible solutions for different expected scenarios.

## 2 Ontology Inconsistency

In Artificial Intelligence, an ontology is an explicit specification of a conceptualization [27]. This conceptualization provides a formal knowledge representation through *concepts* from a domain, and *relationships* between these concepts. The term ontology comes from philosophy, where it corresponds to the study of existence or reality, and as Gruber points out "For knowledge-based systems, what exists is exactly that which can be represented" [27]. Through concepts, individuals of these concepts, relations, and constraints, an ontology provides a vocabulary and a model of the domain it represents. Because of this domain model, it is possible to perform inference. In this work, we consider *Description Logic* based ontologies, as those described through the *Web Ontology Language* (OWL) [63]. OWL is the standard ontology language proposed by the *World Wide Web Consortium* (W3C) [1]. Description logic (DL) is a fragment of *first-order logic* that is decidable, and it has sound and complete reasoners [31, 54, 71].

### 2.1 Description Logic

Description Logic (DL) is a family of logic languages that are used for formal knowledge representation. Each DL language offers different *expressivity*. Some DL can express only atomic negation while others allow hierarchies of relationships (properties). The expressivity of a DL language is tightly connected to the complexity of reasoning on it. In order to enable more efficient reasoning services, DL languages have to trade off expressivity [8, 9]. This issue becomes more significant when we consider that DL is the logical backbone for ontological languages such as OWL, which is intended to provide formal representation for large domains [36].

DL consists of an intentional part (i.e., terminology or *TBox*) and an extensional part (i.e., assertion or *ABox*). The *TBox* has intentional knowledge, as concept definitions, in the form of axioms. The axioms express how complex concepts are built (e.g., through union or intersection of atomic concepts) or the relationships between concepts (e.g., concept C is equivalent to concept D). The *ABox* has extensional knowledge which describes individuals, and it is specific to a domain. The description of individuals correspond to the assertion instances $C(a)$ (or $a : C$) which indicates that "instance $a$ is a member of the concept $C$."

The semantics of the elements in the knowledge base is defined by an interpretation ($\mathcal{I}$). The interpretation consists of a domain ($\Delta$) and a mapping function or interpretation($\cdot^{\mathcal{I}}$). The function maps the axioms and assertions of the knowledge base to the domain. An ontology is *consistent* if it has an interpretation which satisfies all the ontological axioms. A concept is *satisfiable* with respect to the ontology if there is an interpretation of the ontology that maps the concept to the domain.

## 2.2 Inconsistency and Ontology

As stated by Haase and Volker [33], if there is a *logical contradiction* in an ontology, the ontology becomes meaningless because any type of statement can be derived from a set of logical axioms that contradict each other. This issue makes the task of understanding and detecting inconsistencies in an ontology vital for ontology dependent applications.

Flouris et al. [21] breaks down logical contradiction into *inconsistency* and *incoherency.* An inconsistency occurs when an instance of either a class or of a property contradicts an axiom of the ontology. More formally, an ontology is inconsistent if an axiom of the ontology is unsatisfiable. For example, consider an ontology with the disjoint concepts *Professor* and *Student*, and the instance *Student(Fernando).* If we add the instance *Professor(Fernando)*, the ontology will become inconsistent because disjoint concepts cannot share individuals or subconcepts. On the other hand, an incoherency occurs when an axiom of the ontology contradicts another axiom. Formally, an ontology is incoherent if there exists a concept that for any interpretation of the ontology, it leads to false. Consider the previous example of the ontology with the disjoint concepts *Professor* and *Student.* If we add to the ontology the concept *GTF* as a subclass of both *Student* and *Professor*, the ontology becomes incoherent.

Flouris et al. [21] notes that although these two type of logical contradictions can occur independently, they are highly related. If adding an element to an ontology keeps its consistency, then the ontology will maintain its coherency.

Because of this tight relation between the two types of contradictions, and because it most clearly evokes the state of lack of consistency, most authors define logical inconsistency of an ontology as the logical contradiction that does not permit any valid interpretation of its axioms (i.e., unsatisfiability) [15, 32, 33, 35, 37, 61, 62]. For this work, when referring to logical inconsistency, we will be considering the most general definition (logical contradiction).

## 2.3 Change and Inconsistency

As ontologies grow in size and complexity, their development and maintenance has led to interesting research problems, one of the most important being *ontology change* [22]. Ontology change addresses the generic problem of modifying an ontology to reflect changes in the domain of interest, to incorporate new elements to the ontology, or to correct design flaws. Yet, modifying an ontology can have both unexpected and undesired effects, such as the introduction of logical inconsistencies.

In their survey, Flouris et al. [22] identifies 10 subfields in ontology change that can be grouped into three main classes. The first group corresponds to heterogeneity resolution, which intends to solve the differences between two or more ontologies by creating mapping functions, discovering relationships between them, creating intermediary ontologies, or transforming their vocabularies. The second group, which includes ontology evolution, debugging and versioning, corresponds to the process of keeping or obtaining internal ontological consistency, either through change, or in between versions. Finally, the third group manages the combination of two or more ontologies into one from domains with high (merging) or low (integration) overlapping. Since we are interested in ontology inconsistency, we will focus on ontology evolution and ontology debugging.

Both ontology evolution and ontology debugging deal with the presence of inconsistency in an ontology. The difference is that in ontology evolution the inconsistency comes from modifying or adding new information into the ontology while in ontology debugging the inconsistency are already present in the ontology. As pointed out by Flouris et al. [22], since both fields focus on ontology contradiction, a solution to one of these problems should be applicable to the other.

It must be mentioned that although ontology change might also affect syntactical and application driven constraints of the ontology [5, 32], in the present work we focus on logical inconsistency.

## 2.4 Dealing with Ontology Inconsistency

Haase et al. [32] have identified three main approaches when dealing with inconsistency: preventing ontology inconsistency, fixing ontology inconsistency, or reasoning with inconsistency.

### 2.4.1 Preventing Ontology Inconsistency

We would prefer that when changing some aspect of an ontology, it can occur without affecting the consistency of the ontology. However, because of the nature of the underlying DL language, it is very likely that a modification of the ontology can lead to an inconsistency. *Ontology evolution* [32] intends to provide mechanisms to modify an ontology while keeping it consistent by managing change operations. In the same way, but with focus on knowledge bases, *belief revision* defines logical operators that perform consistent changes to the knowledge base [13, 26]. In both cases, the change must be minimal.

When performing a modification to an ontology, a contradiction might rise between the old information (in the ontology) and the new information (updated or added axioms or instances). The consistency can be maintained by either integrating into the ontology only the elements that do not conflict with it, or by removing from the ontology the elements that contradict the added information. Because ontology evolution intends to provide a general framework to manage change, users can define policies for both situations. In the case of belief revision, because most changes done to the knowledge bases take the form of updates, the operators intend to keep consistency with most recent information, such as Dalal *revision operator* [13].

In the first case, the consistency of the ontology is kept by integrating elements that do not produce contradictions. Under this approach, Haase and Stojanovic offer *minimal inconsistent subontology* for ontology evolution [32]. The method identifies this subontology by constructing an alternative ontology from the axioms of the original one. The first axioms added to the alternative ontology is the last axioms added to the original ontology. The following axioms that are selected must share an entity with any axiom in the alternative ontology. This approach guarantees that the search of the minimal inconsistent subontology occurs near the axiom that created the inconsistency (i.e., the last axiom added into the ontology). The method terminates when an inconsistent subontology is reached.

The second strategy for avoiding contradiction when modifying an ontology is to remove older axioms that are inconsistent with the most recent addition of the ontology. Following this strategy, Haase and Stojanovic propose a *maximal consistent subontology* to discover the origin of the inconsistency by removing axioms of the ontology that are connected with the last axiom added to the ontology [32]. On the other hand, Flouris et al. [21, 23] have proposed an adaptation of *AGM theory* into DL. Despite the shortcomings of not managing correctly all type of changes [41] and some triviality in how some changes occur [13], *AGM theory* is the dominant *belief revision* theory. This theory establishes operators to add and remove consistently and coherently elements from a knowledge base. However, AGM theory requires complex negation of clauses, which is not a common feature of all description logic languages. In order to overcome this limitation, Flouris et al. provide definitions for a consistent negation and for a coherent negation [21]. Although the negations do not guarantee the minimality of the change, they do follow the intuition behind

AGM *contraction operator*.

There are two points that might be important to consider about the presented methods. First, although these consistent change mechanisms try to keep as close as possible from the original ontology when performing modifications, the minimality of the change cannot be guaranteed because of the implicit effects of change that might affect the ontology [13, 21]. And second, in general the strategies of ontology evolution are mostly debugging techniques [32]. The methods mostly correct the ontology in case an inconsistency is reached, but they do not offer a consistent evolution of the ontology as offered by change operators. The main reason seems to be that Haase and Stojanovic allow the user to define the policies that must be followed to keep the consistency. Haase and Stojanovic define the evolution of an ontology as managing consistently change operations on the ontology. Their approach consists of an inconsistency detection component and change generator component. The inconsistency detection component determines if the ontology is inconsistent, and in the case it is inconsistent, what subpart of it is causing the inconsistency.

