A HYBRID APPROACH FOR ONTOLOGY-BASED INFORMATION EXTRACTION

by

FERNANDO GUTIÉRREZ

A DISSERTATION

Presented to the Department of Computer and Information Science
and the Graduate School of the University of Oregon
in partial fulfillment of the requirements
for the degree of
Doctor of Philosophy

December 2015

DISSERTATION APPROVAL PAGE

Student: Fernando Gutiérrez

Title: A Hybrid Approach for Ontology-based Information Extraction

This dissertation has been accepted and approved in partial fulfillment of the requirements for the Doctor of Philosophy degree in the Department of Computer and Information Science by:

Dr. Dejing Dou	Chair
Dr. Stephen Fickas	Core Member
Dr. Daniel Lowd	Core Member
Dr. Tyler Kendall	Institutional Representative

and

Scott L. Pratt	Dean of the Graduate School

Original approval signatures are on file with the University of Oregon Graduate School.

Degree awarded December 2015

DISSERTATION ABSTRACT

Fernando Gutiérrez

Doctor of Philosophy

Department of Computer and Information Science

December 2015

Title: A Hybrid Approach for Ontology-based Information Extraction

Information extraction (IE) is the process of automatically transforming written natural language (i.e., text) into structured information, such as a knowledge base. However, because natural language is inherently ambiguous, this transformation process is highly complex. On the other hand, as IE moves from the analysis of scientific documents to the analysis of Internet textual content, we cannot rely completely on the assumption that the content of the text is correct. Indeed, in contrast to scientific documents, which are peer reviewed, Internet content is not verified for the quality and correctness.

Thus, two main issues that affect the IE process are the complexity of the extraction process and the quality of the data.

In this dissertation, we propose an improved ontology-based IE (OBIE) by providing solutions to these issues of accuracy and content quality. Based on a hybrid strategy that combines aspects of IE that are usually considered as opposite to each other, or that are not even considered, we intend to improve IE by developing a more accurate extraction and new functionality (semantic error detection). Our approach is based on OBIE, a sub-area of IE, which reduces extraction complexity by including

domain knowledge, in the form of concepts and relationships of the domain, to guide the extraction process.

We address the complexity of extraction by combining information extractors that have different implementations. By integrating different types of implementation into one extraction system, we can produce a more accurate extraction. For each concept or relationship in the ontology, we can select the best implementation for extraction, or we can combine both implementations under an ensemble learning schema. In tandem, we address the quality of information by determining its semantic correctness with regard to domain knowledge. We define two methods for semantic error detection: by predefining the types of errors expected in the text or by applying logic reasoning to the text.

This dissertation includes both published and unpublished coauthored material.

CURRICULUM VITAE

NAME OF AUTHOR: Fernando Gutiérrez

GRADUATE AND UNDERGRADUATE SCHOOLS ATTENDED:
University of Oregon, Eugene, OR, USA
University of Concepción, Concepción, Chile

DEGREES AWARDED:
Doctor of Philosophy in Computer Science, 2015, University of Oregon
Master of Science in Computer Science, 2009, University of Concepción
Bachelor in Computer Science, 2008, University of Concepción

AREAS OF SPECIAL INTEREST:
Knowledge representation, text mining, information extraction

PROFESSIONAL EXPERIENCE:

GRANTS, AWARDS AND HONORS:

Graduate Teaching & Research Fellowship, Computer and Information Science, 2014 to present

Advanced Human Resource Program Scholarship, CONICYT, Chile, 2009 2013

PUBLICATIONS:

Fernando Gutierrez, Dejing Dou, Stephen Fickas, Daya Wimalasuriya, and Hui Zong 2015. A Hybrid Ontology-based Information Extraction System. (Accepted by) Journal of Information Science, 2015.

Jingshan Huang, Fernando Gutierrez, Dejing Dou, Judith A. Blak, Karen Eilbeck, Darren A. Natale, Barry Smith, Yu Lin, Xiaowei Wang, Zixing Liu, Ming Tan, and Alan Ruttenberg, A semantic approach for knowledge capture of microRNA-target gene interactions, Proc. BHI Workshop at 2015 IEEE International Conference on Bioinformatics and Biomedicine (BIBM-15), IEEE, Washington D.C., Nov. 2015.

Fernando Gutierrez, Dejing Dou, Steven Fickas, and Gina Griffiths 2014. Online Reasoning for Ontology-based Error Detection in Text. In Proceedings of the 13th International Conference on Ontologies, Databases and Application of Semantics (ODBASE 2014). pp. 562-579, 2014.

Fernando Gutierrez, Dejing Dou, Steven Fickas, Adam Martini, and Hui Zong 2013. Hybrid Ontology-based Information Extraction for Automated Text Grading. In Proceedings of the 12th IEEE International Conference on Machine Learning and Applications (ICMLA 2013). December 2013.

Fernando Gutierrez, Dejing Dou, Stephen Fickas, and Gina Griffiths. 2012. Providing grades and feedback for student summaries by ontology-based information extraction. In Proceedings of the 21st ACM international conference on Information and knowledge management (CIKM 2012). October 2012.

Chang-Hwan Lee, Fernando Gutierrez, and Dejing Dou 2011. Calculating Feature Weights in Naive Bayes with Kullback-Leibler Measure. In Proceedings of the 11th IEEE International Conference on Data Mining (ICDM 2011). December, 2011.

Fernando Gutierrez, Daya C. Wimalasuriya, and Dejing Dou 2011. Using Information Extractors with the Neural ElectroMagnetic Ontologies. In Proceedings of the International Conference on Ontologies, Databases and Application of SEmantics (ODBASE 2011). October, 2011.

Fernando Gutirrez, John Atkinson 2010. Adaptive feedback selection for intelligent tutoring systems. Expert Systems with Applications. Volume 38, Issue 5, pp 6146-6152. 2011.

Fernando Gutirrez, John Atkinson 2009. Evolutionary constrained self-localization for autonomous agents. Applied Soft Computing, Volume 11, Issue 4, pp 3600-3607, June 2011.

Fernando Gutirrez, John Atkinson 2009. Autonomous Robotics Self-Localization Using Genetic Algorithms. In Proceedings of the 21st International Conference on Tools with Artificial Intelligence (ICTAI 2009). November 2009.

John Atkinson, Jonnattan Gonzalez, Claudio Castro, Dario Rojas, Aroldo Arriagada, Fernando Gutierrez, 2006. UdeCans Team Description, In Proceedings of the 3rd IEEE Latin American Robotics Symposium (LARS 2006). October 2006.

ACKNOWLEDGEMENTS

I would like to acknowledge my research advisor, Dr. Dejing Dou, for the the tireless support and patience during these years that led this disseratation. I also would like to thank the committee members for my dissertation and the Computer and Information Science (CIS) faculty and staff for their time and help through my time there.

This work is dedicated to my wife and daughters for their encouragement, support and endurance.

TABLE OF CONTENTS

Chapter Page

I. INTRODUCTION 1

II. BACKGROUND 7

2.1. Ontology and Inconsistency 7
2.2. Information Extraction 14

III. HYBRID INFORMATION EXTRACTION 31

3.1. Redefening Information Extractors 32
3.2. Combining Implementations 33
3.3. Evaluation 39

IV. PRECOMPUTING SEMANTIC INCORRECTNESS 55

4.1. Determining Sentence Types 56
4.2. Extracting Sementics Errors 59
4.3. Evaluation 63

Chapter Page

V. ONLINE REASONING FOR SEMANTIC ERROR DETECTION . . . 74

5.1. Transforming Text to Logic Clauses 75

5.2. Single Sentence Analysis 79

5.3. Multiple Sentence Analysis 86

5.4. Evaluation 92

VI. CONCLUSION 100

6.1. Future Work 102

APPENDICES

A. SEQUENT CALCULUS 105

B. INCONSISTENCY AS COMPLEMENT ENTAILMENT 107

REFERENCES CITED 108

LIST OF FIGURES

Figure Page

2.1 Ontology-based Components for Information Extraction. 18

3.1 Graphical representation of a section of the ontology associated to the MUC4 dataset. 41

3.2 Graphical representation of a section of the ontology development for this work. 48

3.3 Precision, recall and F1 measure for information extractors under different levels of error in text, with single implementation (ER and ML), and multiple implementations with our proposed combination strategies (MinError and StackNB) 53

3.4 Precision, recall and F1 measure for information extractors with single implementation (ER and ML), multiple implementations without a combination strategy (1ML-3ER, 2ML-2ER and 3ML-1ER), and multiple implementations with our proposed combination strategies (MinError and StackNB) 54

4.1 Graphical representation of a section of the Ecosystems ontology 65

4.2 Precision, recall and F1 measure for information extractors under different levels of error in text, with single implementation (ER and ML), and multiple implementations with our proposed combination strategies (MinError and StackNB) with the functionality of extracting incorrect statements. 71

4.3 Precision, recall and F1 measure for information extractors with single implementation (ER and ML), multiple implementations without a combination strategy (1ML-3ER, 2ML-2ER and 3ML-1ER), and multiple implementations with our proposed combination strategies (MinError and StackNB) with the functionality of extracting incorrect statements. 72

Figure Page

4.4 Comparison of correct statement extraction and error extraction functionality in terms of precision, recall and F1 measure for information extractors with single implementation (ER and ML), multiple implementations without a combination strategy (1ML-3ER, 2ML-2ER, and 3ML-1ER), and multiple implementations with our proposed combination strategies (MinError and StackNB). . . . 73

5.1 Example of mapping between extracted terms and ontology concepts. . 78

5.2 Graphical representation of a section of the County ontology 98

LIST OF TABLES

Table Page

3.1 Statistical information about the ontology. 40

3.2 Performance of extraction by our proposed methods of selection (*MinError*) and integration (*StackNB*), the method by Wimalasuriya and Dou (OBCIE), rule-based extraction, and machine learning-based implementation. 44

3.3 Number of template sentences for each concept. 47

3.4 Statistical information about the ontology. 49

4.1 Statistical information about the ontology. 65

4.2 Performance of IE 67

4.3 Correlation between grading metrics 68

4.4 Example of summaries. 69

5.1 Comparison between statistical information about the original ontology presented in Section 4.3.1. to evaluate the *precomputed* approach and the extended ontology used to evaluate our proposed *online reasoning* approach. 93

5.2 Precision (top), recall (center), and F1 measure (bottom) for the proposed method (automatic and manually extraction) and for the *precomputed* approach [1]. 95

5.3 Statistical information about the ontology. 98

CHAPTER I

INTRODUCTION

Information Extraction (IE) is the process of automatically transforming natural language text into structured information (e.g., relational databases) [2]. This transformation occurs by identifying semantically relevant elements, such as entities (e.g., concepts and instances) and relationships. IE has become a key approach to text understanding in many applications, such as automatic text grading [3, 4], transforming Web content into structured Semantic Web information [5, 6], and helping identify candidates for clinical trials [7] among others. However, converting available textual information into a structured format is not a trivial task. There are two main issues in this transformation process: the complexity of the IE, and the quality of the data.

Because of the inherent ambiguity of natural language (i.e., words have multiple meanings), the process of extracting information from text is far from trivial. Ontology-based Information Extraction (OBIE), a subfield of IE, mitigates this difficulty by integrating domain knowledge through a domain ontology. An ontology is an explicit specification of a shared conceptualization that represents knowledge through concepts, relationships, and individuals [8]. These concepts and properties guide the extraction process in OBIE [9, 10]. However, OBIE can introduce new problems into the extraction process. Creating and maintaining ontologies used by an OBIE system are rather complex tasks. These difficulties can be mitigated by utilizing domain ontologies offered by a third party (e.g., Bioportal) [11, 12], although some cases require application-specific ontologies [1]. On the other hand, because OBIE systems are created with a specific ontology in mind, they need to be redesigned when

used under a different ontology. These obstacles translate into costly deployment and maintenance of OBIE systems, limiting their adoption.

As a way to promote the adoption of OBIE, Wimalasuriya and Dou have proposed the Ontology-based Components for Information Extraction (OBCIE) architecture [13]. OBCIE aims to encourage re-usability by modeling the components of the IE system with as much modularity as possible. This modularity is achieved through the separation between domain-dependent components (i.e., information extractors) and domain-independent components (i.e., platform components). Information extractors are the IE components that perform the extraction task. Each information extractor encodes a specific component of the ontology (e.g., a concept), so that extractions will depend only on this ontological element. On the other hand, the IE platform components are the elements of the system that implement IE techniques, which are both domain- and corpus- independent. These techniques can be as simple as preprocessing modules (e.g., removing special characters from the text) to complex ontology learning components (i.e., determining hierarchy and relationships between extracted elements).

On the other hand, as IE and OBIE move from the analysis of scientific documents to the analysis of Internet textual content, we cannot rely completely on the assumption that the content of the text is correct. Indeed, in contrast to scientific documents, which are peer reviewed, Internet content is not verified, neither for its quality, nor for its correctness. So, it seems reasonable to consider, as part of the process of extracting textual information, mechanisms, and techniques that allow us to differentiate between correct and incorrect information.

However, it is not easy to determine the correctness of a sentence; hence, this need has been addressed only indirectly. Research from the field of educational, such

as automatic text grading, has mainly treated incorrectness as low similarity to a gold standard. Automatic grading systems based on Latent Semantic Analysis [14] and *n-gram* co-occurrence [15] try to determine how similar a student's summary or essay is with respect to a *perfect* summary or essay. If the student's writings have low similarity to the gold standard, this is interpreted to mean that it is *less correct*. However, low similarity can still be obtained in the process of a correct text, such as if the text was written in an unexpected fashion, or if the text content is broader than the gold standard. On the other hand, incorrectness can be identified in the presence of contradiction. The research area of Contradiction Detection [16], an extension of Textual Entailment, intends to identify in text a pair of sentences that cannot be true at the same time (i.e., a logic contradiction). By identifying specific lexical and syntactical elements, the semantics of the sentences are captured and compared. However, since Contradiction Detection is limited to the content of the text, itself, in order to support the correctness of the contradicting sentences, it cannot determine with certainty which sentence of the pair is false.