### 2.4.2 Fixing Ontology Inconsistency

In the case of an ontology that is already inconsistent, we can try to identify the origin of the inconsistency and determine a mechanism to correct the inconsistency. These two tasks are known as Ontology Debugging. In Section 2.5, we will provide a more in depth analysis of this task.

### 2.4.3 Reasoning with Inconsistency

In some cases, an ontology with logical contradictions cannot be corrected because of the ambiguity of the domain represented, or because correcting the ontology requires modifications that are not feasible. Although the ontology is not consistent, we still want to provide services when possible, such as reasoning. The intuition is that if there is a subontology that is consistent, then we can obtain correct answer from an inconsistent ontology.

Lembo et al. [43] propose inconsistency-tolerant semantics, where the *TBox* of the ontology is satisfiable, but the *ABox* is inconsistent. Lembo et al. propose two semantics to manage inconsistency. The first semantic, named *AR-semantics*, removes the minimal set of inconsistent instances. The second semantic, named *CAR-semantics*, it applies AR-semantics to *Herbrand Base* (grounding of all formulas) of the knowledge base. Since both AR-semantics and CAR-semantics are intractable, sound approximations are offered. The later work by Masotti et al. [48], offers an implementation of the approximation of AR-semantics and CAR-semantics for the DL language $DL - Lite_A$, which OWL2's profile QL is based on. The implementation is an iterative algorithm that uses some properties of the axioms to identify inconsistent elements in the ontology. The experiments with different levels of inconsistency and different sizes of knowledge base show that the AR-semantics approximation scales well and it is not affected by the level of inconsistency. On the other hand, the CAR-semantics approximation does not scale because of the computation required for the Herbrand Base.

In a more general case, where contradictions can appear in any part of the ontology, Huang et al. [37] and Liu et al. [45] propose the use of selection functions to obtain consistent reasoning. Based on ideas from *relevance reasoning* and *paraconsistent reasoning*, Huang et al. consider that an inconsistent reasoner is sound if a formula obtained from an inconsistent theory (ontology) follows from a consistent subtheory using classical reasoning. Since classical completeness is impossible in an inconsistent ontology, inconsistent reasoning considers local completeness, which is the entailment of formulas from the consistent subtheory. The inconsistent reasoner follows a consistent subset, which is iteratively incremented by a selection function that looks into directly related axioms that keep the consistency of the current sub theory. Liu et al. extend inconsistent reasoning by considering the confidence of the reasoning process. By associating a confidence value to each axiom (probabilistic logic), uncertainty is modeled into the reasoning process [45]. In each iteration, when a new axiom is added into the answer set, the confidence values are multiplied to produce the confidence value of the set.

## 2.5 Ontology Debugging

The process of correcting an inconsistent ontology is called *Ontology Debugging* [22]. Ontology Debugging has two main tasks: identifying the elements from the ontology that are causing the inconsistency, and correcting the inconsistency. In general, the first task of ontology debugging has become more relevant because of the overall complexity of identifying the elements that are causing the inconsistency while in most cases, the correction of the ontology can be gained by removing the inconsistent elements from the ontology.

We can identify two main approaches to ontology debugging: logic-based and non-logic based methods.

### 2.5.1 Logic-based methods

Logic based methods use properties of the underlying DL language to discover the inconsistency in the ontology. Usually, the inconsistency will be caused by a small part of the ontology. However, this small set can affect many different parts of the ontology, leading to many explanations of inconsistency. Because of this situation, ontology debugging solutions that focus on local clash of concepts (i.e., inconsistency) can only provide limited results [55]. Based on the definition of entailment justification by Kalyanpur et al. [39], Horridge et al. [35] identifies two types of inconsistent subsets of the ontology. First, we have *inconsistency justification*, which corresponds to an inconsistent subontology. The second is an *ontology repair*, which is the minimal set of inconsistency justifications. This minimal subset is called repair because, in essence, if it is removed from the inconsistent ontology, the resulting ontology will be consistent.

In order to find the origin of the inconsistency in an ontology, we need to first identify all the inconsistency justifications it contains. In the Schlobach and Cornet [61] approach for debugging inconsistent ontologies with unfoldable terminologies (atomic left-side defined acyclic axioms), for each unsatisfiable concept, they determine the *minimal unsatisfiability-preserving sub-TBoxes* (*MUPS*). The *MUPS* of a concept is the set of axioms that cause the concept to be unsatisfiable. In their original work, Schlobach and Cornet [61] obtained the *MUPS* of each concept through a modified $\mathcal{ALC}$ reasoner that inserted traceable labels in the axioms when performing consistency check. But because this approach does not scale well to more expressive DL languages, Schlobach et al. [62] offer an alternative mechanism to identify each concepts *MUPS*. Based on Huang et al. [37] selection function for reasoning with inconsistent ontologies, Schlobach et al. use an informed search to find concept-relevant axioms. The set produced by the selection function is then pruned by removing axioms that do not affect the unsatisfiability of the concepts *MUPS*.

In the case of Horridge et al. [35], the inconsistent subsets of the ontology are obtained by a modified version of the single justification algorithm, from the *entailment justification* method [39]. This algorithm identifies subsets of the ontology that are inconsistent through the division of the ontology. The intuition is that the cause of inconsistency will be in the inconsistent part of the ontology, and not in the consistent part. It is important to note that it is possible to remove accidentally the inconsistency when dividing the ontology. To avoid missing an inconsistent subset, the modified single justification also analyzes the recombination of the divided parts.

Once we have the set of inconsistency justifications, we need to determine the repair of the ontology. In the case of Schlobach and Cornet approach, from the MUPS the *minimal incoherence-preserving sub-TBoxes* (*MIPS*) are determined, which are unsatisfiable sub-TBoxes that can become satisfiable if one atomic concept is removed. Because each element of the *MIPS* set comes from some *MUPS*, the *MIPS* set is contained in the union of all *MUPS* of the original *TBox*. Although the *MIPS* already identifies the minimal inconsistent set of the ontology, Schlobach and Cornet offer an even more fine grained solution. Because inconsistencies can be interpreted as the effect of overspecified concepts, we can identify the actual concepts that are clashing by generalizing the axioms of the MIPS. This new set is obtained by the *generalized incoherence-preserving terminology* (GIT), where all elements in an axiom of the *MIPS*, which do not affect its unsatisfiability, are removed.

On the other hand, Horridge et al. use the Hitting Set Tree algorithm [58] to identify a repair in the inconsistent ontology from set of justifications of the inconsistency. Reiter propose the Hitting Set Tree (HST) [58] as a form to determinate the diagnosis of a faulty system. In a faulty system there can be multiple reasons that explain the actual error (i.e., *conflict sets*). Yet in order to correct or fix the system, it is necessary to identify the minimal conflict set (diagnosis). Reiters HST finds the diagnosis by learning how the conflict sets intersect. The HST algorithm iteratively searches or access the set of conflict sets, and it constructs a tree where each node indicates a conflict set, and the edges indicate an element of the conflict set. The set formed by the labels on the edges along a branch of the HST corresponds to one diagnosis. So, in the case of ontology inconsistency, the HST can identify the repair of an inconsistent ontology by constructing a tree with the inconsistent justifications.

Finally, it must be mentioned that although the two approaches previously presented have an exponential complexity, in most cases they can provide an answer in a reasonable amount of time. First of all, the exponential complexity of these methods comes mainly from the fact that they do consistency checking, which is a decidable but intractable problem. In the case of Schlobach and Cornet approach, when they create the *MUPS* the algorithm does a consistency check while labeling the axioms. In the case of Horridge et al., the simple justification algorithm performs many consistency checks in order to identify a justification. The HST algorithm includes a series of optimizations that intend to reduce the amount of justifications needed to complete the HST and avoid following non-interesting or repeated branches of the HST. However, experimental results in both works have shown that it is possible to obtain reasonable performance in most of the cases.

### 2.5.2 Non-logic methods

Two methods are presented that do not use the underlying logic as a main mechanism to determine the origin of the inconsistency. Although based on different approaches, the following methods can provide interesting insight into regaining the consistency of an ontology.

As a consequence of their observations in teaching ontology construction, Wang et al. [72] propose a heuristic approach to detect the cause of inconsistency in an ontology. The method, which is based on common errors, first identifies a minimal unsatisfiable core of the ontology. Then, the method tries to identify global conditions that might create general conflicts. These general conflicts propagate inconsistency through entailment or inheritance (i.e., properties passed through ISA relations between classes). By identifying these general conflicts, it is possible to reduce the size of the unsatisfiable core since any correction done to the general axioms (modification or removal) will affect the entailed axioms.

In order to find the unsatisfiable core and the most general conflict set, the authors define three categories of causes of unsatisfiability: local, propagated, and global. Local unsatisfiability is the most easy to detect, and it is directly produced by an individual or class not following a restriction. Propagated unsatisfiability can be produced by ancestor (e.g., subclass is unsatisfiability because its superclass is unsatisfiable), or by fillers of existential restriction (the filler is unsatisfiable). Global unsatisfiability are usually related to domain or range constraints that are not met by some class. Wang et al. have defined around 15 rules to detect causes of unsatisfiability. The authors point out that although the proposed method is not a complete solution (heuristic), it does cover a large solution space. Because the rules are based on common situation observed directly by the authors, other more complex and specific unsatisfiability cases might be missed by their proposed method.

On the other hand, Deng et al. [15] proposed the use of the *Shapley value*, which is a measure borrowed from game theory, to evaluate the level of inconsistency of an ontology. The Shapley value is a measurement from game theory that estimates the *contribution* of an agent in a multi-agent coalition. It considers all possible formation of the coalition of agents, and it averages each agent's *marginal contribution*. Deng et al. propose to treat the set of inconsistent axioms as a coalition of agents. The Shapley value will indicate the contribution of each axiom in the inconsistency of the ontology. The authors have defined functions to give numeric values to inconsistent sets of axioms, so it can be plug into the Shapley equation.

Because Shapley values take into account all possible combination of axioms to obtain the marginal contribution, the complexity of the proposed approach is exponential. The authors offer a method to avoid evaluating axioms that are not related to the inconsistency problem. They define direct structurally related axioms as axioms that share concept names and role names. A second optimization method that is proposed considers establishing convergent subset, which is the maximal inconsistent subset that need to have its value calculated. Once a subset is convergent, no larger subset needs to be computed. Both mechanisms provide upper bounds to the size of the subsets to be computed.

An interesting side effect of these two non-logic based methods is that they provide an implicit correction mechanism. In contrast with logic-based debugging mechanism, which can provide the minimal inconsistent subontology, the non-logic based methods point out which axioms can be removed from the ontology to regain consistency. In the case of Wang et al. heuristic method [72], when the inconsistency rules detects a problem, it will point to an axiom that is causing the inconsistency. For Deng et al. game theory approach [15], since there is a ranking of axioms that participate in the inconsistent subset of the ontology, the axiom with the highest Shapley value is the most responsible axioms for the inconsistency. Although these methods do not guarantee that consistency can be gained by removing the highlighted axioms, they do offer a reasonable hint in that direction.

## 3 Information Extraction

Information Extraction (IE) is the task of automatically acquiring knowledge from natural language text. In the process of extracting, IE attempts to retrieve specific elements from text such as concepts, relationships, or instances, and it leaves out irrelevant information to reduce the complexity associated to the task.

The main goal behind IE is to transform unstructured information (i.e., text) into structure information (databases, knowledge bases). However, this transformation of information is not a trivial process because of the inherent ambiguity of natural language. A fact can be stated in many ways, and a statement can have more that one interpretation. The complexity of extracting information from text has kept IE from being more widely adopted, with most

IE systems being implemented for specific domains.

In order to reduce the complexity of analyzing the text and identifying relevant elements in it, information extraction is divided into subtasks. Some of these tasks can be seen as steps that need to be fulfilled in order to perform the following task [2], but in most cases each task can be carried out independently [3, 50, 80]. Jurafsky and Martin [38] define the following Information Extraction tasks:

- **Named entity recognition**: is the process of detecting and classifying proper names. It usually consists in determining if a proper noun is the name of a person, place, and organization. A more specialized version of this task intends to identify names of genes and proteins [64].
- **Coreference resolution**: is the process of determining if the mention of a same or similar name refer to the same entity, and it includes the resolution of anaphoric references. This process is tightly related to name entity recognition.
- **Relationship extraction**: is the process of discovering semantic relations between entities in the text. This process has become one of the most researched sub areas of Information Extraction since it is fundamental for other tasks such as ontology learning [47], knowledge base population [10], and semantic annotation [50].
- **Event extraction**: is the process of identifying events that are related to the entities in the text. Similarly to entity recognition, there is a need for coreference resolution since many actors can be participating in an event, and the text can mention one or more events.
- **Temporal analysis**: is the process of determining what is the temporal relations between events. This task intends to identify temporal elements, such as date and time, that are related to events, and provide a resolution mechanism that allows ordering of such events.
- **Template filling**: is the process of identifying documents that have information in a form that is shared by other documents (i.e., stereotypical) which allows direct extraction of entities into templates.

As mentioned, in the present work we are interest in analyzing domain specific information that is present in the text, which can be mapped to an ontology. This type of information mostly appears in the form of a relationship between two concepts (property or subsumption relation) or between concept and individual (membership). Because of this situation, we will mostly focus on systems that do relationship extraction.

## 3.1 Ontology-based Information Extraction

Ontology-based Information Extraction (OBIE) is a subfield of IE, which uses an ontology to guide the information extraction process. As presented in Section 2, an ontology is defined as a formal and explicit specification of a shared conceptualization [27]. The concepts and relationships of this conceptualization are represented through classes, properties, instances and other type of axioms. This formal and explicit specification guides the extraction process in OBIE [77].

The presence of an ontology in the extraction process does not only provide guidance in the sense of indicating the specific sentences that need to be looked into; the ontology can also provide contextual or structural information that can enhance the extraction process. A clear example is the use of the concepts hierarchical structure to provide additional information to the extraction [30, 66]. If we know that the concept *Killer Whale* has type *Dolphin*, then we can use information from *Killer Whale* (e.g., an extractor for this concept) to extract objects related to *Dolphin*.

Ontologies in information extraction allows the possibility of Semantic Annotation. Semantic Annotation is the process of adding meta-data information that establishes relationships between unstructured data (text) and some entity that provides context. Although an ontology is not strictly required for semantic annotation, by annotating a text with ontological entities (formal annotation) provides formalism and structure of the ontology to the text, which is the main goal of the Semantic Web [53].

Even when the use of ontologies can improve the extraction process, it has become more evident in recent years that systems that can be classified as OBIE have been defined as information extraction systems by their authors. This trend might reflect the current approach of extraction systems that can be applied of open domains, such as the Internet. With that in mind, an ontology-based information extraction system seem constrained and without the flexibility to scale to the Web. However, we argue that any information extraction system that focuses on the extraction of relations can be more or less integrated into an ontology-based information extraction process.

## 3.2 Classification of Information Extraction

In their survey of Ontology-based Information Extraction, Wimalasuriya and Dou [77] offer classification of information extraction systems based on different characteristics of the systems, such as the extraction methods used, if the system constructs or updates the ontology, what type of components of the ontology are extracted, and the source of the text that is used by the system. Yet, current information extraction systems cannot be easily classified by any of these features. If we consider the extraction mechanism, most systems use a blend of techniques such as gazetteer list and linguistic features (part-of-speech, dependency parse trees) in rule pattern [20, 49] or as feature of a machine learning based extractor [30, 80]. Most approaches have focus on extracting instances of concepts and relationship [3], they use available ontologies [30] and knowledge bases [50], and the Internet is their corpus of analysis [20, 49, 81].

Because most recently IE systems are being applied over very large corpus, such as Internet, a new characteristic has risen that allows to differentiate between type of extraction systems: the amount of human intervention in the preparation and deployment of the system. This factor has led to three strategies for IE: supervised, semi-supervised, and unsupervised. In some cases, a fourth type of information extraction system has been proposed: self-supervised systems. In self supervised systems, the data set used for training the information extractors is generated by the system itself. However, if we pay attention to the mechanism of the system, it is possible to distinguish elements that will classify it as either semi-supervised (e.g., *Kylin* [73]) system or unsupervised (e.g., *TextRunner* [3])

In the following sections we provide more details about each of the strategies.

### 3.2.1 Supervised systems

Supervised information extraction systems, also known as closed or traditional information extraction system [4], rely on labeled training sets and handcrafted extraction patterns to produce high quality extraction from text. However, because it is not possible to offer labeling to all instances and define patterns to extract all the possible representations of a relationship, supervised systems tend to have a limited coverage of possible extractions, and do not always perform well on new data. Because of this limitation, supervised systems are mostly used for domain specific extraction, such as OBIE [28, 30, 52]. Based on the type of information extraction, there are two main strategies followed by supervised systems [77]: *extraction rules* and *machine learning*.

Extraction rules capture information by identifying specific syntactic and lexical elements in text, such as keywords [52], part-of-speech labels or other semantic/syntactic structures. In most cases, extraction rules are simple to design, and because they are handcrafted, extraction rules can be very accurate. Although they can be define following regular expression, languages like *SystemT's* Annotation Query Language (AQL) [42] and GATE's Java Annotation Patterns Engine (JAPE) [11] have been created to specify extraction patterns. These specially design languages allow the creation of complex extraction rules through the manipulation of annotations. AQL includes a series of optimizations that can reduce significantly the execution time of an extraction when compared to regular expressions based extraction rules [42], while JAPE can directly execute Java code from the matching of a pattern [11].

On the other hand, machine learning methods such as classification methods and probabilistic models try to identify which elements from a sentence are part of the sought after information. However, machine learning techniques are data-driven, so the performance of these methods depend on the quality and quantity of the data used for the training. For machine learning based extraction systems, this tight relation comes from the fact the training data used by the classifier or sequence model has been labeled by an expert. In the case of extraction rules, the rules are created and tuned by hand, based on data and knowledge of the domain.