In this dissertation, we propose an improved Ontology-based Information Extraction approach by providing solutions to these issues of accuracy and content quality. Based on a hybrid strategy that combines aspects of IE that are usually considered as opposite to each other, or not even considered, we intend to improve OBIE, and expand IE, in terms of a more accurate extraction and a new functionality. Brief descriptions of these directions are presented below.

1. **Hybrid Implementation.** Independent of the ontological component it represents, an information extractor can be implemented as an extraction rule, or by applying machine learning methods [13]. Based on regular expressions, extraction rules capture information by identifying specific elements in a text.

They can be based on lexical elements (i.e., keywords), syntactical elements (e.g., noun phrases), or both. On the other hand, information extractors can also be based on machine learning methods, such as Naive Bayes [17] and Conditional Random Fields [18]. Under this approach, the information extraction process is transformed into a supervised learning task, in which classification methods and probabilistic models try to identify which elements of a sentence are part of the sought information [13]. Although for any given implementation strategy, there are some concepts that are more difficult to extract than others, most IE systems only consider one type of implementation. With this in mind, we have proposed a hybrid OBIE system, which incorporates both extraction rules and machine learning-based information extractors.

We begin by proposing proposed combining information extractors that have different implementations. This approach leads to higher precision and recall than using only one type of implementation. Then, to obtain the best performance from this hybrid implementation approach, we also propose two types of strategies for combining information extractors: selection and integration. While the *selection* strategy identifies the set of information extractors that commits the fewest extraction errors, the *integration* strategy combines the outputs of different implementations to produce a more accurate extraction. For each one of these strategies, we propose a specific method that focuses on obtaining the highest accuracy. For the selection strategy, we follow an error minimization approach in order to obtain the subset of information extractors that perform the most accurate extraction.

2. **Semantic Error Detection.** Although traditional IE makes the assumption that the content of the text is correct, when we consider domains such as the

Internet, where there are no guarantees about the correctness of the content, this assumption does not hold. So, it seems reasonable to consider, as part of the process of extracting textual information, mechanisms and techniques that allow us to differentiate between correct and incorrect information. However, the text, itself, is not a sufficient basis for determining the correctness of its content. At most, it can be used to identify internal contradiction, i.e., a set of sentences that cannot all be true at the same time [16, 19].

We propose the use of domain knowledge, through an ontology, as a framework to determine the semantic correctness of a text. We define two methods for semantic error detection based on ontology debugging, which is the area of research that identifies the origin of inconsistency in an ontology [20], and ontological constraints (e.g., disjointness between concepts). The first method uses a heuristic to generate domain-inconsistent axioms that are encoded into information extractors. These domain-inconsistent information extractors can identify semantically erroneous sentences. The second method uses logic reasoning to detect errors in a statement from text online. Such an approach applies Information Extraction to transform text into a set of logic clauses. The logic clauses are incorporated into the domain ontology to determine if they contradict the ontology or not.

The remainder of this dissertation is organized as follows. In Chapter II, we discuss background areas related to the original research work presented. The main contributions of this dissertation are presented in Chapters III, IV, and V. In Chapter III we discuss our hybrid implementation approach, while in Chapters IV and V we discuss our two approaches to semantic error detection. Finally, in

Chapter VI, we provide some concluding remarks and discuss future directions for our research.

This dissertation includes published and unpublished coauthored materials. I acknowledge the contribution of Dr. Dejing Dou, my advisor, who participated in the design and development of the hybrid implementation approach to OBIE described in Chapter III, and in the design of the semantic error detection methods described in Chapters IV and V. I am also thankful to each of the following coauthors for their unique contributions: to Prof. Stephen Fickas, who contributed to the design of semantic error detection methods in Chapters IV and V; to Dr. Gina Griffiths, who contributed to the study on the students' summary dataset in Chapters IV and V; to Dr. Daya Wimalasuriya, who contributed to the design hybrid implementation explored in Chapters III; to Dr. Hui Zong, who contributed to the study on the cell biology exam answers dataset described in Chapters III and IV; and to Adam Martini, who contributed to the processing of the cell biology dataset used in Chapters III and IV.

CHAPTER II

BACKGROUND

This chapter covers the background areas and related work necessary to understand the contributions of this dissertation. We propose a method to combine information extractors into a hybrid implementation. This has led to consider OBIE platforms architectures. We also propose a method to identify semantic error in text by incorporating domain knowledge to the extraction process. This has led us to consider research regarding Consistency Checking, Ontology Debugging, and Information Extraction.

2.1. Ontology and Inconsistency

In Artificial Intelligence, an ontology is an explicit specification of a conceptualization [8]. 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" [8]. 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) [21]. OWL is the standard ontology language proposed by the *World Wide Web Consortium* (W3C) [22]. Description logic (DL) is a fragment of *first-order logic* that is decidable, and it has sound and complete reasoners [23–25].

2.1.1. Consistency Checking

As mentioned, an ontology describes a domain through concepts and relationships, categorizing the entities of the domain and their properties and relations. In this work, we will focus on ontologies that are based on Description Logic, such as those described by the Web Ontology Language (OWL) [21] proposed by the World Wide Web Consortium (W3C).

Description Logic (DL) is a set of knowledge representation languages. Many DL languages are decidable fragments of first-order logic (e.g., $\mathcal{SHOIQ}$), with sound and complete reasoners such as *HermiT* [26] and *Pellet* [24]. In DL, concepts and relationships (i.e., roles) are defined by boolean constructors, such as conjunction ($\sqcap$) and disjunction ($\sqcup$), existential ($\exists$) and universal ($\forall$) value restrictions. In DL, a knowledge base $\mathcal{K}$ consists of a tuple $(\mathcal{R}, \mathcal{T}, \mathcal{A})$. The *TBox* $\mathcal{T}$ is a set of *general concept inclusions* (GCI) of the form $C \sqsubseteq D$, for concepts C and D. The *ABox* has concepts and role assertions of the form $C(a)$ and $R(a, b)$. Finally, the *RBox* $\mathcal{R}$ consists of complex role constructions such as role inclusion ($R_1 \sqsubseteq R_2$). However, not all DL knowledge bases define a *RBox*, such as $\mathcal{ALC}$, because they do not have role construction.

The semantics of a DL knowledge base $\mathcal{K}$ is defined by an interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$. The function $\cdot^{I}$ maps the knowledge base $\mathcal{K}$ to the domain $\Delta^{\mathcal{I}}$. Under $\mathcal{I}$, each concept C of $\mathcal{K}$ is a subset of the domain ($C^{\mathcal{I}} \subset \Delta^{\mathcal{I}}$), each role R of $\mathcal{K}$ is a subset of product of the domain ($R^{\mathcal{I}} \subset \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$), and each individual a of $\mathcal{K}$ is an element of the domain ($a^{\mathcal{I}} \in \Delta^{I}$). If $\mathcal{I}$ satisfies all axioms of $\mathcal{K}$, then $\mathcal{K}$ is consistent, and $\mathcal{I}$ is a model of $\mathcal{K}$. The interpretation consists of a domain ($\Delta^{\mathcal{I}}$) and a mapping function ($\cdot^{I}$). The function maps the concepts, roles and assertions of the knowledge base to the domain. If $\mathcal{I}$ satisfies all axioms of $\mathcal{K}$, then I is a model of $\mathcal{K}$, which makes $\mathcal{K}$

consistent. A basic service of a DL reasoner is to determine if a knowledge base $\mathcal{K}$ is satisfiable.

Determining the satisfiability of a knowledge base $\mathcal{K}$ (i.e., consistency checking of $\mathcal{K}$) is a fundamental task for a DL reasoner. Its importance comes from the fact that other types of inferences, such as entailment, can be reduce to satisfiability [27]. Satisfiability can be proved by a decision procedure such as a *semantic tableau* (i.e., tableau algorithm). This method creates a sequence $1..n$ of *ABoxes*, where the application of derivation rules on *ABox* ($\mathcal{A}_{i-1}$) result in a new *ABox* ($\mathcal{A}_i$) [26]. Following are commonly used tableau derivation rules for DL:

- Given $C \sqsubseteq D$ and an individual s, derive $(\neg C \sqcup D)(s)$.
- Given $(C \sqcup D)(s)$, derive $C(s)$ or $D(s)$.
- Given $(C \sqcap D)(s)$, derive $C(s)$ and $D(s)$.
- Given $(\exists R.C)(s)$, derive $R(s,t)$ and $C(t)$ for a new individual t.
- Given $(\forall R.C)(s)$ and $R(s,t)$, derive $C(t)$.

The tableau algorithm terminates if no more derivation rules can be applied to *ABox* ($\mathcal{A}_n$), or if we reach a contradiction. In the case of contradiction, the algorithm backtracks to the last OR derivation and *choose* a different *path*. For example, if deriving $C(s)$ from $(C \sqcup D)(s)$ leads to a contradiction, we need to derive $D(s)$. If all choices lead to contradiction, $\mathcal{K}$ is unsatisfiable.

2.1.2. 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 [20]. 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.

As stated by Haase and Volker [28], 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. [29] splits 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. [29] 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) [28, 30–35]. For this work, when referring to logical inconsistency, we will be considering the most general definition (logical contradiction).

2.1.3. Ontology Debugging

The process of correcting an inconsistent ontology is called *Ontology Debugging* [20]. 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 obtained by removing the inconsistent elements from the ontology.

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 [36]. Based on the definition of entailment justification by Kalyanpur et al. [37], Horridge et al. [32] 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 first need to identify all the inconsistency justifications it contains. In the Schlobach and Cornet [34] 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 [34] 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. [35] offer an alternative mechanism to identify each concept's *MUPS*. Based on Huang et al. [33] 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 concept's *MUPS*.

In the case of Horridge et al. [32], the inconsistent subsets of the ontology are obtained by a modified version of the single justification algorithm, from the *entailment justification* method [37]. This algorithm identifies subsets of the ontology that are inconsistent through the division of the ontology. The intuition suggests 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 [38] to identify a repair in the inconsistent ontology from set of justifications of the inconsistency. Reiter propose the Hitting Set Tree (HST) [38] 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). Reiter's HST finds the diagnosis by learning how the conflict sets intersect. The HST algorithm iteratively searches or access the set of conflict sets, to constructs a tree where each node indicates a conflict set, while 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.

2.2. 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 structured 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 [39], but in most cases each task can be carried out independently [17, 40, 41]. Jurafsky and Martin [2] 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 [42].

- **Coreference resolution**: is the process of determining if the mention of a same or similar name refers to the same entity; 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 [43], knowledge base population [44], and semantic annotation [40].

- **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.

2.2.1. 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 2.1., an ontology is defined as a formal and explicit specification of a shared conceptualization [8]. 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 [9].

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 [10, 11]. 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 [45].

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 to open domains, such as the Internet. With that in mind, an ontology-based information extraction system seems 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.

2.2.2. Ontology-based Components for Information Extraction

As mentioned, the Ontology-based Components for Information Extraction (OBCIE) architecture [13] was proposed to promote the adoption of OBIE systems by reducing the costs of deployment and management through modularity. In OBCIE, an IE system is constituted of a set of modules that perform specific tasks. The modules can be grouped as domain-dependent (i.e., information extractors) and domain independent (i.e., IE platform). This separation in OBCIE promotes re-

usability in two forms: by allowing an information extractor to be used (and re-used) by any IE platform, and by allowing an IE platform to use any set of information extractors it requires.

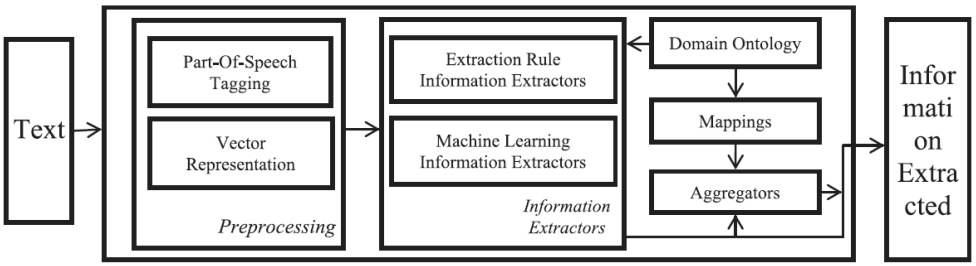

FIGURE 2.1. Ontology-based Components for Information Extraction.

In order to provide a more clear understanding of our proposed strategies, in the following section we provide a brief introduction to the main OBCIE components that are involved in the extraction process (Figure 2.1), which interact with the elements of our hybrid approach.

2.2.2.1. Ontology

As previously mentioned, an ontology is an explicit specification of a shared conceptualization [2]. Through concepts, relationships, axiomatic constraints, and individuals, an ontology provides a formal representation of domain knowledge. In OBCIE, the ontology is also a module that can be reused. Therefore, for any given domain where the OBCIE platform is going to be deployed, if there are available ontologies, they can be used for the OBIE process. For example, in the biomedical domain there are publicly accessible ontologies. Through BioPortal [21] at the National Center of Biomedical Ontology, it is possible to access more than 300 ontologies (e.g., BioModels Ontology, CRISP).

2.2.2.2. Preprocessors

Preprocessors are the modules that perform modifications to the text to facilitate and improve the extraction process. The modifications can filter unwanted elements from the text (e.g., stopword removal), enhance the text with new information (e.g., as *part-of-speech* tagging), or transform the representation of the text (e.g., vector representation). These modifications are mostly independent of each other, and they are -usually- applied in a sequence. For example, it is very common that before a text is transformed into a vector representation, *stopwords* are filtered, and *part-of-speech* tagging is applied.

In general, preprocessors are independent of the domain and the information extractors because they remove noisy features and enhance the text's representation. However, there are preprocessing tasks that are developed for a specific domain. For example, in some domains, concepts might be referred to by their name and their acronym (e.g., ETC and electron transport chain). To avoid multiple interpretations between different representations of the same concept, a preprocessor would replace all representations of a concept into a single one (e.g., *ETC*, *ETCs*, and electron transport chain are changed to *ETC*). It is also the case that some information extractors have specific requirements. For example, a machine learning-based information extractor will very likely require a vector representation of the text, such as *term frequency-inverse document frequency* (TF-IDF). In some cases, the information extractor might require a vector representation that includes alternative features, such as the position of the words in the sentence [15].