The Ontology-based Components for Information Extraction (OBCIE) architecture offers a *two-phase* machine learning extraction approach [76]. This approach determines in the first phase which sentences of a text might contain extractable information. Since this phase is handled by a classifier, sentences are transformed into binary vectors that have features as keywords. In other words, if a sentence has the first keyword but not the second, then the vector representation of the sentence will have 1 for the first keyword and 0 for the second keyword. In the second phase, this approach determines if the sentences actually has the sought information. This is done by sequence model that uses an enhanced sentence, which has the labels of a set of lexical and syntactic features. If we also include the output of the sentences classifier as part of the input of the sequence model, it is possible to obtain extractions from sentences that have been incorrectly been classified in the initial phase [29].

### 3.2.2 Semi-supervised systems

Semi-supervised systems use the connection between sentences and hand built knowledge bases to learn information extraction models and patterns for specific relationships. In contrast with supervised systems which have an explicit link between text and instances, semi-supervised systems have to discover this connection. In some cases the connection is clear, for example *Kylin* that exploits the relation between Wikipedias infoboxes and articles [78, 79, 80]. On other cases, where the corpus of text is the Web [2, 10, 50, 65], or text in other language [6], the connection is not that evident.

Although each system follows a different approach on how to determine the connection between the knowledge base and the text, semi-supervised systems work in a specific form, following three main steps: instances from knowledge base are looked up in sentences of the text, selected sentences are transformed into sets of relevant elements, patterns or models are learned based on the enhanced sentences.

The first step performed by a semi-supervised system is to identify sentences that might represent the instances or tuples from the knowledge base. In the case of *Snowball* [2], *Distant Supervision* (DS) [50], and the system by Snow et al. [65], if a sentence contains a pair of entities that have a relationship in the knowledge base, the sentence most likely represent the relationship. Even more, if there is a group of sentences that have the same pair of entities, then it is very likely that they represent the same relationship. This is not strictly true since it is possible for a pair of entities to have a sentence that represent different relationships [59].

On the other hand, *Kylin* [79] determines the sentences where the instance are mention by following a two-phase classification approach. The first classifier determines if a given document contains the instances sought. If the sentence does contain the relational instance, then it passes by a sentence classifier that determines which sentence of the document might have the instance. In the case of *Kylin*, the sentence selection process can provide higher quality examples because it uses Wikipedia articles with their infoboxes. Wikipedias infoboxes provide a tabular summary of attributes from an article. In other word, the most relevant information of an article will appear in both the text of the article and in the infobox of the article.

The second step performed by a semi-supervised system is to determine what elements of the sentence are important for the extraction process. In general, the elements from the sentence are generalized to reduce it from its written form into a set of features that are shared between sentences. Most systems use as lexical features (specific words from the sentences), and syntactic features (part-of-speech, dependency parsing). In some cases, semantic information (named entity) might be included as feature [2, 50, 65, 79]. In DS considers that although a selected sentences has the relations entities, it is quite possible that the sentences also have *noise*. To learn a robust classifier that can manage this noise, each sentence is transform into a large set of feature. These feature are lexical and syntactical, and model the words before, after, and in between entities. In this step sentences are also transformed into a representation that can facilitate the next task. In DS and in the system by Snow et al. sentences are transformed into vectors by encoding the features of the sentences, while *Kylin* enhance the text with lexical and syntactical labels. In the case of *Snowball*, the sentence is transformed into a combination of labeled terms and weighted terms from the sentence.

The third step is learn from the example sentence. In the case of Snowball, this task is mostly reduced to evaluate the set of extraction patterns to determine the best set of extractors for the example sentences. The evaluation is done by determining a matching score between a pattern and the set of examples sentences. For *Kylin* and DS, this task consist on applying a machine learning technique. *Kylin* uses Conditional Random Fields to learn a sequence model from the sentence. For the Conditional Random Fields, *Kylin* uses a set of features such as the actual words from the sentence, part-of-speech, if the word is a the first or second half of the sentence, as also the output of the sentence and document classifiers. In the case of Distant Supervision, the system uses multi-class logistic classifier. The output is a relation name and a confidence score.

Some systems integrate a fourth step that intends to use the underlying ontology or representation structure to improve the quality of the extraction process. *Kylin Ontology Generator* [80] improves the quality of Wikipedias infobox ontology by refining the relation between classes and attributes. This leads to propagate properties and instance through infoboxes, following the relation between their concepts. In a similar form, Carlson et al. [10] approach also performs a sharing of instances depending on the logical relation between concepts. This structure-based refinement is extended by filtering instances that are mutually excluded (instances of disjoint concepts), or that have an erroneous type.

After a model or pattern of extraction is learned, new instances can be extracted from text [81]. These new instances can lead to a new learning process, that can produce higher quality extractors [78].

### 3.2.3 Unsupervised systems

Unsupervised systems perform information extraction without requiring any labeling or specific pattern construction. They perform extraction based on linguistic features that are embedded in the text. By evaluating the quality of the relationships extracted, unsupervised systems can learn more robust patterns a models that provide a higher coverage of the extractions that the system can perform.

Core to all unsupervised systems are *Hearst extraction patterns* [34]. Hearst has identified a small set of specific linguistic structures (combination of lexical and syntactical elements) that represent an *hyponymy* relationship between two or more entities. For example, if the pattern,

$$NP_0 \text{ such as } [NP_1,\ NP_2,\ ...,\ (\text{and}|\text{or})\ NP_n]$$

is applied to the sentence "Motor vehicles such as automobiles, and motorcycles..." leads to the extraction of the relations *hypony(automobiles,motor vehicles)* and *hyponym(motorcycles, motor vehicles).* A hyponymy relations between two entities $L_0$ and $L_1$ refers to membership relations in the form $L_0$ is a (kind of) $L_1$. In this case, hyponymy is roughly equivalent to the ontological relation between a concept and its super concept.

In order to extract different type of relationships, Hearst original set of extraction patterns have been extended to consider other patterns. New patterns, such as $NP_0$ *Verb* $NP_1$, have allowed systems like *KnowItAll* [18, 19] and *TextRunner* [3, 4] to the extraction of a wide variety of relationship instances. In their case study, Banko et al. [3] found that this extended set of extraction rules can cover up to 95% of all binary relationship from their text corpus. However, because *TextRunner* combines these extraction rules with either Naive Bayes [3] or Conditional Random Fields [4], it can produce *incoherent* and *uninformative* extractions. Incoherent extractions are produced when the sequence of decision lead to an incorrect extractor. Uninformative extraction occur when relevant information is removed from the relational phrase because it is incorrectly handle.

In order to reduce these erroneous extractions, *ReVerb* [20] propose a refinement in the extraction patterns by better defining the syntactical structure that represents the relationship:

$$Verb((Noun|Adj|Adv|Pron|Det) * (Prep|Particle|Inf.marker))$$

Because this constraint can deal with phrases that have multiple verbs, it can mostly eliminate incoherent extractions and reduce uninformative extractions.

Some systems include confidence value as a mechanism to support and validate the extraction process. *KnowItAll* [18] measure the quality of an extraction pattern based on redundancy of instances being extracted together [16]. It queries a search engine with the output of the extraction, and based on the number of documents retrieved by the query, a probability of correctness is estimated. *TextRunner* [4] also uses the redundancy estimation of *KnowItAll*, but the probability is estimated over the set of normalized (i.e., lemmatized) extractions. On the other hand, *ReVerb* [20] learns a logistic regression classifier to estimate the confidence of an extraction. The logistic regression classifier uses a set of syntactic and lexical feature from the sentences. *ReVerb* also removes infrequent relations (less than 20 instances) to avoid over specification.

Although the set of general extraction rules should allow the extraction of most type of relationships (from 85% [20] to 95% [4] of all binary relationships), it is possible to extend it by learning new extraction rules or robust extraction models and patterns. From the initial extraction, it is possible to extend the extraction strategy following an approach similar to semi-supervised system. The initial set of extracted relations are used to learn new extraction patterns [34, 49] or a extraction model [3, 4]. In the case of *Ollie* [49], the new extraction patterns are actually templates. From a set of high confidence relations extracted by *ReVerb*, *Ollie* analyze the dependency of the extractions to learn more general patterns. By including dependency parsing, *Ollie* can manage complex relationships, defined by verb phrase structure, between complex entities. This leads to higher coverage of the extraction patterns without loosing accuracy.

It must be noted that in general, the unsupervised systems strength is in the coverage of the extraction rules without the need of a labeled training set. However, they tend to have a low accuracy when compared with supervised or semi-supervised systems. If the application requires high coverage over accuracy, then the best approach is to consider an unsupervised system for the extraction process.

## 4 Analysis of Incorrect Statements

In the following section, we propose a set of strategies to perform Ontology-based Error Detection based on the research reviewed in Section 2 and Section 4.

To understand what an *incorrect statement* is an how to identify it, we need to define what a *statement* is and what makes it incorrect. We define a statement as a proposition expressed in written natural language that has an equivalent logical representation. The connection between text and logic can easily be seen at the sentences level. The typological elements of a sentence (*subject-verb-object* structure) can be mapped to the triple form by a relationship between two concepts. For example, the sentence “Herbivores eat plants” is equivalent to the axiom $eat(Herbivores, plants)$.

We consider that the *correctness* of a statement is determined by the domain. Under the intuition that a text from a domain is logically entailed by the domain, and based on the definition of textual entailment by Dagan and Glickman [12], we consider a statement to be *correct* if it can be inferred from the domain ontology under the interpretation given by the domain ontology. In other words, a *correct statement* from a text is the logic consequence of the concepts and properties that define the domain, which makes the statement also consistent with respect to the ontology. It is important to note that for a statement to be correct, it must be entailed from the domain and not only to be consistent with it. This difference comes from the fact that a statement does not require to be part of the domain to be consistent with it. For example, let us consider the domain of astronomy and a statement regarding gene properties. The statement most likely will not contradict the domain; however, it cannot be considered strictly correct for an astronomy text since it is not entailed from the domain. Statements that are consistent but not entailed from the domain shall be called *incomplete statements*. These incomplete statements are neither correct nor incorrect since it is not possible to determine their truth value from the domain ontology (open world assumption [57]).

As a consequence of the above definition of correct statement, it seems natural to consider that a statement is *incorrect* if it logically contradicts some aspects of the domain knowledge. As defined by Haase and Stojanovik [32], a logical contradiction or inconsistency in an ontology is produced by an axiom (or a statement) that violates a constraint. So, if a statement (e.g., "Planet Jupiter is a star") breaches a constraint of the domain (the concepts *Planet* and *Star* cannot share any sub concepts or instances), then the statement is both inconsistent and incorrect with respect to the ontology.

As previously stated, we propose the use of a combination of Information Extraction (and OBIE) and Ontology Debugging to identify incorrect statements in texts. Information Extraction will allow us to identify domain statements (or relationships) from the text while Ontology Debugging tools will provide us with a mechanism to determine the correctness of the statements. With these two research areas in mind, we have considered two main approaches to identify incorrect statements: *precomputing incorrectness*, and *online incorrectness inference*. In the approach of *precomputing incorrectness*, we upfront determine what would an inconsistent axiom be. Then, based on this inconsistent axiom, we define an information extractor to identify the incorrect sentence. On the other hand, in the *online incorrectness inference*, we extract every possible statement from the text in a generic way, so we can transform the sentences into logical clauses or axioms. These axioms are then added to the ontology where the consistency of the new extended ontology (domain ontology plus axiom from the text) is tested. In the following section, we give a more detailed discussion of these two proposed approaches.

## 4.1 Precomputing Incorrectness

This approach can be seen as Ontology-based Information Extraction (OBIE) with a small twist. Following the definition of Wimalasuriya and Dou [77], OBIE uses an ontology to guide the information extraction process. In the precomputed incorrectness approach, the domain ontology also guides the extraction process, but indirectly. The ontology, through a debugging mechanism, produces a set of axioms which are inconsistent to each other. The information extraction process is then guided by these inconsistent axioms.

As mentioned, the precomputed incorrectness works in two steps: determining inconsistent axioms, and extracting statements based on the incorrect axioms. In the following sections we provide more details into each step.

### 4.1.1 Determining Incorrect Axioms

As mentioned, before determining which statements are incorrect, we need to know how a statement can be incorrect with respect to the domain. Considering that an ontology only has correct domain information, we need a mechanism or strategy to infer incorrectness from correct information. In previous work [28, 29], we used Wang et al. heuristic approach [72] that looks for inconsistency based on the violation of specific ontological constraints. Let us consider the following example:

| | |
|---|---|
| Ontology | $Student \sqsubseteq Person$<br>$Professor \sqsubseteq Person$<br>$Student \sqsubseteq \neg Professor$<br>$GradStudent \sqsubseteq Student$<br>$UndergradStudent \sqsubseteq Student$ |
| Inconsistent Axiom | $GradStudent \sqsubseteq Professor$ |

In the example, because of the disjointness between the concepts *Professor* and *Student*, they cannot share subclasses or individuals. By creating membership between disjoint concepts, we are creating an inconsistent axiom. Although the example seems to be a very simple case of inconsistency, statements that represent these types of inconsistent axioms are not so unlikely [28].

An interesting aspect of using Wang et al.'s approach is that the inconsistent axioms can be generated from the ontology before any text is analyzed. This advantage provides great flexibility for the implementation of the information extraction process, such as distributed or parallel analysis of text. However, defining upfront inconsistent axioms has two shortcomes: the incompleteness of heuristic approach, and the exponential size of the set of inconsistent axioms. The first problem refers to the incompleteness of the solution with the method proposed by Wang et al. It is possible that some type of inconsistency cannot be generated by the method, which leads to not having an extractor that can identify a statement with that type of error. Yet, it is probable that for any given domain, there are types of incorrect statements that do not appear in any text. This can occur when the text being analyzed focus on a subset of the domain, or when some elements of the domain, although mentioned in the ontology, are not frequently discussed (e.g., highly abstract concept). In other words, although completeness is needed for detecting incorrectness, it might not be a strict requirement for our approach in contrast to domain-awareness [67].

The second problem refers to large set of inconsistent axioms that can be generated from a small set of consistent axioms. Suppose that our heuristic method indicates that inconsistent axioms can occur by the erroneous statement $teaches(X, Student)$, where $X$ can be any concept from our domain ontology except $Professor$. If $X$ is *Professor*, then we would obtained the correct axiom. Now, if our ontology has $n$ concepts, and each relationship can have the same structure as the example, then the size of the set of inconsistent axioms for each relationship is $n-1$. Considering that an ontology can have many relationships, and this type of incorrect axioms can also be built from the hierarchical relations between concepts, the amount of inconsistent axioms that can be predefined from an ontology can grow very fast with respect to the size of the ontology. This problem has also a linguistic dimension since an axiom can have many different representations in written natural language.

### 4.1.2 Information Extraction

After determining the incorrect axioms, we encode them into information extractors, which are the actual components that do the extraction. As mentioned in Section 3, information extractors can be based on linguistic patterns and regular expressions (i.e., extraction rules), or they can be based in some type of machine learning method [77]. In most cases, extraction rules are relatively accurate, but they are difficult to generalize and do not scale well [52, 82]. On the other hand, machine learning techniques are data-driven, so the performance of these methods depend on the quality and size of the data set used for the training.

When evaluating extractions of incorrect statements, we have found that rule based extractions have a higher precision (more correct extraction) while machine learning based extractions have a higher recall (more complete extraction) [29]. This behavior can be explained by considering how the extractors are built. The information extractors can be built by considering specific patterns or general patterns. In the case of rule based extractors, the patterns used for extraction are built by considering very specific lexical and syntactic cues from the text. It is very possible to have a set of rules to extract one concept or relationship. With patterns built to extract specific instances, it is more difficult that an error can occur when detecting an instance to be extracted. However, the specificity of rule based extraction is also its weakness. If there is an instance that is not similar to any of the known instances that are used to construct the patterns, it is most likely that the instance will not be extracted by any pattern. This weakness leads to a less complete extraction process because extractors overlook instances that should be extracted. In the case of machine learning based information extractors, a model is learned which should fit the training data in a fashion that can guarantee some flexibility to manage unseen instances. This flexibility produces a more complete extraction process since the extractor can consider instances that have not been seen yet. And in a similar way as rule based extraction, this flexibility can also be the weakness of the machine learning based extraction. Since the model is more general, it is possible that unrelated elements can be extracted, leading to a lower precision.

In our most recent work, we intend to maximize the overall performance while taking into account the relation between the type of implementation of the extractor and the quality of extraction [29] by extending the Ontology-based Components for Information Extraction (OBCIE) architecture [76]. OBCIE promotes reusability by defining components to be as modular as possible. Taking advantage of OBCIE's modularity, we propose an hybrid OBIE system that integrates information extractors of different natures into one platform. In other words, we have combined rule based extraction with a machine learning based extraction into a single system. In our hybrid approach, we can create different configuration of extractors, where each concept can have a rule based extractor or machine learning based extractor. We found that the overall performance of the system (F1 measure) with an hybrid setting was higher than that with only one type of extractor [29]. From these results, we believe that this hybrid approach can provide an improvement in the performance for information extraction.

Since our proposed hybrid extraction system is mainly a proof of concept, there are at least two extension of this work that have not been analyzed yet. The first extension is to determine a mechanism that can select the optimal configuration of extractors. This is a natural extension of our most recent work. Because there are many possible hybrid configuration, an automatic selection of configuration based on performance can seen as an optimization problem, where we intend to optimize the performance of the extraction process. The second extension is to determine if an aggregation of outputs can improve the performance of the extraction system with respect to all three metrics (precision, recall, F1 measure). The idea of aggregation of outputs of different extractors comes from the results obtained by Wimalasuriya and Dou's multiple ontology information extraction approach (MOBIE) [75]. In the case of MOBIE, mapping between elements of different ontologies can provide guidelines to combine the different information extractors. In the case of aggregation outputs of different implementation of extractors, there is no need for any extra information since both type of extractors represent the same ontological element.