2.2.2.3. Information Extractors

Information extractors are the main components of the OBIE system since they perform the extraction from the text. Each information extractor is defined by an ontological element, to which they are bound [8, 16]. An information extractor identifies the textual representation of a specific ontological element. In other words, for each concept (or property) of the ontology we intend to extract from the text, we need to define a specific information extractor. More formally, let us consider the sentence $x_s \in X$, where xx corresponds to a set of sentences from the domain xx. Let us also consider a concept c from the ontology xx of the same domain xxx. An information extractor ec will determine the connection between sentence xx and concept c by resolving the sentence's semantic content yyy x in the form:

$$e^c(x_s) = y_s^c \qquad \text{(Equation 2.1.)}$$

Depending on how e^c is specified, y_s^c can vary. In the most simple case, $y_s^c \in \{0, 1\}$ tells us if sentence xx contains a reference to concepts c (i.e., $y_s^c = 1$), or if it does not contain the reference (i.e., $y_s^c = 0$). It is also possible that $y_s^c \subseteq x_s$, meaning that there is a specific part of the sentence that is referring to concept c. This output is useful when performing tasks such as semantic annotation over the text. Another alternative is for the information extractor ec to produce a triple as output. In this case, $y_s^c = R_c(a, b)$, where R_c represents a property of c (i.e., relationship), with a as domain of xx (and also as an instance of c), and b as range. This output is useful when trying to populate a knowledge base with information from text.

Independent of the ontological component it represents, an information extractor can be implemented as an extraction rule or by applying machine learning methods [13].

In OBCIE, an information extractor component must contain most (if not all) the elements that are required for it to be used by the system. For example, a rule-based extractor component has defined the extraction patterns it uses, plus gazetteer lists that are associated with that component. For a machine learning-based extractor, it will have the set of features (e.g., keywords) needed for the extraction. This approach (i.e., self-contained extractor) allows us to reconfigure the OBIE system, in term of extraction, with minimal change to the whole system. We can remove or add extractors without affecting the rest of the extractors or the domain-independent components.

2.2.2.4. Aggregators

In most cases, the outputs of the information extractors of an OBIE system correspond to the final extraction output. However, there are cases where the combination of extracted outputs can improve the extraction process. For example, Wimalasuriya and Dou [6] have proposed a mechanism to do OBIE by using two or more ontologies of the same domain. By using mappings between concepts from different ontologies, we can determine which information extractors to combine (through set operators). Because two ontologies, in most cases, can offer different interpretations of the same domain, Wimalasuriya and Dou's approach can produce a more semantically complete extraction. OBCIE architecture has included this combination approach as an aggregation module [8].

2.2.3. Classification of Information Extraction

In their survey of Ontology-based Information Extraction, Wimalasuriya and Dou [9] offer a classification of information extraction systems based on different characteristics of the systems, such as: (i) the extraction methods used, (ii) if the system constructs or updates the ontology, (iii) what type of components of the ontology are extracted, and (iv) 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 [46, 47] or as features of a machine learning based extractor [11, 41]. Most approaches have focus on extracting instances of concepts and relationship [17], they use available ontologies [11] and knowledge bases [40], and the Internet is their corpus of analysis [46–48].

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 types 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* [49]) or unsupervised (e.g., *TextRunner* [17]) system.

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

2.2.3.1. Supervised Information Extraction Systems

Supervised information extraction systems, also known as closed or traditional information extraction system [18], 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 it is not possible 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 [1, 11, 50]. Based on the type of information extraction, there are two main strategies followed by supervised systems [9]: *extraction rules* and *machine learning*.

Extraction rules capture information by identifying specific syntactic and lexical elements in text, such as keywords [50], 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 defined following regular expressions, languages like *SystemT's* Annotation Query Language (AQL) [51] and GATE's Java Annotation Patterns Engine (JAPE) [52] have been created to specify extraction patterns. These specially designed 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 [51], while JAPE can directly execute Java code from the matching of a pattern [52].

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 that 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 [13]. 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. Equivalently, 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 a 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 classified in the initial phase [53].

2.2.3.2. Semi-supervision Information Extraction 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, as in the work of *Kylin* that exploits the relation between

Wikipedia's infoboxes and articles [41, 54, 55]. In other cases, where the corpus of text is the Web [39, 40, 44, 56], or text in other language [57], 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: first instances from knowledge base are looked up in sentences of the text; afterwards selected sentences are transformed into sets of relevant elements; finally 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* [39], *Distant Supervision* (DS) [40], and the system by Snow et al. [56], 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 [58].

On the other hand, *Kylin* [55] determines the sentences where the instances are mentioned 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. Wikipedia's 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 features [39, 40, 55, 56]. DS considers that although a selected sentence has the relation's entities, it is quite possible that the sentence also have *noise*. To learn a robust classifier that can manage this noise, each sentence is transformed into a large set of features. These features are lexical and syntactical, and they 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, 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 to learn from example sentences. 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 consists on applying a machine learning technique. *Kylin* uses Conditional Random Fields to learn a sequence model from the sentence by consideration of 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* [41] improves the quality of Wikipedia's 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, the Carlson et al. [44] 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 either mutually excluded (instances of disjoint concepts), or have an erroneous type.

After a model or pattern of extraction is learned, new instances can be extracted from text [48]. These new instances can lead to a new learning process, that can produce higher quality extractors [54].

2.2.3.3. Unsupervised Information Extraction 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* [59]. 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 relation 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, the 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* [60, 61] and *TextRunner* [17, 18] to the extraction of a wide variety of relationship instances. In their case study, Banko et al. [17] 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 [17] or Conditional Random Fields [18], it can produce *incoherent* and *uninformative* extractions. Incoherent extractions are produced when the sequence of decisions lead to an incorrect extractor. Uninformative extraction occur when relevant information is removed from the relational phrase because it is incorrectly handled.

In order to reduce these erroneous extractions, *ReVerb* [46] 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* [60] measure the quality of an extraction pattern based on redundancy of instances being extracted together [62]. 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* [18] 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* [46] 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% [46] to 95% [18] 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 [47, 59] or an extraction model [17, 18]. In the case of *Ollie* [47], 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.

CHAPTER III

HYBRID INFORMATION EXTRACTION

This chapter consists of work publish in the "Journal of Information Science" in 2015 [63], and in the "Proceedings of the 12th IEEE international conference on machine learning and applications" in 2013 [53]. Dr. Dejing Dou, Dr. Stephen Fickas, and Dr. Daya Wimalasuriya contributed in the design of the method propose in this chapter. Dr. Hui Zong contributed with the biology exam answers dataset. Adam Martini contributed with the processing of the dataset.

Independent of the ontological component it represents, an information extractor can be implemented as an extraction rule or by applying machine learning methods [13]. Based on regular expressions, extraction rules capture information by identifying specific elements in a text. They can be based on lexical elements (i.e. keywords), syntactical elements (e.g. noun phrases), or both. On the other hand, information extractors can also be based on machine learning methods such as Naive Bayes [17] and Conditional Random Fields [18]. Under this approach, the information extraction process is transformed into a supervised learning task, in which classification methods and probabilistic models try to identify which elements of a sentence are part of the sought information [13]. Although for any given implementation strategy, there are concepts that are more difficult to extract than others, most IE systems only consider one type of implementation.

With this in mind, we propose a *hybrid implementation* for OBIE systems, which incorporates both extraction rules and machine learning-based information extractors. We have found that our combination of information extractors that have different implementations can obtain a higher precision and recall than using

only one type of implementation. In order to obtain the best performance from this hybrid implementation approach, we also propose two types of strategies for combining information extractors: selection and integration. While the selection strategy identifies the set of information extractors that commits the smallest quantity of extraction errors, the integration strategy combines the outputs of different implementations to produce a more accurate extraction. Because of its modular approach to design OBIE systems, we have used OBCIE, mentioned in Section 2.2.2. to incorporate this new approach to implementing information extractors.

3.1. Redefening Information Extractors

In order to implement this hybrid approach into the OBCIE architecture, we offer a new characterization of information extractors. Traditionally, an information extractor has been defined by the concept or property from the ontology [12, 41] it extracts. We have extended the definition of information extractors by considering a new fundamental and orthogonal aspect (i.e., dimension): implementation of the information extractor [53]:

$$e_i^c(x_s) \qquad \text{(Equation 3.1.)}$$

Each information extractor (e) encodes an ontological concept or property (c), under a rule-based or machine learning-based implementation (i). This new dimension (implementation) allow us to include information extractors of different concepts, using different implementations, into the same extraction system, i.e., a hybrid OBIE system.

The original definition of information extractor included requirements to be self-contained and to have platform independence. This new characterization of

an information extractor does not conflict with those requirements. In terms of platform independence, the OBCIE architecture considers domain-independent elements, such as preprocessors, as per-demand functions. If, for example, an information extractor requires part-of-speech labels, the platform will add them to (or with) the text. In terms of self-containment, this new characterization maintains the information extractors' modularity. As mentioned, information extractors in OBCIE had implementation elements already contained in the extractor, as a mechanism to support the modular approach of the architecture. Our new definition simply makes this aspect visible to the system at any time. On the other hand, this new characterization allows establishing relations between information extractors. For each concept, there is one information extractor based on machine learning, and another based on extraction rules.

3.2. Combining Implementations

In most IE systems, the selection of a type of implementation for the extraction process is made by considering the guarantees the implementation can offer in terms of accuracy [47], and the features and restrictions the extraction process as whole might have [39, 55]. From the information extractor $e_i^c(x_s)$, we expect to obtain the semantic content y_s^c by following implementation strategy i, which can be extraction rules or machine learning. Once the selection is made, it is applied to the complete IE process. However, any real implementation of e_i^c can only offer an approximation of the actual semantic content of the sentence:

$$e_i^c(x_s) \approx y_s^c \qquad \text{(Equation 3.2.)}$$

Even more, as it can be seen from the experimental results of different IE systems, one implementation strategy cannot reach the same level of accuracy across all extracted ontological elements [1, 12, 41, 44, 55]. This behavior might be originated when some fundamental characteristic of an implementation strategy collides with the textual representation of some ontological elements. Extraction rules are built on patterns observed from a set of examples. In some cases, the examples lead to tight patterns that allow very little error in the extraction process. However, the high specificity of extraction rules does not permit many variations in the instance to be extracted, and it can lead to an incomplete extraction. If unobserved instance diverges from the set used for the construction of the extraction rule, it is possible that it will not be extracted. In other cases, if examples differ significantly from each other, it leads to error-prone patterns or multiple highly specific patterns. On the other hand, machine learning-based information extractors learn a model that should fit the training data in a fashion that can guarantee some flexibility to manage unseen instances. This flexibility produces an almost complete extraction process, since the extractor can identify instances that have not been seen. However, in a similar way as extraction rules, this flexibility can also be the weakness of the machine learning-based extraction. Because the model is more general than the instances observed in the training set, it is possible that the method can extract unrelated elements.

Based on the OBCIE architecture, we have designed and included into an OBIE system information extractors with different types of implementation, i.e., a hybrid OBIE. We explore the impact that a hybrid OBIE can have when extracting information as part of an evaluation system [53]. We found that improvements were observable even when choosing an arbitrary configuration, e.g., for extracting $n + m$ concepts, we use n machine learning-based extractors and m extraction rules.

Some of these configurations can produce more accurate extraction than when one implementation approach is used for all information extractors. However, not all configurations lead to improvement. Some configurations can also perform worse than the single implementation approach, e.g., selecting the worst implementation strategy for each concept [53].

To take full advantage of this hybrid implementation approach, we propose two types of strategies that can determine which information extractors are used: selection and integration. The first strategy intends to determine the most accurate implementation of each information extractor, while the second strategy combines the outputs of the implementations to improve accuracy.

3.2.1. Selection Strategy

The main goal behind the selection strategy is to determine the best subset of information extractors of the OBIE system that can achieve highest accuracy. In other words, we want to define a selection strategy that permits us to identify the information extractor that possesses the most accurate implementation, for each concept. At the beginning of Section 3.2., we have defined the output of an information extractor as an approximation to the semantic content y_s^c of sentence x_s with respect to concept c. An implementation $e_i^c(x_s)$ will produce an accurate extraction if its difference with the actual semantic content is minimal. In other words, the difference between the approximation offered by the implementation i and the semantic content of the sentence x_s is an indication of the error level of the implementation. Because we are interested in estimating the overall error of an implementation for each concept, we estimate the error across all sentences as:

$$E_i^c(S) = \sum_{s \in S} |e_i^c(x_s) - y_s^c| \qquad \text{(Equation 3.3.)}$$

where E is the accumulated error over the set of sentences S, of implementation i, and concept c of the domain ontology O. We will consider the output of the information extraction to be $e_i^c(x_s) \in 0, 1$, and the semantic content of the sentence to be $y_s^c \in 0, 1$. So, when there is no difference between the information extractor's output and the semantic content of the sentences (i.e., when $e_i^c(x_s) - y_s^c = 0$), then they are equivalent. This extraction error can easily be extended to the case where the semantic content of a sentence and the output of an information extractor is a relation of the type $y_s^c = R_c(a, b)$. The difference between the two relations can be determined by considering semantic similarity or using some variation of string matching. To keep the description of the selection strategy simple, we have chosen $y_s^c \in 0, 1$.

Because we need to select information extractors that produce the most accurate extraction, the selection strategy minimizes the extraction error. This translates to identifying the implementation i that has the minimal error $E_i^c(S)$:

$$I^c(S) = argmin_i(E_i^c(S)) \qquad \text{(Equation 3.4.)}$$

where $I^c(S)$ is the implementation with minimum error when extracting concept c over the set of sentences S. We can consider that the selection of the most accurate implementation is a function of the concept it extracts given the sentences observed. So, we will restate $I^c(S)$ as $I(c, S)$.