## 4.2 Online Incorrectness Inference

In *online incorrectness inference*, the OBIE process is performed in an different fashion. Instead of having the ontology guiding the extraction process, the IE is performed based on structural elements from the text, while the ontology is used to validate the correctness of the statements extracted. Although this approach differs from the definition of OBIE [77], we argue that it is still an OBIE process since the approach relies on the domain ontology to determine the correctness of each statement.

The main goal of this approach is to provide the most complete analysis of the correctness of a text. In that sense, the extraction process intends to extract every possible relationship in the text, and the inconsistency analysis use the complete ontology to infer the correctness or incorrectness of the text. The online incorrectness inference approach consists of three steps. In the first step, statements are extracted from the text following an approach similar to open information extraction. The extraction process provides a transformation of the statement from its original textual form into a logical bound form. As a second step, the statements are added to the domain ontology so their correctness can be determined, and justified if needed. The third and final step is to determine the correctness of the statement. In case the statement is inconsistent with respect to the ontology, we determine why it is inconsistent.

### 4.2.1 Extraction and Validation of Statements

In order to do IE without guidance of a domain ontology, we can follow strategies such as unsupervised or semi-supervised information extraction.

In contrast to the extraction stage of the *precomputing incorrectness* approach, which focus only on sentences from the text that are directly related to the domain, the extraction stage in *online incorrectness inference* following an unsupervised strategy will intend to identify all relationships that appear in a text. Because of the large amount of possible relationships that could be extracted from a document, when adding the extracted statements into the ontology, we will need a *validation stage* to guarantee that the extracted statements are related to the ontology. In the case of validating relationships at the terminology level of the domain (*TBox*), it seems feasible to accomplish it with simple mechanisms, such as *gazetteers* (i.e., list of words) of ontological terms. In the case of validating extraction from the assertion level of the domain (*ABox*), the solutions might not be so trivial. Let us consider the following example:

| | |
|---|---|
| *Extraction 1*: | *Student(Fernando)* |
| *Extraction 2*: | *University(University of Oregon)* |
| *Extraction 3*: | *studies(Fernando, University of Oregon)* |

We want to include in our analysis only the extracted relations that address the domain, which in this case is about universities. From the example, it seems safe to assume that *Extraction 1* and *Extraction 2* are referring to elements of the domain. Although *Extraction 3* seems to be also part of the domain, we can only consider its membership when taking into account *Extraction 1* and *Extraction 2*. In the case of our example, if we analyze the elements that are involved in the relation (e.g., string matching), we might still be able to determine that the extracted instance is part of our domain; however, this is not guaranteed.

The filtering of statements in the *validation stage* intends to reduce the complexity of the analysis of the statements and provides a clearer interpretation of the correctness of the text. As we will see in the following section, the analysis of statements that belong to the domain is already a task of high complexity. By not including unnecessary statements, we are reducing the overall complexity of the task. On the other hand, if we consider the previously defined incomplete statement, these statements will most likely be consistent with the domain ontology, but they are not entailed by it. So, the analysis of these out of domain statements are not useful for the task of identifying incorrect statement.

In the case of semi-supervised extraction, the need to provide a validation of the extraction process is mostly eliminated if we can identify the text that is directly linked to the instances that are used in the extraction process. If we cannot provide these highly related texts, we will need to consider procedures similar to the ones used for unsupervised extraction. In those cases where we have assertional elements in the ontology that are connected to a specific body of documents, we have a new question to answer: how many instances are needed to perform a semi-supervised extraction? If we consider the results obtained by Kylin [80], this number cannot be small. Yet, in the case of Kylin, because of normalization performed to the extraction, the connection between the instances and the text is reduced. We need to restate the question: how does the connection between text and knowledge base affects the minimum set of instances needed to perform a semi-supervised extraction?

### 4.2.2 Analysis of Extractions

Once we have extracted all the relations from the text and have filtered out those that are not part of the domain, we proceed to analyze the correctness of the statements. Because we have moved the text into the ontology, we have new options when analyzing the text. We have considered two main scenarios under which we can analyze the correctness of statements:

1. **Single statement**: each statement is analyzed independently from the rest of the statements from the text.

2. **Multiple statements**: a group of statements is analyzed simultaneously. Since they are considered as a group, it is possible that inconsistency among the grouped statement might arise. Although the solutions under this scenario should be independent of the size of the set, since inference is the main tool used by ontology debugging methods, it is very likely that large set of statements (statements from a corpus) might require different strategies than the ones used for smaller sets.

The analysis of single statement consist of determining the correctness (or incorrectness) of one statement of a text at a time. This analysis can be done by creating a *test ontology* which consists of the domain ontology plus the statement we want to analyze. We evaluate the consistency of the *test ontology* to determine if the statement is correct or not. If the *test ontology* is inconsistent, then we analyze this test ontology to determine the origin of the inconsistency. In general, the Ontology Debugging methods reviewed in the Section 2.5 can be applied here. We will focus on the logic-based approaches since we are looking for a sound and complete solution for correctness analysis.

As seen in the review, logic based ontology debugging methods are formed by a series of steps, which have some level of similarity between methods. It seems possible that because of this similarity, they can be combined to provide a more optimal solution to our problem. For example, we can use a *selection function* to determine inconsistent subsets of the ontology [62], and it can be combined with HST methods to determine the minimal set of inconsistent elements of the ontology [58, 35]. This minimal subset can be analyzed by the generalized incoherence-preserving terminology [61] to determine the actual parts of the axiom that are involved in the inconsistency. In our case, because we know the element that is causing the inconsistency, we can simplify the initial search to determine the subset of the ontology that is inconsistent. For example, the statement being analyzed can be the seed for the selection function.

Considering that in practice we will be faced with the task of analyzing the correctness of a complete document and not single sentences, it seems reasonable to examine how correctness analysis will perform when looking into sets of statements. A simple and straightforward approach is to *sequentially* (or in parallel) apply the solution for single statements to each statement of the document. However, if statements are considered independently when analyzed, it is possible that we might overlook some incorrectness, as the following example illustrates:

$$\begin{array}{rl} & Student \sqsubseteq Person \\ & Professor \sqsubseteq Person \\ \text{Ontology} & Student \sqsubseteq \neg Professor \\ & teaches(Professor, Student) \\ & Person(Fernando) \\ & \\ \textit{Statement 1} & Student(Fernando) \\ \textit{Statement 2} & teaches(Fernando, Somebody) \end{array}$$

If we analyze the two statement separately, we would find that *Statement 1* is consistent since all individuals that are member of the class *Student* are also member of class $Person$. In the same way, *Statement 2* is consistent with the ontology, and it actually allows us to infer that $Professor(Fernando)$. The inconsistency of the statements rise only when both of them are considered simultaneously for the analysis. *Statement 1* is incorrect if *Statement 2* is correct, or *Statement 2* is incorrect if *Statement 1* is correct.

An alternative approach is to consider the set of statements as a whole and do the analysis as group rather than individually. As in the last example, by analyzing the correctness of set of statements, it is possible to identify text incorrectness that only rises in the presence of more than one statement from the text. However, although this approach would definitely provide a more complete solution, it clearly leads to an increase in the complexity of the task of finding incorrectness. An option is to incrementally analyze the set of statements. Iteratively, we add statements into the ontology and perform consistency checks. If there is an inconsistency, we try to identify the origin. This incremental approach allows us to keep some control over the complexity of the process while still providing completeness over the analysis. In this approach, a key element is the order of the statements that are being added into the ontology. For example, from a text we produce the set of statements $S = s_1, ..., s_i, ..., s_j, ..., s_n$ (with $i$ much smaller than $j$). Let us assume that the inclusion into the ontology of statements $s_i$ and $s_j$ together makes it inconsistent. Then, since $i$ is much smaller than $j$, in our incremental approach $s_j$ will be added many iteration after $s_i$. If we sort the statements with a selection function, the analysis with both statements can be performed earlier. Although this efficient ordering of statements does not reduce the complexity of the consistency check, it can reduce the complexity when trying to find the origin of the inconsistency.

## 5 Conclusion

In the present work, we present a review of relevant research for the new area of Ontology-based Error Detection. We have first considered research on Ontology Inconsistency, with focus on Ontology Debugging. Afterwards, we have included research on Information Extraction in which we have classified current research based on the amount of intervention required for their deployment.

We propose two approaches to perform Ontology-based Error Detection. In the first approach, we precompute possible inconsistencies that can arise from the ontology when considering certain type of violation of constraints. Based on the set of possible inconsistencies, we can develop information extractors to analyze the text. In the second approach, we perform Information Extraction on the text to identify all stated relationships. From the set of extracted relationships, we determine the subset that is in the domain of the ontology. The domain related relationships are added into the ontology, so we can perform inference to determine the correctness of the text. In case of inconsistency, we perform Ontology Debugging to identify the elements that are causing the inconsistency.

From the results of our initial work, we have determined the feasibility of detecting domain related errors in text. From the results of this work, plus insight gained when analyzing the factibility of error detection based on current research, we have determined a series of issues that can be focus of new research. The resolution of some of these issues, such as hybrid information extractors, can lead into improvements beyond error detection.