To extend the selection of information extractors to all concepts, we pick the information extractors for each concept with implementation E_i^c:

$$\bigcup_{\forall c \in O^{\Delta}} e_{I(c,d)}(x_s) \qquad \text{(Equation 3.5.)}$$

The implementation that has the minimal amount of errors will be selected as part of the OBIE. This selection leads to having a hybrid OBIE system because, for concepts $c, c' \in O$, their information extractors can have the same or different implementations.

In general, OBIE systems perform this same selection process, but implicitly, and at the system level. An OBIE designer will select the implementation strategy that leads to a minimum set of errors by the system. Because our approach does the selection at the concept level, the error of each information extractor is minimized, which leads to a smaller total error.

3.2.2. Integration Strategy

Integration strategy intends to combine outputs of different extractors to improve the OBIE process. The integration strategy is inspired by Wimalasuriya and Dou's approach of mapping information extractors from concepts of different ontologies for OBIE (MOBIE) [12]. In MOBIE, if two concepts of different ontologies are mapped as equivalent, the concepts' information extractors' outputs are combined into one set.

Our integration strategy comes as an answer to the cases wherein it is difficult to select one type of implementation because their performances are very similar. When the level of accuracy between two implementations of information extractors is close, the difference in performance can be originated by how the documents were selected for evaluation. This performance improvement can be obtained by considering the extraction process as an ensemble method. In machine learning, ensemble methods

use multiple learning algorithms to obtain a better performance than any of the individual methods that confirm the ensemble [64]. There are different types of ensemble methods (e.g., boosting, bagging), which follow different mechanisms (e.g., manipulating training set, voting of different classifiers) to produce the best output.

However, given the constraints of integrating information extractors, most ensemble methods are not suited for integrating information extractors. Voting is an ensemble method that considers the output of each of the methods that are part of the system as a vote. This approach requires an odd number of voting participants or to have votes with different importance (i.e., some votes have higher importance than others) to avoid drawing. In our case, voting does not seem to be a good option because there are only two voters (i.e., two types of implementation for an information extractor), and there is not a clear way to determinate which of the methods is more important (i.e., weighted voting system). A different ensemble approach consists of altering the training of the underlying methods, such as in the case of bagging and boosting. In the case of bagging, each one of the underlying methods uses a randomly selected subset of the training data set to learn a model. Once the models are learned, their outputs are combined as an average or through voting. In the case of boosting, the ensemble learns iteratively by training new models on instances that previous learners misclassified. Because we cannot affect the design process of extraction rules by applying some strategies to the training set (in contrast to machine learning), neither bagging nor boosting are an option as an integration strategy.

For this work, we have selected stacking. Also known as stacked generalization, it consists of training a model (i.e., top-level classifier) that uses as input the predictions of several other methods (i.e., bottom-level classifiers). On other words, the set of outputs of the bottom level classifiers create instances, which are passed to the top-

level classifier. The top-level classifier finally produces a single output. Stacking can be used as an integration strategy, since it can use the output of both types of implementations as input for a top-level classifier. In most cases, stacking uses linear regression as a top-level classifier with a set of meta-features and first-level classifier outputs [65]. The input for the top-level classifier will be:

$$\langle e^c_{ML}(x_s), e^c_{ER}(x_s), y^c_s \rangle \qquad \text{(Equation 3.6.)}$$

where ML corresponds to the machine learning-based implementation, ER corresponds to the extraction rule implementation, and y^c_s is the semantic content of sentence x_s given concept c. In our case, because it is not clear what elements of a sentence can be used as meta-features, linear regression does not perform as well as Naive Bayes or decision trees. For this current work, we have selected Naive Bayes as the top-level classifier.

3.3. Evaluation

In the following section, we provide details regarding the evaluation of our proposed hybrid OBIE system. We have evaluated the effectiveness of our approach with two different datasets.

3.3.1. Study Case: MUC4 Dataset

3.3.1.1. Dataset and Ontology

The *MUC4* dataset we have selected for evaluation is the one presented by Wimalasuriya and Dou in their multiple ontology approach to OBIE [12] and their proposed OBCIE architecture [13]. This dataset is a subset consisting of 200 of

the 1700 documents of the original dataset from the 4th Message Understanding Conference. The documents of this dataset are news articles on terrorist activities in countries in Latin America during the late 1980s and early 1990s.

The dataset also contains *keys* that are a description of details of terrorist activities, such as the instrument used in the activity, the location the activity occurred, and the target of the activity. Each news article can have multiple keys associated with the article that can indicate different types of details or different details of the same type. As a whole, these keys provide a gold standard of relevant content for the dataset.

We have used the ontology constructed by Wimalasuriya and Dou multiple ontology approach (Figure 3.1). The ontology is based on the structure offered by the keys of the MUC4 dataset (Table 3.1). Since the *keys* indicate details such as the instrument used in a terrorist-related activity, they can offer a classification of entities (e.g., person) described by the documents of the dataset.

3.3.1.2. Implementation Details

To implement both our proposed hybrid approaches and the comparison methods, we have used following the general approach.

In the case of rule-based extractors, we have randomly selected a subset of documents to be used to generate extraction patterns for each ontological element to be extracted. We have selected 30% of the dataset, which corresponds to 60

Element type	Number of element
Concepts	15
Relationships	6
Subclass relationships	6

TABLE 3.1. Statistical information about the ontology.

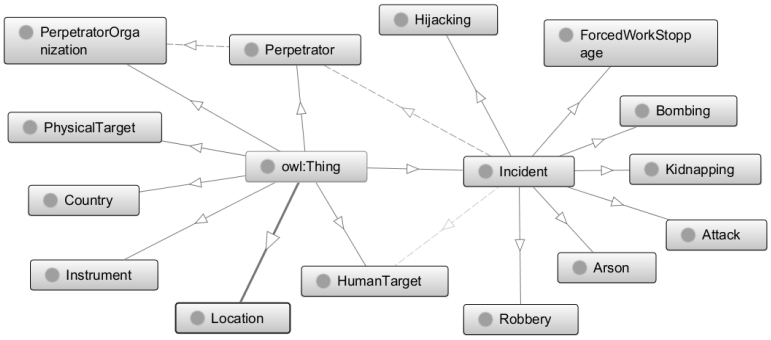

FIGURE 3.1. Graphical representation of a section of the ontology associated to the MUC4 dataset.

documents, to define the patterns for the rule-based extractors for the different ontological elements to be extracted. Since the dataset is not large, this subset should offer a reasonable sample of sentences representing the sought concepts and relationships.

In the case of machine learning-based extractors, we have selected a large subset of documents to be used for training the machine learning methods. Similar to Wimalasuriya and Dou machine learning-based implementation [13], we have selected 80% of the dataset (i.e., 160 documents) for training, while the rest is use for testing. We incorporated the two-phase extraction approach previously described (Section 3.1.2), which is also seen in Kylin [41, 55] and in the study case of OBCIE [13]. In the first phase, we try to determine if a sentence contains the sought ontological element (e.g., relationships) through a binary classifier. One class corresponds to the sentences that carry the information while the other class corresponds to sentences that do not have the information. In this phase, sentences are transformed into vectors. The features of the vectors correspond to ontological metadata of the concepts or relationships to extract (as defined in OBCIE): keywords, part-of-speech

labels, and WordNet synsets (i.e., sets of synonyms) [66]. For example, the metadata for Myosin has keywords such as stuck, stay, get, and binding, while it has as synset the words stick and releases. For this phase, we use a Nave Bayes classifier, which is a popular option for text classification because of its simplicity and good general performance [67]. In the second phase, we determine the part of the sentence (i.e., words) that represents the information. A probabilistic model (in our case Conditional Random Fields [68]) determines if the sequence of words corresponds to the sought information or not. This phase uses the sentence's original metadata information used in the first phase, plus the output of the previous phase classifier. It is possible to have a large number of information extractors based on different machine learning methods, such as Support Vector Machine [69] or Maximum Entropy [41, 55]. We have selected Naive Bayes and CRF as the methods for the machine learning implementation strategy because they have shown consistent and accurate results in IE [17, 18, 41, 55].

3.3.1.3. Comparison Methods

We will compare our proposed combination methods, selection (*MinError*) and integration (*StackNB*), with single implementation systems. The single implementation approach is when the implementation strategy is considered as a guideline for the entire IE system. In other words, we will compare our proposed methods to a system with all information extractors are implemented as rule-based, and we will compare against a system with all information extractors implemented based on machine learning.

We will also consider results obtained by Wimalasuriya and Dou when they evaluated their OBCIE platform [13]. Although used to evaluate a machine learning-based implementation of OBCIE, the implementation details such as the selection of

features and training sets can offer an alternative view of how an extraction system might behave when applied this dataset.

3.3.1.4. Evaluation Metric

The performance of an IE system is measured with the metrics Precision, Recall, and F1 measure. While Precision measures how much of the extraction is correct, Recall measures how complete is the extraction. The F1 measure is the average between Precision and Recall that provides an overall measure of the IE system. To estimate these measures, a golden standard must be established. In the present study, the golden standard is defined as all possible instances that the extraction rules should detect (or extract) when they are applied to the summaries.

Precision is calculated by dividing the number of correct extractions or *true positives* (tp) by the total number of extractions, which are *true positives* plus *false positives* ($tp + fp$).

$$P = \frac{tp}{tp + fp}$$

Recall is calculated by dividing the number of correct extractions (tp) by the total number of instances that should be extracted, which are *true positives* plus *false negative* ($tp + fn$).

$$R = \frac{tp}{tp + fn}$$

F1 measure is the simplest and most commonly used form of F-measure in OBIE, which provides an average of Precision and Recall.

$$F1 = \frac{2 \cdot P \cdot R}{P + R}$$

Metric	MinError	StackNB	OBCIE	Rule-based	ML-based
Precision	52.2%	55.6%	23.6%	44.9%	26.7%
Recall	55.1%	59.6%	42.7%	60.0 %	36.4%
F1	53.6%	56.2%	30.4%	57.6%	30.8%

TABLE 3.2. Performance of extraction by our proposed methods of selection (*MinError*) and integration (*StackNB*), the method by Wimalasuriya and Dou (OBCIE), rule-based extraction, and machine learning-based implementation.

These traditional evaluation metrics do not consider the semantic relation between domain elements when evaluating the correctness and completeness of the extraction process [9]. An extraction (or label) is either correct or incorrect. Metrics such as Balanced Distance Metric (BDM) [70] and Learning Accuracy (LA) [71] take into account the similarity between the correct extraction and the system's output. Both metrics evaluate an extraction based on its semantic distance in the ontology's structure to the correct extraction. For example, if there is a subclass relationship between two concepts, they are considered to be close.

3.3.1.5. Results

Table 3.2 presents the performance results of our proposed hybrid methods selection (*MinError*) and integration (*StackNB*), the method by Wimalasuriya and Dou (OBCIE), rule-based extraction, and the single implementation approaches (rule-based and machine learning-based).

We can see clearly that both of our hybrid based methods provide a more accurate performance. This difference is more notable than in the case of our previous dataset (Section 3.3.) because MUC4 is more suitable of rule-based extraction that machine learning-based extraction. Because our methods evaluate the dataset, it can provide a more smart selection of implementation, leading to better results.

It must be mention that it is possible that labels (keys) in the dataset are incomplete. When revising the dataset, it became evident that for some type of keys (e.g., Perpetrator) not all instances are indicated. This issue must have affected both implementation strategies, but machine learning is more sensible to this problem.

Finally, as seen in Section 3.3., our proposed hybrid approach can produce a more accurate extraction than single implementation strategies, such as rule-based and machine learning-based extraction.

3.3.2. Study Case: Cell Biology Dataset

3.3.2.1. Data

Original Data Set The original data set corresponds to students' answers to an exam from an undergraduate biology class. From the biology exam, we have selected one question that requires the students to present a short, justified answer. Following is the selected question:

> *If you generate a mutation that breaks down the electron transfer chain in mitochondria, will myosin proteins fall off microfilaments or get stuck to it? Why?*

Each answer is a short paragraph that consists of at most four sentences: the answer to the question followed by a short justification. For the answer to be correct, the paragraph must mention specific relations between four concepts: myosin, adenosine triphosphate (ATP), adenosine diphosphate (ADP), and electron transport chain (ETC). An example of a correct answer:

An answer is considered incorrect if the answer sentence is incorrect, or the justification is incorrect. An example of an incorrect answer:

They will tend to get stuck because the exchange of ATP for ADP causes the myosin head to release the microfilament. If the ETC is halted, ATP will no longer be produced.

They will fall off. This is because a mutation in the ETC will cause an absence of ATP.

The answers have been labelled by domain experts (the instructor of the class and his teaching assistants) indicating whether they are correct or incorrect and whether the answers provide enough justification.

The nature of the text (i.e., student answers to an exam) has led to the data set being less diverse, in terms of sentence structure and vocabulary, than other data sets in IE. Because the documents of the data set are answers from an exam, it is more likely that students will focus on content rather than the style of their answer. On the other hand, the answers are focused on a very specific set of concepts and relationships of the domain. For the text to be an effective answer, the text must refer to concepts and relationships relevant to the questions.

Synthetic Data Set In order to evaluate our proposed extensions, there are some requirements that the data set must meet. Although the original set of students' answers is sufficient to other IE implementation approaches, the proposed combining strategies for multiple implementations require a larger data set. For both combining strategies, the data set needs to be large enough to allow three subsets: a first set for training and designing the information extractors, a second set that is used for initial evaluation by the selection strategy and for top-level training by the integration strategy, and a third set for a final evaluation of the system (i.e., testing). To evaluate both extensions, we have constructed from the original data set a synthetic data set.

Concept	Number of correct sentences	Number of correct sentences
ATP	7	15
ADP	3	11
ETC	8	21
Myosin	12	28

TABLE 3.3. Number of template sentences for each concept.

As previously mentioned, the correct answer to the exam's question can be constructed by combining sentences that reference the relationships among four concepts. The statement that provides the answer to the question is a property of Myosin. The justification of the answer comes from a combination of properties of ETC, ATP, and ADP. Therefore, to produce an answer that meets content requirements, we need to create a paragraph that contains a statement from each of the mentioned concepts. To provide diversity in synthetic answers, we created a template set of correct sentences for each concept. We have also created a template set of incorrect sentences for each concept. In general, the sets of incorrect sentences are much larger than the sets of correct statements, because the incorrectness of a sentence can be caused by multiple factors, such as an incorrect relation between a pair of concepts or a contradiction of a logical constraint. Both correct and incorrect sets of sentences for each concept contain sentences from the original data set, plus sentences created based on domain knowledge.

A synthetic data set is generated by creating a number of answers with a probability of having erroneous sentences. An answer from the synthetic data set is created by first selecting a correct or incorrect sentence of a concept, based on the probability of erroneous sentences in the data set (Table 3.3). The correctness of the sentence for each concept is determined independently. Once the correctness of the sentence has been determined, the actual sentence that will be included in the

answer is selected from the set of correct (or incorrect) sentences for the concept. For example, for the concept ATP we can select one of seven correct sentences and one of 15 incorrect sentences.

3.3.2.2. Ontology

Currently, there are a large number of biology-related ontologies that are available. Through the National Center for Biomedical Ontology's BioPortal website, it is possible to access more than 300 biomedical ontologies. By searching in BioPortal, it is possible to identify eight ontologies (e.g., BioModels Ontology, CRISP 2006 Thesaurus) that contain the concepts (e.g., myosin and ATP) which are required to analyze the students' answers. However, these ontologies do not offer all of the necessary relationships that are required to analyze the students' answers. This difference originates because many ontologies are created with the purpose of providing a hierarchical classification of entities from the domain knowledge (i.e., taxonomy).

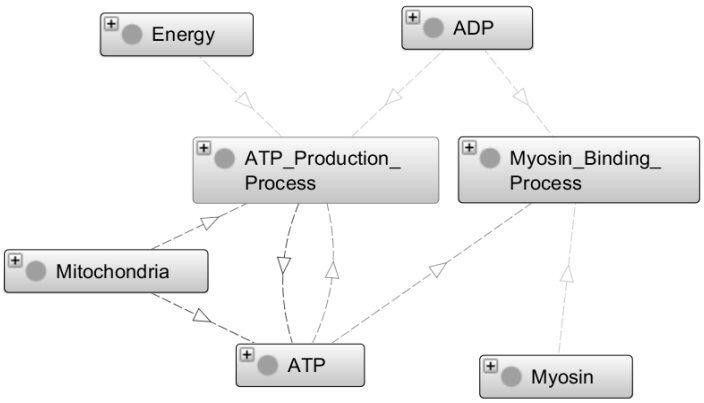

FIGURE 3.2. Graphical representation of a section of the ontology development for this work.

Element type	Number of element
Concepts	17
Relationships	10
Subclass relationships	3

TABLE 3.4. Statistical information about the ontology.

For this work, we have designed an application-driven ontology (Figure 3.2). Although we could have opted to extend one of the available ontologies with the relationships needed to analyze the answers, the construction of an ontology was significantly simpler when considering the logical consistency and complexity of the domain ontology. For the construction of the ontology, we have followed two main guidelines: it must contain all concepts and relationships that will allow for answering the exam's question, and it must not include any other concepts that are not required to answer the question. The first requirement intends to provide the sufficient domain knowledge to analyze the arguments of the answer, i.e., why the myosin is affected by mitochondrial defect. The second requirement tries to reduce the complexity of the ontology by keeping its focus on the part of the domain that is relevant to the task. These criteria lead to an ontology that is highly connected, although it has a small number of hierarchical relationships between concepts.

Based on the mentioned guidelines, we focus the ontology around the four main concepts that need to be stated in an answer for it to be correct. These concepts mostly have cause-effect (i.e., process) type relationships. For example, ETC presence affects the production of ATP, or ADP affects the binding process of Myosin. Because ontologies usually represent domain knowledge by classifying concepts (taxonomy) and properties, process or cause-effect relationships can be difficult to define. We represented these process-type relations as intermediary concepts, e.g., Myosin Binding Process in Figure 3.2. These intermediary concepts

have led the ontology having a rather sparse structure, with few concepts in an ISA (i.e., subclass) relationship (Table 3.4).

3.3.2.3. Implementation Details

In general, the creation of individual information extractors mostly follows the same considerations for single implementation (i.e., traditional OBIE), for multiple implementations [53], and for our proposed combination strategies. In other words, all extraction approaches, both the ones we propose (selection and integration) and the comparison methods, are based on rule-based extractors and machine learning-based extractors.

In the case of extraction rules, we have randomly selected a small subset of instances to be used as examples. The examples are used to identify patterns that can perform the extraction of a specific concept. We have considered the 20% of the corpus to be used as examples for each concept. Since the complete data set consists of 1000 synthetic answers, the number of examples for identifying patterns for each concept is approximately 200 instances. This allowed having a good insight of instances that could be expected for each concept while still being manageable. The following extraction rule identifies the consequence of the break down of the electron transfer chain (ETC):

```
$_ =~ /(It|Myson).+(((stay|get) stuck)|(bind))/i
```

Since the statement answers the question (if it breaks down the electron transfer chain, the myosin gets stuck), a good portion of the answer references the concept Myosin implicitly. This co-reference (i.e., It) was the only one observed in the data, which made it significantly simpler to define in a pattern. The following extraction rule identifies the effect of reduction of ATP, if ETC is broken:

```
$_ =~ /(ETC(|s)).+(stops|loss|less|required).+(ATP)/i
```

In the case of the machine learning-based information extractors, we randomly defined a training set (consisting of 65% of the data set), and a testing set (35% left from the data set). We have used the two-phase approach, described in Section 3.3.1., where a classifier will identify relevant sentences while a probabilistic graphical model determines the part of the sentences that refers to the sought information.

While all information extractors use the same implementation approach (as previously described), our proposed combination strategies use the data in a slightly different way. We divide the data set into three groups: a training set, first stage testing set, and second stage testing set. We define the information extractors with the training set, using 50% of the instances for the machine learning-based extractor and a 20% of instances for the extraction rules. The first stage testing set is used to evaluate and select the best set of extractors in the selection strategy, while the integration strategy is for training the second level classifier. The first stage testing set consists of 25% of the synthetic data set. Finally, the second stage testing set is for evaluating the combined strategy.

3.3.2.4. Comparison Methods and Metrics

We will compare our proposed combination methods, selection (*MinError*) and integration (*StackNB*), with single implementation systems and multiple implementation systems. The single implementation approach is when the implementation strategy is considered as a guideline for the entire IE system. Multiple implementation systems have information extractors implemented as extraction rules and machine learning-based extractors for each concept [53]. For this experiment, there are four concepts and two types of implementations; we have identified

five straightforward configurations of information extractors that the OBIE system can use. Two of the five configurations are equivalent to single implementation systems (i.e., pure configurations). There also are three hybrid configurations: using three machine learning extractors and one extraction rule (3ML-1ER), using two machine learning extractors and two extraction rules (2ML-2ER), and using one machine learning extractor with three extraction rule extractors (1ML-3ER). When considering one mixed configuration, it is possible to define multiple types of settings. For example, in the case of using three machine learning extractors and one extraction rule (3ML-1ER), we can choose an extraction rule implementation for any one of the four concepts and use machine learning extractors for the rest. This has led us to create 8 information extractors by combining all four possible concepts (Myosin, ETC, ATP, ADP), and two implementations (i.e., machine learning and extraction rules).

As in previous section (Section 3.3.1.), we will use traditional IE metrics to evaluate the performance of the different compared approaches. *Precision* determines the correctness of the extraction while *recall* determines the completeness of the extraction. *F1 measure* offers the harmonic mean between precision and recall.

3.3.2.5. Results

In this section, we present and discuss the results of the evaluation of our proposed combination methods, selection (*MinError*) and integration (*StackNB*). The results are presented in detail (Figure 3.3) with respect to the amount of errors in the data set, which provides an insight into how errors can affect the extraction process.

The combined methods obtain, in general, better performance than both the pure methods and the mixed methods which do not have any combination strategy. However, both combined strategies depend on the quality of the extraction performed

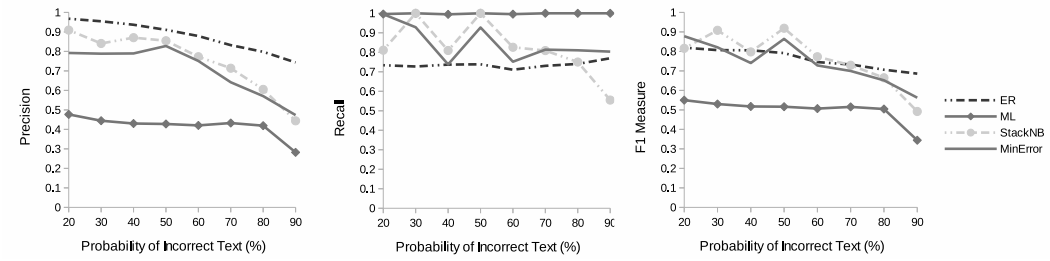

FIGURE 3.3. Precision, recall and F1 measure for information extractors under different levels of error in text, with single implementation (ER and ML), and multiple implementations with our proposed combination strategies (MinError and StackNB)

by extraction rule and machine learning-based extractors. This dependency is more obvious in the case of integration strategy (*StackNB*), where if one of the underlying extractors has a low accuracy, it can significantly affect the performance of the whole process.

We also provide a general view of the experimental results, which allows a more accessible comparison between methods. To keep the analysis clear, we present the average performance of each configuration setting. We also include the performance of the best and worst setting of each concept). With these three values (best, average, and worst), it is possible to get a reasonable understanding of the performance behavior of a configuration.

We see (Figure 3.4) that although the combined strategies outperform the other methods in the case of best performance, their average performance is close (precision) if not worse (recall) than single stage approaches. Because machine learning-based extractors over-extract (i.e., extract more than the actual instance), they have a low precision but a perfect recall. This behavior affects the combined strategies in different ways when integrated with extraction rule performance. In the case of recall machine learning dominates.

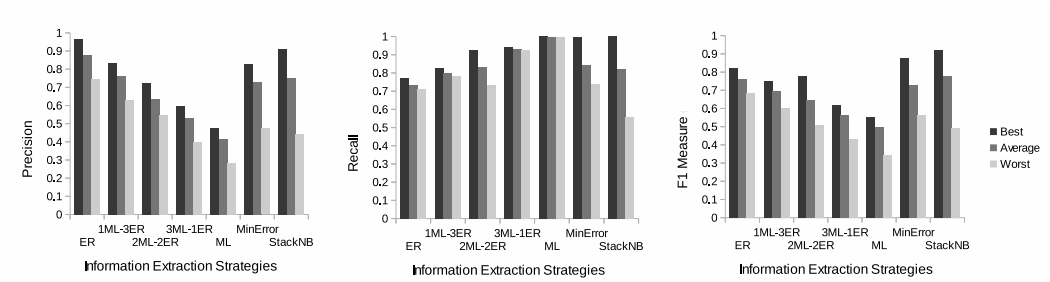

FIGURE 3.4. Precision, recall and F1 measure for information extractors with single implementation (ER and ML), multiple implementations without a combination strategy (1ML-3ER, 2ML-2ER and 3ML-1ER), and multiple implementations with our proposed combination strategies (MinError and StackNB)

From Figure 3.4, one might conclude that our proposed combination strategies are sensitive to the performance of the underlying implementations, the performance of the worse implementation seems to dominate.

Finally, we can see a clear impact to the extraction process of sentences that represent incorrect facts of the domain (i.e., semantic incorrect statements). From these results we cans see a need for a mechanism to identify these types of statements.

CHAPTER IV

PRECOMPUTING SEMANTIC INCORRECTNESS

This chapter consists of work publish in the "Proceedings of the 21st ACM international conference on information and knowledge management" in 2012 [1], and in"Journal of Information Science" in 2015 [63]. Dr. Dejing Dou, Dr. Stephen Fickas contributed in the design of the method propose in this chapter. Dr. Gina Griffiths contributed with the students' dataset while Dr. Hui Zong contributed with the biology exam answers dataset. Adam Martini contributed with the processing of the dataset.

The second contribution of this work is new extraction functionality for OBIE. Traditionally, IE, and by extension OBIE have performed the functionality of extracting information from sentences that express correct content. When we categorize text as *correct content*, we mean that the sentences form a statement that *agrees* with the domain knowledge, i.e., that is consistent with respect to the domain. By contrast, a text with *incorrect content*, i.e., semantically erroneous text, contradicts the domain. Considering that we have defined incorrect text as a false or contradicting statement, it is reasonable to consider logic as a mechanism to identify it.

However, the information contained in the text, itself, is not a sufficient basis for evaluating whether or not it is incorrect. We need to know facts (i.e., true statements) about the domain to verify if whether a sentence from the text is false or not. The domain knowledge, represented through an ontology, can provide us with the frame of *what is correct* within the domain. Therefore, combining this correct knowledge frame with logic, we resolve the correctness (or incorrectness) of a text's content.

We propose *precomputed semantic error detection* as a mechanism to incorporate into OBIE the functionality of extracting semantically erroneous information from text. In OBIE the domain ontology guides the extraction process by encoding ontological axioms into information extractors. Each information extractor is bound to an ontological element (e.g., a concept), and it extracts *in-text* references about this ontological element. However, because an ontology only represents true knowledge about the domain, we need a mechanism to determine or generate domain-inconsistent axioms that can guide the information extractors for semantic error detection. We have proposed a heuristic method, based on an ontology debugging technique, which can generate the domain-inconsistent axioms (precompute), that will be encoded later into information extractors.

Precomputed semantic error detection works in two steps: determining inconsistent axioms (i.e., precomputing), and extracting statements based on the incorrect axioms. In the following sections we provide more details regarding each step.

4.1. Determining Sentence Types

Based on their relationship with the domain, we have identified three types of sentences [1]: correct sentences, incorrect sentences, and unknown sentences (or incomplete).