## References

[1] World Wide Web Consortium.
`http://www.w3.org/`.

[2] E. Agichtein and L. Gravano. Snowball: Extracting relations from large plain-text collections. In *In Proceedings of the 5th ACM International Conference on Digital Libraries*, pages 85–94, 2000.

[3] M. Banko, M. J. Cafarella, S. Soderland, M. Broadhead, and O. Etzioni. Open information extraction from the web. In *Proceedings of the 20th international joint conference on Artifical intelligence*, IJCAI'07, pages 2670–2676, San Francisco, CA, USA, 2007. Morgan Kaufmann Publishers Inc.

[4] M. Banko and O. Etzioni. The tradeoffs between open and traditional relation extraction. In *Proceedings of ACL-08: HLT*, pages 28–36, Columbus, Ohio, June 2008. Association for Computational Linguistics.

[5] M. Benaissa and Y. Lebbah. A constraint programming based approach to detect ontology inconsistencies. *International Arab Journal of Information Technology*, 8(1):1–8, 2011.

[6] A. Blessing and H. Schütze. Crosslingual distant supervision for extracting relations of different complexity. In *Proceedings of the 21st ACM international conference on Information and knowledge management*, CIKM '12, pages 1123–1132, New York, NY, USA, 2012. ACM.

[7] E. Brent, C. Atkisson, and N. Green. *Monitoring and Assessment in Online Collaborative Environments: Emergent Computational Technologies for E-Learning Support*, chapter Time-Shifted Online Collaboration: Creating Teachable Moments Through Automated Grading, pages 55–73. IGI Global, 2010.

[8] D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. Dl-lite: Tractable description logics for ontologies. In *20th National Conference on Artificial Intelligence (AAAI 2005)*, pages 602–607, 2005.

[9] D. Calvanese, G. De Giacomo, M. Lenzerini, R. Rosati, and G. Vetere. Dl-lite: Practical reasoning for rich dls. In *Proc. of the 17th Int. Workshop on Description Logics (DL 2004)*, volume 104 of *CEUR Electronic Workshop Proceedings, http://ceur-ws.org/*, 2004.

[10] A. Carlson, J. Betteridge, E. R. Hruschka, and T. M. Mitchell. Coupling semi-supervised learning of categories and relations. In *SemiSupLearn '09: Proceedings of the NAACL HLT 2009 Workshop on Semi-Supervised Learning for Natural Language Processing*, pages 1–9, Morristown, NJ, USA, 2009. Association for Computational Linguistics.

[11] H. Cunningham, D. Maynard, K. Bontcheva, V. Tablan, N. Aswani, I. Roberts, G. Gorrell, A. Funk, A. Roberts, D. Damljanovic, T. Heitz, M. A. Greenwood, H. Saggion, J. Petrak, Y. Li, and W. Peters. *Text Processing with GATE (Version 6)*. 2011.

[12] I. Dagan and O. Glickman. Probabilistic Textual Entailment: Generic Applied Modeling of Language Variability. In *Learning Methods for Text Understanding and Mining*, Jan. 2004.

[13] M. Dalal. Investigations into a theory of knowledge base revision. In *AAAI*, pages 475–479, 1988.

[14] M.-C. de Marneffe, A. N. Rafferty, and C. D. Manning. Finding contradictions in text. In K. McKeown, J. D. Moore, S. Teufel, J. Allan, and S. Furui, editors, *ACL*, pages 1039–1047. The Association for Computer Linguistics, 2008.

[15] X. Deng, V. Haarslev, and N. Shiri. Measuring inconsistencies in ontologies. In *Proceedings of the 4th European conference on The Semantic Web: Research and Applications*, ESWC '07, pages 326–340, Berlin, Heidelberg, 2007. Springer-Verlag.

[16] D. Downey, O. Etzioni, and S. Soderland. A probabilistic model of redundancy in information extraction. In *Proceedings of the 19th international joint conference on Artificial intelligence*, IJCAI'05, pages 1034–1041, San Francisco, CA, USA, 2005. Morgan Kaufmann Publishers Inc.

[17] O. Etzioni, M. Banko, S. Soderland, and D. S. Weld. Open information extraction from the web. *Commun. ACM*, 51(12):68–74, Dec. 2008.

[18] O. Etzioni, M. Cafarella, D. Downey, S. Kok, A.-M. Popescu, T. Shaked, S. Soderland, D. S. Weld, and A. Yates. Web-scale information extraction in knowitall: (preliminary results). In *Proceedings of the 13th international conference on World Wide Web*, WWW '04, pages 100–110, New York, NY, USA, 2004. ACM.

[19] O. Etzioni, M. Cafarella, D. Downey, A.-M. Popescu, T. Shaked, S. Soderland, D. S. Weld, and A. Yates. Unsupervised named-entity extraction from the web: an experimental study. *Artif. Intell.*, 165(1):91–134, June 2005.

[20] A. Fader, S. Soderland, and O. Etzioni. Identifying relations for open information extraction. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '11, pages 1535–1545, Stroudsburg, PA, USA, 2011. Association for Computational Linguistics.

[21] G. Flouris, Z. Huang, J. Z. Pan, D. Plexousakis, and H. Wache. Inconsistencies, negations and changes in ontologies. In *Proceeding of the 21st National Conference on Artificial Intelligence*, page 6. AAAI, 2006.

[22] G. Flouris, D. Manakanatas, H. Kondylakis, D. Plexousakis, and G. Antoniou. Ontology change: Classification and survey. *The Knowledge Engineering Review*, 23(02):117–152, 2008.

[23] G. Flouris, D. Plexousakis, and G. Antoniou. On applying the agm theory to dls and owl. In Y. Gil, E. Motta, V. Benjamins, and M. Musen, editors, *The International Semantic Web Conference (ISWC)*, volume 3729 of *Lecture Notes in Computer Science*, pages 216–231. Springer Berlin / Heidelberg, 2005.

[24] P. W. Foltz, D. Laham, and T. K. Landauer. Automated essay scoring: Applications to educational technology. In B. Collis and R. Oliver, editors, *Proceedings of World Conference on Educational Multimedia, Hypermedia and Telecommunications 1999*, pages 939–944, Chesapeake, VA, 1999. AACE.

[25] M. Franzke and L. Streeter. Building student summarization, writing and reading comprehension skills with guided practice and automated feedback. White paper from Pearson Knowledge Technologies, 2006.

[26] P. Gardenfors. *Belief revision*, chapter Belief revision: An introduction, pages 1–20. Cambridge University Press, 1992.

[27] T. R. Gruber. A translation approach to portable ontology specifications. *Knowl. Acquis.*, 5(2):199–220, June 1993.

[28] F. Gutierrez, D. Dou, S. Fickas, and G. Griffiths. Providing Grades and Feedback for Student Summaries by Ontology-based Information Extraction. In *Proceedings of the 21st ACM Conference on Information and Knowledge Management (CIKM)*, pages 1722–1726, 2012.

[29] F. Gutierrez, D. Dou, S. Fickas, A. Martini, and H. Zong. Hybrid Ontology-based Information Extraction for Automated Text Grading. In *Proceedings of the 12th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages ???–???, 2013.

[30] F. Gutierrez, D. C. Wimalasuriya, and D. Dou. Using Information Extractors with the Neural ElectroMagnetic Ontologies. In *Proceedings of the International Conference on Ontologies, Databases and Application of Semantics (ODBASE) (poster paper)*, pages LNCS 7046, 31–32, 2011.

[31] V. Haarslev and R. Müller. Racer system description. In R. Goré, A. Leitsch, and T. Nipkow, editors, *Automated Reasoning*, volume 2083 of *Lecture Notes in Computer Science*, pages 701–705. Springer Berlin / Heidelberg, 2001. 10.1007/3-540-45744-5_59.

[32] P. Haase and L. Stojanovic. Consistent evolution of owl ontologies. In *Proceedings of the 2nd European Semantic Web Conference (ESWC), LNCS 3532*, pages 182–197. Springer, 2005.

[33] P. Haase and J. Vlker. Ontology learning and reasoning – dealing with uncertainty and inconsistency. In P. C. Costa, C. D'Amato, N. Fanizzi, K. B. Laskey, K. J. Laskey, T. Lukasiewicz, M. Nickles, and M. Pool, editors, *Uncertainty Reasoning for the Semantic Web I*, pages 366–384. Springer-Verlag, Berlin, Heidelberg, 2008.

[34] M. A. Hearst. Automatic Acquisition of Hyponyms from Large Text Corpora. In *Proc. of the Fourteenth Conference on Computational Linguistics, Nantes, France*, pages 539–545, Morristown, NJ, USA, July 23-28 1992. Association for Computational Linguistics.

[35] M. Horridge, B. Parsia, and U. Sattler. Explaining inconsistencies in owl ontologies. In *SUM*, pages 124–137, 2009.

[36] I. Horrocks. Semantics $\sqcap$ scalability $\models \bot$? *Journal of Zhejiang University - Science C*, 13(4):241–244, 2012.

[37] Z. Huang, F. van Harmelen, and A. ten Teije. Reasoning with inconsistent ontologies. In *Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence (IJCAI'05)*, pages 454–459, Edinburgh, Scotland, August 2005.