4.1.1. Correct Sentence

A sentence is consider semantically correct if it is consistent with the domain knowledge. So, any sentence that expresses an aspect of the ontology, such as a

relationship between concepts, is correct. This definition also extends to subsumed axioms. Let us consider the following example:

$$\begin{array}{ll} \text{Ontology} & Producer \equiv Autotroph \\ & Producer \sqsubseteq \exists produce.Food \\ \\ \text{Axiom 1:} & Producer \equiv Autotroph \\ \text{Axiom 2:} & Autotroph \sqsubseteq \exists produce.Food \end{array}$$

In Biology, a *Producer* is an organism that can produce their own food. They are also known as *Autotrophs*. In the example, Axiom 1 is consistent with the ontology because it makes direct reference of a definition of the domain, i.e., *Autotrophs* and *Producers* are the same. Axiom 2 is an infered fact from the ontology. Both axioms in the example are consistent with the ontology, making them semantically correct.

As mentioned in Section 2.2.1., OBIE extracts information from sentences that mention ontological elements. Because an ontology can have a large number of explicit and implicit (i.e., infered) axioms, only a subset of the ontology is selected to be used in a OBIE system.

4.1.2. Incorrect Sentence

A natural consequence of the definition of semantic correct statement is the definition of incorrect statement. We consider a sentence to be semantically incorrect if it is a logical contradiction of some aspect of the domain ontology (i.e., domain-inconsistency). However, an ontology only contains correct facts of the domain it represents. We need a mechanism to determine axioms that are inconsistent with respect to the domain ontology.

We have proposed a mechanism to determine axioms that are inconsistent with respect to the domain based on the heuristic-based ontology debugging approach

seen in Wang et al. [72]. As mentioned in Section 2.1.3., in ontology debugging, research is focused on identifying the origin of inconsistency in an ontology. Wang et al.'s approach looks for specific types of inconsistencies. They have identified a set of common errors that are committed in the process of constructing an ontology. Wang et al. have encoded these common errors into a set of pattern-based rules that can identify inconsistency. Following the approach of Wang et al., it is possible to determine a set of axioms that, if included in the domain ontology, would make the ontology inconsistent. We use Wang et al.s heuristic as a generating mechanism to define domain-inconsistent axioms.

$$\begin{aligned} \text{Ontology} \quad & Producer \sqsubseteq \exists produce.Food \\ & Producer \sqsubseteq \neg Carnivore \end{aligned}$$

$$\text{Axiom} \quad Carnivore \sqsubseteq \exists produce.Food$$

In the example, because of the disjointness between the concepts *Producer* and *Carnivore* (i.e., $Producer \sqsubseteq \neg Carnivore$), they cannot share subclasses or individuals. By establishing as domain of a relationship a concept that is disjoint with the original domain, such as "Carnivores produce their food" (i.e., $Carnivore \sqsubseteq \exists produce.Food$), we are generating a domain-inconsistent axiom. Although the example seems to be a very simple case of inconsistency, sentences that represent these types of inconsistent axioms are not so unlikely [1, 53].

Although this mechanism cannot generate all possible domain-inconsistent axioms (incomplete), it still can generate a large set of axioms for semantic error detection. To understand the domain and type of text to be analyzed becomes a fundamental aspect to help produce the necessary domain-inconsistent axioms. This understanding can lead to an effective and mostly complete analysis of the text.

4.1.3. Unknown Sentence

Finally, we classify as semantically unknown those sentences that are neither correct or incorrect. This type of sentence reflect the *open world* assumption that ontologies follow where we cannot determine the truth value of undefined elements. If the sentence contains elements that are not defined in the domain ontology, we cannot determine if it contradicts the domain, or not.

$$\text{Ontology} \quad Producer \sqsubseteq \exists produce.Food$$

$$\text{Axiom} \quad Tree \sqsubseteq \exists produce.Food$$

In the example, the sentence "Trees produce their food" (i.e., $Tree \sqsubseteq \exists produce.Food$) is incomplete because it is indicating a property of the concept $Tree$ that is not defined in the ontology. Although we know that *Trees* are *producers*, making the sentence true, the ontology does not have an axiom to indicate that Tree is a Producer (i.e., $Tree \sqsubseteq Producer$).

In practice, semantically unknown information extractors are difficult to encode. In most cases, they can be a vocabulary checker that determines if a sentence contains unknown elements. However, this approach tends to be vague, leading to some overlap with extractors for incorrect sentences. Because of these difficulties, we have not included unknown extraction as functionality.

4.2. Extracting Sementics Errors

After determining the set of domain-consistent and domain-inconsistent axioms, for the extraction of correct and incorrect sentences respectively, we need to encode the axioms into information extractors. To integrate our precomputed semantic error

detection into OBIE, we extend the definition of information extractor to include a new aspect, which is orthogonal to both ontology and implementation [53]:

$$e_i^{cf}(x_s) \qquad \text{(Equation 4.1.)}$$

Each information extractor (e) encodes an ontological concept or property (c), following a correct or incorrect functionality (f), under a rule-based or machine learning-based implementation (i). Because the semantic correctness of a sentence is based on its logical relation (e.g., logic contradiction) with the domain and is not affected by other statements, the inclusion of the functionality dimension to information extractors does not affect the modularity of OBCIE. Just as we can include information extractors with different implementations into the same OBIE system, we can also include information extractors with different functionality into the same system.

4.2.1. Rule-based Information Extractor

We first proposed our precomputed semantic error detection approach implemented as extraction rules [1]. Extraction rules use patterns to capture elements from text. The patterns can be built over specific words, part-of-speech, or other linguistic features (e.g., dependency relations). Extraction rules can be simple to implement, and they are mostly an accurate extraction method. For our precomputed error detection approach, extraction rules are based on keywords (e.g., concepts), with each extraction rule representing one axiom (or domain-inconsistent axiom) of the ontology.

In the example, Axiom 1 is a domain-consistent axiom inferred from the domain ontology from the second (*Professor* teaches *Student*) and third statements in the

Ontology	$Student \sqsubseteq \neg Professor$
	$Professor \sqsubseteq \exists teaches.Student$
	$UndergradStudent \sqsubseteq Student$
Axiom 1	$Professor \sqsubseteq \exists teaches.UndergradStudent$
Axiom 2	$Professor \sqsubseteq \exists teaches.Professor$

ontology. Because it shares a similar *textual representation* as one of the ontology axioms ($Professor \sqsubseteq \exists teaches.Student$), we can encode both axioms as a rule-based extractor by the following regular expression (in Perl):

```
$_ =~ /(P|p)rof(\.|\w+) teach(|es) ((\w+) student|) (\w+)/
```

On the other hand, Axiom 2 is a domain-inconsistent axiom because it states that the range of the relationship *teach* is a concept (*Professor*) that is disjoint with the range defined in the domain. The encoding of domain-inconsistent axioms as rule-based extractors is the same as with the domain-consistent axioms. Axiom 2 can represented as the regular expression (in Perl):

```
$_ =~ /(P|p)rof(\.|\w+) (\w+) teach(|es) /(P|p)rof(\.|\w+) (\w+)/
```

4.2.2. Machine Learning-based Information Extractors

As seen in Section 2.2.2. there is a wide variety of possible methods that can be used for implementing OBIE. However, because precomputed semantic error detection generates a large number of domain-inconsistent axioms from a small set of consistent axioms, the method used cannot rely on a large training set. We have consider as machine learning-based implementation a two-phase classification scheme [13, 55].

In the first phase, the method identifies which sentences from the document contain the information the extractor seeks. The process is defined as a binary classification task (Naive Bayes [67]), where one class corresponds to sentences that

carry the information and the other class corresponds to sentences that do not have the information. In this phase, sentences are transformed into vectors. The features of the vectors correspond to ontological metadata of the concepts or relationships to extract (as defined in OBCIE): keywords, part-of-speech labels, and *WordNet synsets* (i.e., sets of synonyms) [66].

The second phase of the platform identifies the elements of the sentence (words) that contain the information. This is done by a probabilistic model (Conditional Random Fields [68]). This phase uses metadata information from the first phase, the output of the previous phase classifier, and a group of extra features that are proposed and used by the *Kylin* system (e.g., capitalized words) [55].

It is possible to have a large number of information extractors based on different machine learning methods, such as Support Vector Machine [69] or Maximum Entropy [41, 55]. We have selected Naive Bayes and CRF as the methods for the machine learning implementation strategy because they have shown consistent and accurate results in IE [13, 17, 41, 55, 73].

4.2.3. Hybrid Extraction

We have also constructed information extractors following the hybrid implementation approaches described in Chapter III. Because functionality is an *orthogonal aspect* to implementation for an information extractor, it can be easily integrated into the combination strategies for hybrid implementation extraction.

To include the semantic error detection functionality to the *selection* strategy, we need to extend the original method which minimize both the extraction error of each concept to minimizing the extraction error for each concept and the functionality. By

incorporating functionality into the selection strategy for hybrid implementation, we simple perform the process for a large number of axioms.

To include semantic error detection functionality for *integration* strategy, the process of integrating outputs is as simple as for the *selection* strategy. For each concept c and functionality f, a top level classifier uses as input the outputs of the rule-based ($e^{cf}_{ER}(x_s)$) and machine learning-based ($e^{cf}_{ML}(x_s)$) extractors for the given concept and functionality. In other words, for each concept and functionality there is a stack of extractors.

4.3. Evaluation

We have evaluated the effectiveness of our approach with two different datasets.

4.3.1. Study Case: Ecosystem Dataset

The Ecosystem dataset is one of several datasets that are part of the study by Sohlberg et al. [74]. Because the dataset is small, we used rule-based information extractors. The following sections provide details regarding the data itself, the ontology constructed, and the metrics used for evaluation.

4.3.1.1. Data

In this work we will use a set of summaries collected on an earlier study by Sohlberg et al. [74] that looked at the use of electronic strategies (eStrategies) for reading comprehension of college students. As part of the study, students were asked to provide oral summaries of each article they had read, where each article is roughly 3 pages in length (4 articles were used). The oral summaries were manually transcribed into text form. From the Sohlberg et al.s collection, we will consider for the present

work 18 summaries from the Ecosystems article. The summaries vary in length from a pair of sentences to 60 sentences. A section of a summary from the Ecosystem set can be seen in the following example:

> *In the ecosystem there are different types of animals.*
> *Producers make their own food from the environment.*
> *Consumers eat other consumers and producers.*
> *The producers are plants, trees, algaes.*
> ...

Because the text are originally oral summaries, they slightly differ from written ones (as it can be seen in the previous example). The transcribed summaries contain repeated statements, and in some cases there are contradictions when considering the summary as a whole. However, because we focus on resolving the semantic correctness of the text one sentence at a time, these cohesion issues do not affect our analysis.

The summaries have been preprocessed in order to simplify the extraction process. The preprocessing has been focused on resolving *anaphoras* and *cataphoras* (e.g., pronouns) and on correcting misspellings. The summaries have also been labeled at the sentence level according to the correctness of their content. The labeled summaries have been used as the gold standard for the purpose of evaluation.

4.3.1.2. Ontology

We constructed an application-driven ontology for the domain of Ecosystems. We used the introductory article that the student used for their summaries as the sole guideline for the construction of the ontology. We included into the ontology only explicit facts stated in the article, and we do not include facts from the entire domain of Ecosystems. By keeping our ontology centered on the introductory article,

Element type	Number of element
Concepts	45
Relationships	28
Subclass relationships	7

TABLE 4.1. Statistical information about the ontology.

we intended that the ontology can better cover concepts and relationships from the students summaries, which are also solely based on the article.

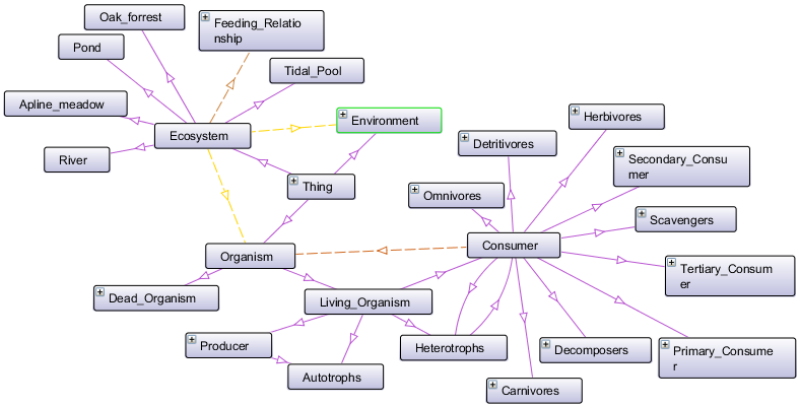

FIGURE 4.1. Graphical representation of a section of the Ecosystems ontology

Because of the strict construction criteria, the ontology has many concepts that do not have a membership relationship with another concept, as well as not having instances (Table 4.1). This is originated by the nature of the Ecosystems article. Because the article is an introduction to the domain, a broad set of concepts and relationships of the topic are presented rather than details, such as specific examples. In Figure 4.1 we presents a graphical representation of a part of the Ecosystems ontology.

4.3.1.3. Evaluation Metrics and Comparison Methods

Just as in Section 3.3.1.4., the metrics used to evaluate are precision, recall, and F1 measure. Although the ontology in this case study has more hight (hierarchical structure) that the ontology used for the hybrid implementation evaluation, this evaluation focus on the functional extraction of semantic errors.

Because error detection is a new functionality for IE, there is no other direct method for comparison. For this reason, we will present evaluation metrics (precision, recall, and F1 measure) separated by functionality, and the comparison will be between functionalities. Although this is not an ideal approach for evaluating an extraction method, it still can provide us with insight into what can be expected in terms of quality of extraction when performing error detection.

However, we can obtain some insight of the new functionality by a indirect method. From study by Sohlberg et al., the summaries were evaluated by an instructor (i.e., gold standard), and by a Latent Semantic Analysis (LSA) evaluation system.