[38] D. Jurafsky and J. H. Martin. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics and Speech Recognition*. Prentice Hall, second edition, Feb. 2008.

[39] A. Kalyanpur, B. Parsia, M. Horridge, and E. Sirin. Finding all justifications of owl dl entailments. In *Proceedings of the 6th international The semantic web and 2nd Asian conference on Asian semantic web conference*, ISWC'07/ASWC'07, pages 267–280, Berlin, Heidelberg, 2007. Springer-Verlag.

[40] J. D. Karpicke and H. L. Roediger. The Critical Importance of Retrieval for Learning. *Science*, 319(5865):966–968, Feb. 2008.

[41] H. Katsuno and A. O. Mendelzon. On the difference between updating a knowledge base and revising it. In *KR*, pages 387–394, 1991.

[42] R. Krishnamurthy, Y. Li, S. Raghavan, F. Reiss, S. Vaithyanathan, and H. Zhu. Systemt: a system for declarative information extraction. *SIGMOD Record*, 37(4):7–13, 2008.

[43] D. Lembo, M. Lenzerini, R. Rosati, M. Ruzzi, and D. F. Savo. Inconsistency-tolerant semantics for description logics. In *Proceedings of the Fourth International Conference on Web Reasoning and Rule Systems*, RR'10, pages 103–117, Berlin, Heidelberg, 2010. Springer-Verlag.

[44] C.-Y. Lin. Rouge: a package for automatic evaluation of summaries. In *Workshop On Text Summarization Branches Out*, pages 25–26, 2004.

[45] B. Liu, J. Li, and Y. Zhao. A query-specific reasoning method for inconsistent and uncertain ontology. In *Proceedings of the International MultiConference fo Engineers and Computer Scientists 2011*, volume 1, Hong Kong, March 2011.

[46] R. Lyons. The Spread of Evidence-Poor Medicine via Flawed Social-Network Analysis. *Statistics, Politics, and Policy*, 2, May 2011.

[47] A. Maedche and S. Staab. Ontology learning for the semantic web. *IEEE Intelligent Systems*, 16(2):72–79, Mar. 2001.

[48] G. Masotti, R. Rosati, and M. Ruzzi. Practical abox cleaning in dl-lite (progress report). In *Description Logics*, 2011.

[49] Mausam, M. Schmitz, S. Soderland, R. Bart, and O. Etzioni. Open language learning for information extraction. In *EMNLP-CoNLL*, pages 523–534. ACL, 2012.

[50] M. Mintz, S. Bills, R. Snow, and D. Jurafsky. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2 - Volume 2*, ACL '09, pages 1003–1011, Stroudsburg, PA, USA, 2009. Association for Computational Linguistics.

[51] T. Mitchell, T. Russell, P. Broomhead, and N. Aldridge. Towards robust computerised marking of free-text responses. In *Proceedings of the 6th CAA Conference*. Loughborough University, 2002.

[52] H. Muller, E. Kenny, and P. Sternberg. Textpresso: An ontology-based information retrieval and extraction system for biological literature. *PLoS Biology*, 2(11):1984–1998, 2004.

[53] E. Oren, K. M?ller, S. Scerri, S. Handschuh, and M. Sintek. What are semantic annotations? Technical report, DERI Galway, 2006.

[54] B. Parsia and E. Sirin. Pellet: An OWL DL Reasoner. In *3rd International Semantic Web Conference (ISWC2004)*, 2004.

[55] B. Parsia, E. Sirin, and A. Kalyanpur. Debugging owl ontologies. In *Proceedings of the 14th International Conference on World Wide Web*, WWW '05, pages 633–640, New York, NY, USA, 2005. ACM.

[56] D. Perez, E. Alfonseca, and P. Rodriguez. Upper bounds of the bleu algorithm applied to assessing student essays. In *Proceedings of the 30th International Association for Educational Assessment (IAEA) Conference*, 2004.

[57] R. Reiter. On closed world data bases. In *Logic and Data Bases*, pages 55–76, 1977.

[58] R. Reiter. A theory of diagnosis from first principles. *Artificial Intelligence*, 32(1):57 – 95, 1987.

[59] S. Riedel, L. Yao, and A. McCallum. Modeling relations and their mentions without labeled text. In *Proceedings of the 2010 European conference on Machine learning and knowledge discovery in databases: Part III*, ECML PKDD'10, pages 148–163, Berlin, Heidelberg, 2010. Springer-Verlag.

[60] A. Ritter, D. Downey, S. Soderland, and O. Etzioni. It's a contradiction—no, it's not: A case study using functional relations. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '08, pages 11–20, Stroudsburg, PA, USA, 2008. Association for Computational Linguistics.

[61] S. Schlobach and R. Cornet. Non-standard reasoning services for the debugging of description logic terminologies. In *IJCAI*, pages 355–362, 2003.

[62] S. Schlobach, Z. Huang, R. Cornet, and F. van Harmelen. Debugging incoherent terminologies. *Journal of Automated Reasoning*, 39(3):317–349, Oct. 2007.

[63] J. H. I. H. D. L. M. P. F. P.-S. L. A. S. Sean Bechhofer, Frank van Harmelen. OWL Web Ontology Language. `http://www.w3.org/TR/owl-ref/`.

[64] B. Settles. Biomedical named entity recognition using conditional random fields and rich feature sets. In *Proceedings of the International Joint Workshop on Natural Language Processing in Biomedicine and Its Applications*, JNLPBA '04, pages 104–107, Stroudsburg, PA, USA, 2004. Association for Computational Linguistics.

[65] R. Snow, D. Jurafsky, and A. Y. Ng. Learning syntactic patterns for automatic hypernym discovery. In *Advances in Neural Information Processing Systems (NIPS 2004)*, November 2004. This is a draft version from the NIPS preproceedings; the final version will be published by April 2005.

[66] P. Srinivasan and X. Y. Qiu. Go for gene documents. *BMC Bioinformatics*, 2007.

[67] G. Stoilos, B. C. Grau, and I. Horrocks. How incomplete is your semantic web reasoner? In *AAAI*, 2010.

[68] D. Swanson. Fish oil, raynauds syndrome, and undiscovered public knowledge. *Perspect. Bio. Med*, 30:7–18, 1986.

[69] D. R. Swanson. Migraine and magnesium: eleven neglected connections. *Perspect Biol Med*, 31(4):526–557, 1988.

[70] D. Tatar, G. Serban, A. Mihis, and R. Mihalcea. Textual entailment as a directional relation. *Journal of Research and Practice in Information Technology*, 41(1):53, 2009.

[71] D. Tsarkov and I. Horrocks. Fact++ description logic reasoner: System description. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 4130 LNAI:292–297, 2006.

[72] H. Wang, M. Horridge, A. Rector, N. Drummond, and J. Seidenberg. Debugging owl-dl ontologies: A heuristic approach. In *Proceedings of the 4th international conference on The Semantic Web*, pages 745–757. Springer, 2005.

[73] D. S. Weld, R. Hoffmann, and F. Wu. Using wikipedia to bootstrap open information extraction. *SIGMOD Rec.*, 37(4):62–68, Mar. 2009.

[74] D. Whittington and H. Hunt. Approaches to the computerized assessment of free text responses. In *In: M. Danson (Ed.), Proceedings of the Sixth International Computer Assisted Assessment Conference*, pages 207–219, 1999.

[75] D. C. Wimalasuriya and D. Dou. Using Multiple Ontologies in Information Extraction. In *Proceedings of the 18th ACM Conference on Information and Knowledge Management (CIKM)*, pages 235–244, 2009.

[76] D. C. Wimalasuriya and D. Dou. Components for Information Extraction: Ontology-Based Information Extractors and Generic Platforms. In *Proceedings of the 19th ACM Conference on Information and Knowledge Management (CIKM)*, pages 9–18, 2010.

[77] D. C. Wimalasuriya and D. Dou. Ontology-based information extraction: An introduction and a survey of current approaches. *J. Inf. Sci.*, 36:306–323, June 2010.

[78] F. Wu, R. Hoffmann, and D. S. Weld. Information extraction from wikipedia: moving down the long tail. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '08, pages 731–739, New York, NY, USA, 2008. ACM.

[79] F. Wu and D. S. Weld. Autonomously semantifying wikipedia. In *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*, CIKM '07, pages 41–50, New York, NY, USA, 2007. ACM.

[80] F. Wu and D. S. Weld. Automatically refining the wikipedia infobox ontology. In *Proceedings of the 17th international conference on World Wide Web*, WWW '08, pages 635–644, New York, NY, USA, 2008. ACM.

[81] F. Wu and D. S. Weld. Open information extraction using wikipedia. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, ACL '10, pages 118–127, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics.

[82] B. Yildiz and S. Miksch. ontox - a method for ontology-driven information extraction. In *ICCSA (3)*, pages 660–673, 2007.

[83] I. Zipitria, J. A. Elorriaga, A. A. Lasa, and A. D. de Ilarraza Snchez. From human to automatic summary evaluation. In *Intelligent Tutoring Systems 2004*, pages 432–442, 2004.