LSA [14] is a method that has become popular in automatic grading systems, such as Laburpen Ebaluaketa Automatikoa (LEA) [75], Intelligent Essay Assessor (IEA) [76], and Knowledge Analysis Technologies (KAT) engine from Summary Street [77]. It treats essays as a matrix of word frequencies and applies singular value decomposition (SVD) to the matrix to find an underlying *semantic space*. It then represents each to-be-graded essay in that space, as a vector, and assesses the cosine similarity between the essay and the graded or standard essays or the text students read. The cosine similarity can be transformed to the grade. Although LSA is not a knowledge based approach to semantic error detection, it provides a method that can determine some level of semantic incorrectness.

Metric	Type of Extraction Rules	
	Correct	Incorrect
Precision	91.9%	97.4%
Recall	83.3%	88.63%
F1	87.4%	92.8%

TABLE 4.2. Performance of IE

Although OBIE does not provide a grade, we can define a type of evaluation metric based on the extracted information from the summaries. We have used the ratio of semantically correct sentences extracted over the total number of sentences contained in the summary, i.e., a *correctness ratio*. We have removed the semantically incorrect sentences before capturing (i.e., extracting) the correct ones. This gives a sense of *how relevant* the content of the summary is with respect to the domain.

4.3.1.4. Results

Table 4.2 provides the performance of the OBIE. In general, the extraction for both functionalities has a high accuracy. This result can be expected from both the type of implementation used and the characteristics of the dataset. Rule-based extraction can have a high precision, specially if there is a set of patterns for the extraction of one concept or relationship (as it was the case for this evaluation). On the other hand, because the summaries were initially provided orally, the vocabulary tends to be smaller than a written document which can be edited and revised before submission.

Because the ranges of grades from human graders, LSA, and our systems (ratio of correctness over complete summary) are totally different, we have only conducted correlation studies among them. We have found that the grades from our OBIE's *correctness ratio* does have a positive correlation with human grading. In other words,

there is agreement between the human grader and OBIE. On the other hand, there is no correlation between the LSA and OBIE's *correctness ratio*. This can be easily seen from Table 4.3, where there is almost no agreement between the methods. It is interesting to note that, for the summaries used in the present work, both the LSA grading and the instructor's grading are not positively correlated. The most straightforward answer is that LSA does not address incorrect statements. Given that we found that 75% of the summaries contained at least one error, the divergence from LSA is not surprising.

	Pearson Correlations		Spearman Correlations	
	Instructor	LSA	Instructor	LSA
LSA	-0.316		-0.163	
Relevance	0.55	-0.068	0.531	0.176
Completeness	0.502	-0.168	0.547	-0.018
Importance	0.488	-0.119	0.559	0.016

TABLE 4.3. Correlation between grading metrics

It is worth looking at an example in the discussion of semantic error detection by OBIE. Summary STIR33 (Table 4.4) has a grade of 7 from the instructor and 0.811 from LSA. The OBIE's *correctness ratio* score is 0.222. OBIE found a number of errors in this summary, including:

1. Detritivores do not eat inorganic matter.

2. Omnivores eat only plants and animals. They do not eat organic waste or fragments of dead organisms.

3. Herbivores eat plants.

It is worth noting that the instructor's score was 17 out of 20 for this summary. We contacted the grader to try to gain insight into her high score given the number

STIR33 Ecosystems are composed of different types of living organisms. There are herbivores, carnivores, detritivores and omnivores. Detritivores eat inorganic matter or non-living matter. Omnivores eat everything. Herbivores eat meat and other organisms. And herbivores eat vegetation.
STIR26 Carnivores are fish. And I figure out what to say in my head.

TABLE 4.4. Example of summaries.

of errors we found. Our note to her prompted her to look at her raw scores again and find a typo - her raw score was 7 not 17.

4.3.2. Study Case: Cell Biology Dataset

The Cell Biology dataset is the same as the one presented in Section 3.3. Following is a short revision of the data, the ontology constructed, and the comparison methods used in the evaluation. A more extensive description of he dataset can be found in Section 3.3.

4.3.2.1. Overview of Data, Ontology and Comparison Methods

The dataset correspond to student answers in the final exam of an undergraduate biology class. The corpus consists of 77 student answers. Each answer is a short paragraph that may contain at most four sentences. The answers have been labeled by domain experts (the instructor of the class and his teaching assistants) indicating if they are correct or incorrect, and if the answers provide enough justification. Because the size of the data is not large enough for the combination strategies, we have create a

set of templates based on the students' answers. We have generated a larger synthetic dataset for the templates.

We have constructed an ontology for the biology domain of the final exam's question. Although there are many biology-related ontologies available (the National Center of Biomedical Ontologys BioPortal1 website offers access to more than 300 biomedical ontologies), they do not offer the necessary relationships that are required to analyze the students answers. To overcome this limitation we have developed our own ontology. To construct the ontology we have followed two main guidelines: it must contain all concepts and relationships that will allow answering the exams question, and it must not include any other concepts that are not required to answer the question. The first requirement intends to provide the sufficient domain knowledge to analyze the arguments of the answer, i.e., why the myosin is affected by a mitochondrial defect. The second requirement tries to reduce the complexity of the ontology by keeping it focus on the part of the domain that is relevant for the task. This criteria leads to an ontology that is highly connected, but has a small number of hierarchical relationships between concepts.

As comparison methods, we have used a set of extraction approaches that are based on rule-based and machine learning-based extraction. We can have single implementation systems and multiple implementation systems. In the single implementation systems, all information extractors are implemented as machine learning-based extractors or as rule-based extractors, while multiple implementation systems have information extractors implemented as extraction rules and machine learning-based extractors for each concept. In this case, there are four concepts which leads into three types of multiple implementation configurations: using three machine learning extractor and one extraction rule (3ML-1ER), using two machine

learning extractors and two extraction rules (2ML-2ER), and using one machine learning extractor with three extraction rule extractors (1ML-3ER).

4.3.2.2. Results

As in Section 3.3., we present the evaluation results in detail with respect to the amount of errors in the dataset. We also offer a general view of the evaluation by presenting the best, the average, and the worse performance of each configuration setting.

In general, as the amount of errors increases (higher probability of error in an answer), the precision of all the methods increments (Figure 4.2). This is the inverse trend of the one observed in Figure 3.3: as the error level increases in the data set, the extraction of semantically correct sentences becomes less precise. In contrast, the completeness (i.e., recall) of the extraction seems not to be affected by the level of error.

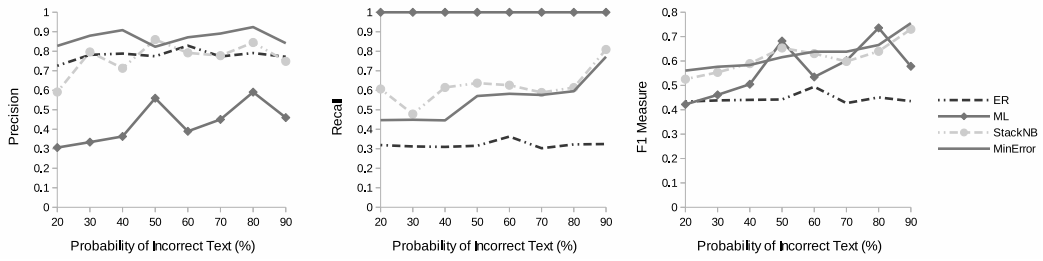

FIGURE 4.2. Precision, recall and F1 measure for information extractors under different levels of error in text, with single implementation (ER and ML), and multiple implementations with our proposed combination strategies (MinError and StackNB) with the functionality of extracting incorrect statements.

Figure 3.4 showed that, when extracting a semantically correct sentence, the performance of in both combination strategies are influenced by the worse implementation. The effect seems to differ from correct statement to error extraction

functionality. In the case of extracting semantically incorrect sentences, our proposed combination strategies for error extraction produce a more averaged performance between the underlying implementations (Figure 4.3).

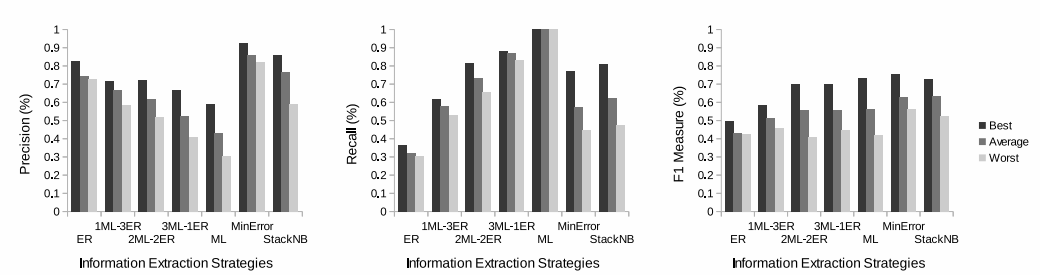

FIGURE 4.3. Precision, recall and F1 measure for information extractors with single implementation (ER and ML), multiple implementations without a combination strategy (1ML-3ER, 2ML-2ER and 3ML-1ER), and multiple implementations with our proposed combination strategies (MinError and StackNB) with the functionality of extracting incorrect statements.

Finally, Figure 4.4 compares the average performance of each configuration given its functionality. In general, information extractors that extract correct statements have a higher precision, recall, and F1 measure than their error extraction part, for any given implementation. This difference in performance is a natural consequence of how facts and errors can be represented in text. For example, we see in Table 3.3 that there are 12 types of correct sentences for Myosin, in contrast to the 28 types of incorrect sentences. The information extractor for incorrect sentences needs to consider more types of cases than an information extractor for correct sentences, which leads to a higher possibility of inaccuracy. This situation is accentuated in the case of machine learning implementation because not all errors are present in the same frequency within the training set. This leads not only to the machine learning-based extractor having to consider a wider range of types, but also that not all available

types are available enough or frequent enough in the training set to be considered relevant.

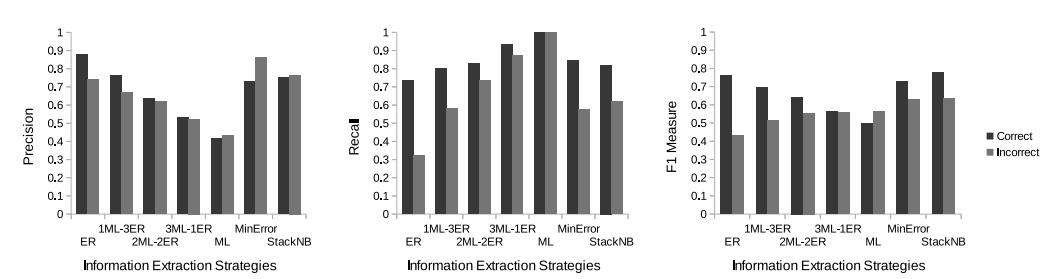

FIGURE 4.4. Comparison of correct statement extraction and error extraction functionality in terms of precision, recall and F1 measure for information extractors with single implementation (ER and ML), multiple implementations without a combination strategy (1ML-3ER, 2ML-2ER, and 3ML-1ER), and multiple implementations with our proposed combination strategies (MinError and StackNB).

Figure 4.4 also shows that the integration strategy *StackNB* performs better for correct statement extraction, while *MinError* slightly outperforms the rest for the error extraction.

CHAPTER V

ONLINE REASONING FOR SEMANTIC ERROR DETECTION

This chapter consists of work publish in the "Proceedings of the 13th international conference on ontologies, databases and application of semantics" in 2014 [78]. Dr. Dejing Dou, Dr. Stephen Fickas contributed in the design of the method propose in this chapter. Dr. Gina Griffiths contributed with the students' dataset.

In the *precomputed semantic error detection* approach described in Chapter IV, domain-incorrect sentences are identified by predefined information extractors. These information extractors encode domain-inconsistent axioms that are generated from the domain ontology. In this way our previous method was able to identify errors (i.e., incorrect statements) which have been previously defined. However, this approach can only recognize semantically incorrect sentences that represent one of the domain-inconsistent axioms, if they were part of the training set or very similar to a sentence in the training set. New sentences can not be judged correctly.

In order to provide the most complete analysis of text content, we propose *online reasoning for semantic error detection*, a method for identifying domain-incorrect content in text by incorporating *online* logic reasoning (i.e., inference) and domain knowledge. Instead of having the ontology guiding the extraction process, the IE is performed based on structural elements from the text, while the semantic error detection comes from determining if the text is logically consistent with respect to the modeled domain knowledge (i.e., ontology).

Our proposed inference-based approach consists of two steps. In the first step, sentences are transformed into logic clauses through a combination of IE and

vocabulary mapping. This step intends to take written natural language (i.e., the sentence) to a formalized representation that is compatible with the domain ontology (i.e., ontological axiom). In the second step, the transformed sentence is included into the ontology to determine its consistency with the domain. This process, which is performed by a reasoner, is known as *consistency checking*. If the domain ontology becomes inconsistent after the extracted sentence is added into it, then the sentence is semantically incorrect with respect to the domain.

We have identified two approaches when analyzing the extracted sentences: single sentence analysis and multiple sentence analysis. In single sentence analysis, we intend to determine the semantic correctness of text by considering one sentence at a time. Under this approach the semantic content of each sentence is considered independent from the rest of the text. In the case of multiple sentence analysis, a group of sentences from the text are analyzed as set of clauses. Although the analysis of multiple sentences leads to a higher computational complexity, it allows us analyze the correctness between sentences. There are cases where sentences can be consistent when considered independently, but become inconsistent when analyzed as a set.

Because we first identify all the relationships in the text, and then we determine their semantic correctness against the whole ontology, it is possible to offer a complete analysis of the text. Although this approach differs from the definition of OBIE [9], we argue that it is still an OBIE process since the approach relies on the domain ontology to determine the correctness of each statement.

5.1. Transforming Text to Logic Clauses

In the first step of our proposed *online reasoning* approach, sentences need to be transformed from their written form into logic clauses which uses the same vocabulary

than the domain ontology. This transformation is achieved by through IE and a mapping mechanism.

5.1.1. Information Extraction

As previously mentioned in Section 2.2.3., there are three main strategies to IE depending on the level of human intervention (i.e., data preparation). However, because our approach intends to determine the correctness of each sentence presented in the text, not all three strategies are suited for our approach. Supervised IE cannot provide a complete extraction from the text since the process is guided by known labeled data and predefined patterns. Similarly, semi-supervised IE systems are guided to extract relationships based on sets of known individuals. Plus, in order to provide quality extraction, semi-supervised IE requires a significant set of training individuals.

For the present work, we have chosen the unsupervised strategy followed by the Open Information Extraction system *OLLIE* [47]. Open Information Extraction systems intend to extract binary relationships without using any training data (or handcraft patterns). The main goal behind this approach is to offer an IE system that can scale to the Web. To do this, Open Information Extraction follows a set of general patterns to extract every possible relationship from a text [17, 46, 73, 79]. In the case of *OLLIE*, the patterns are built by generalizing extractions with high confidence (i.e., high quality extraction). The set of high quality extractions is obtained from Open Information Extraction system *ReVerb* [46], which uses a verb-based patterns to identify relations in text. These extractions (e.g., tuples) have two constraints: they contain solely proper nouns as entities participating in the extracted relation, and they have a high confidence value. Then, similar to semi-supervised IE systems, *OLLIE*

gathers a set of sentences that contain the entities and relations from the extracted tuples. To avoid collecting sentences that might introduce errors, *OLLIE* only gathers sentences with a structure that is centered in the elements of the extracted tuple, i.e., elements of the relation must be in a linear path of at most size four in the dependency parse [47]. From the selected sentences, *OLLIE* learns a set of general extraction patterns. If the structure of a sentence meets a set of requirements (e.g., the relation is in between the two entities in the sentence), a pure syntactic pattern can be learned from the sentence (e.g., most general pattern). If the structure of the sentence does not meet the requirements, lexical aspects of the sentence are considered in order to produce a general pattern. These generalized patterns are used to extract new tuples from text. For example, from the sentence "Scavengers feed from dead organisms," OLLIE will produce the tuple *feed(Scavengers, dead organism)*.

Because we are focused on determining the correctness of the text content, we considered *OLLIE* as a *blackbox* component of our system. This approach to the extraction component of our method allows us change to other unsupervised IE systems, such as *ClausIE* [79] or *ReVerb* [46], in the future without needing to redesign our method.

5.1.2. Mapping Extractions to Ontology

Although the text and the ontology belong to the same domain, it is very possible that the selection of words to represent concepts and relationships might differ. So, to be able to use the domain ontology to evaluate the correctness of the text's semantics, we need to solve first the lexical gap that might exist between the text and the ontology. In other words, we will need a mapping mechanism that can allow us pass

from the vocabulary of the extracted entities and relationships to the vocabulary of the ontology.

Because we are focused on semantic error detection, we have opted for a simple and direct solution for the translation (i.e., vocabulary mapping) task. The mapping mechanism that we proposed is based on two *dictionaries of terms*: one for managing concepts, and another for managing relationships. In the case of the dictionary for managing concepts, an extracted entity will lead to the equivalent ontological concept. For example in Figure 5.1, both *dead organisms* and *dead animals* lead to the concept *Dead Organism*.

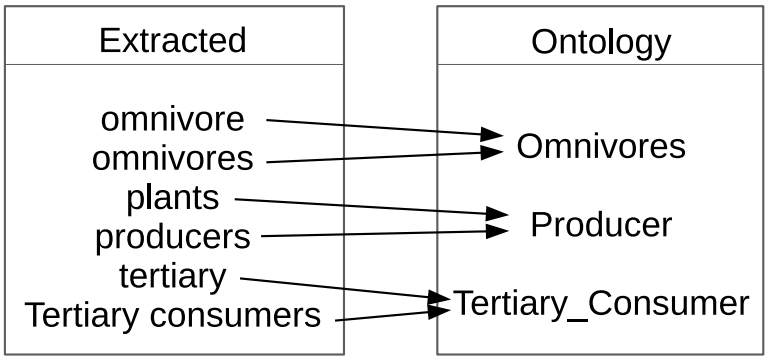

FIGURE 5.1. Example of mapping between extracted terms and ontology concepts.

In the case of managing relationships, because a relationship might have different meaning depending on other elements in the sentence, we consider both subject entity and relation to determine the ontological property. For example, the concept *Carnivores* and the relation *feed* will lead to the property *feed_from_herbivore*, while concept *Herbivore* and relation *feed* will lead to the property *feed_from_producer*. Both dictionaries are generated by considering a subset of extracted relationships (i.e., sample) from the data set.

5.2. Single Sentence Analysis

Once we have extracted all the relations from the text (e.g., "Autotrophs produce their food," to *produce(Autotrophs, food)*), and the relations have been mapped to the vocabulary of the ontology (e.g., *produce(Autotrophs, food)* to $Autotrophs \sqsubseteq \exists produce.Food$), we proceed to analyze the correctness of the sentences by using consistency checking.

As mentioned, we have identified two approaches when analyzing text extractions: single sentence analysis and multiple sentence analysis. In single sentence analysis, we intend to determine the correctness of text by considering one sentence at a time. Under this approach the semantic content of each sentence is considered independent from the rest of the text. In the case of multiple sentence analysis, a group of sentences from the text are analyzed as set of clauses. Although the analysis of multiple sentences leads to a higher computational complexity, it allows us analyze the correctness between sentences.

In this section, we focus on single sentence analysis. Each sentence will be included into the domain ontology independently. After the analysis of the sentence has concluded, the sentence's relationship will be removed from the domain ontology.

However, to be able to determine the semantic correctness of a sentence, we need to consider some requirements for our approach. First, because our online reasoning approach to semantic error detection uses logic reasoning, we need a more strict definition of sentence types. Second, the domain ontology needs to be consistent and complete. In the following sections, we provide details regarding these requirements

5.2.1. Redefining Sentence Types

In Section 4.1., we present a classification of sentences based on their relationship with the domain. Although the original definition is necessary for determining the sentence's type, it is not sufficient when using our *online reasoning* approach. In the following sections, we offer a new definition of sentence types for reasoning-based semantic error detection method.

5.2.1.1. Correct Sentences

In Section 4.1.1., we define a sentence is semantically if it is consistent with respect to the domain. A sentence is consistent if the domain does not provide the sentence to be false. However, although consistency is required, it is not sufficient to prove correctness. Even more, if a sentence is completely unrelated to the domain, it is more likely that the statement will not violate any constraint of the domain. Let us consider the following example:

Ontology	$Planets \sqsubseteq \exists orbits.Stars$
	$orbits(Earth, Sun)$
	$Stars(Sun)$
Axiom 1	$Planets(Earth)$
Axiom 2	$Myosin \sqsubseteq Amino_Acid$

In the example, neither sentence contradicts the domain. We can see that Axiom 1, which states that *Earth* is a *Planet* ($Planets(Earth)$), is logically consistent with the domain ontology because from the domain we know that *Planets* orbit around *Stars* ($Planets \sqsubseteq \exists orbits.Stars$), *Earth* orbits around the *sun* ($orbits(Earth, Sun)$), and the sun is a star ($Stars(Sun)$). Axiom 2, which states that *Myosin* is an *amino acid* ($Myosin \sqsubseteq Amino_Acid$), also is consistent with the domain because it does

not contradict the domain. However, the ontology does not have any information regarding the elements that are being referred in Axiom 2 that would allow us to effectively determine its semantic correctness.

We have revised our definition of semantic correctness. A sentence is semantically correct if it is a logic consequence of the domain, i.e., a semantically correct sentence s can be *entailed* from the domain $\mathcal{O}$ ($\mathcal{O} \vDash s$) [78]. For a sentence to be entailed by the domain, it must either express explicit or implicit (i.e., inferred) facts of the domain. From the previous example, we can see that while Axiom 1 is entailed by the domain, the second sentence, although consistent, cannot be entailed from the domain.

5.2.1.2. Incorrect Sentences

The definition of semantic incorrect sentence presented in Section 4.1.2. is still valid for our *online reasoning* approach. A sentence s is semantically incorrect if it is inconsistent with respect to domain ontology $\mathcal{O}$ ($\mathcal{O} \cup s \vDash \bot$) [78].

$$\begin{array}{rl} \text{Ontology} & Producer \sqsubseteq \neg Carnivore \\ & Producer \sqsubseteq \exists produce.Food \\ \\ \text{Axiom 1} & Carnivores \sqsubseteq \exists produce.Food \end{array}$$

In the example, the domain ontology states that *Producers* can create (i.e., produce) *Food* ($Producer \sqsubseteq \neg Carnivore$), and that *Producers* are not *Carnivores* ($Producer \sqsubseteq \exists produce.Food$). Axiom 1 is semantically incorrect because it defines the relationship *Carnivores* produce their *food* ($Carnivores \sqsubseteq \exists produce.Food$), which contradicts the domain ontology.

Because of the new definition of semantic correct sentence presented previously (Section 5.2.1.1.), the definition of semantic incorrect sentence becomes more natural. In general, if a sentence is *not* correct, it is considered to be incorrect. For example, automatic text grading systems based on LSA follow this approach. In LSA-based systems [14], a text is correct if it is very similar to the gold standard (i.e., a perfect text), while an incorrect text has very low similarity. Our definition of semantic incorrect sentence can be restated as the consequence of a false statement, i.e., a sentence s is semantically incorrect if its negation is a consequence of the domain ontology ($\mathcal{O} \models \neg s$).

5.2.1.3. Unknown Sentences

All those sentences that are neither correct or incorrect, shall be consider in this work as *unknown*. Although it might seem that this is the same definition given in Section 4.1.3., it is not based on if the elements mentioned in the sentence are defined in the ontology or not. Because of the new definition of semantic correctness (Section 5.2.1.1.), a sentence is consider as unknown if it is neither true or false with respect to the domain ontology ($\mathcal{O} \not\models s$ and $\mathcal{O} \not\models \neg s$).

Ontology	Producer is not a Carnivore. ($Tree \sqsubseteq Producer$)
	Producers create their own food. ($Producer \sqsubseteq \exists produce.Food$)
Axiom 1	Trees produce their food. ($Tree \sqsubseteq \exists produce.Food$)

In the example, we can see that Axiom is a semantically unknown sentence because it states *Trees* produce their *food* ($Tree \sqsubseteq \exists produce.Food$), and the domain ontology only mentions that *Producers* produce their own *food* ($Producer \sqsubseteq \exists produce.Food$). From the ontology, we cannot determine if the sentence is true or false.

As mentioned in Section 4.1.3., determining if a sentence is semantically unknown in the *precomputed* approach is not practical because its implementation lead to an overlap with determining if a sentence is semantically incorrect. In contrast, under the *online reasoning* approach, identifying a sentence unknown becomes an effect of verifying if a sentence is semantically($\mathcal{O} \vDash s$) or if it is semantically incorrect ($\mathcal{O} \vDash \neg s$).

5.2.2. Ontology Consistency and Completeness

As seen in the preceding section, a sentence type depends on the relationship between a sentence and the domain ontology. In order for the ontology to be able to help us determine the semantic correctness of a sentence, it must meet two requirements: consistency and completeness.

The first requirement, i.e., consistency, is the most important one. In general, a domain ontology is expected to be consistent (i.e., no logical contradictions). A consistent domain ontology provides an unambiguous taxonomic classification of concepts and relationships of a domain. If the ontology is inconsistent, there is a concept or relationship that has two or more irreconcilable interpretations (e.g., disjoint concepts stated in a subclass relationship). An inconsistent ontology is not only problematic in terms of utility. From a more theoretical point of view, an inconsistent ontology is seen as useless since anything can be inferred from a set of contradicting axioms [28].

An inconsistent ontology is not useful in determining text correctness because any sentence would be considered correct. So, consistency of the domain ontology, is fundamental for determining the correctness of a sentence. Although it can be argued that methods, such as those of Huang et al. [33] *inconsistent reasoner* and Liu

et al. [80] *probabilistic logic*, can manage logic contradiction in an ontology, it is not clear the meaning of a sentence that contradicts an inconsistent domain ontology.

On the other hand, the requirement of completeness of the domain ontology has more practical implications. If the domain ontology has only a partial coverage of the domain, it is more likely to leave out of the analysis a set of unknown sentences, which should have been labeled as correct or incorrect. The more complete the domain ontology is, the more accurate is the analysis of the text.

Although it might seem simple to address the issue of completeness of an ontology, e.g., use ontology learning methods [28, 81] or ontology population [44] to make the ontology more complete, change can easily lead to inconsistency (Section 2.1.2.). Ontology completeness might be difficult to address since it can lead to the need of a redesign of the ontology.

5.2.3. Determining Correctness of a Sentence

After a sentence has been transformed from its written form to logic representation that is compatible with the domain ontology, we analyze the sentence semantic correctness.

We start by determining if an extracted statement is correct by entailment (Algorithm 1, line 2). If the extracted statement can be entailed, it is labeled as *correct*. If it cannot be entailed, the statement is added to the ontology to determine its consistency (Algorithm 1, line 3). If the domain ontology becomes inconsistent after an extracted sentence is added to it, then the sentence is *incorrect*. If the extracted statement is not entailed by the domain but consistent with it, the statement is labeled as *unknown* (i.e., incomplete) with respect to the domain

ontology. In this work we have selected as reason HermiT because of its higher efficiency (i.e., hypertableau reasoning algorithm) [26].

```
1  while i ≤ n do
2      if O ⊭ s_i then
3          if O ⊭ ¬s_i then
4              s_i is unknown
5          else
6              s_i is incorrect
7          end
8      end
9      s_i is correct
10 end
```

Algorithm 1: *Online reasoning* approach for semantic error detection in single sentence analysis.

In case of inconsistency (i.e., incorrect sentence), we preferred that the error detection approach could provide an *explanation* of the origin of the inconsistency. For that purpose, we have included into our approach the ontology debugging solution proposed by Horridge et al. [32]. As previously mentioned, Horridge et al. *explanation* approach integrates Reiter's Hitting Set Tree (HST) [38] to identify the minimal inconsistent sub-ontology, i.e., subset of axioms from the ontology that cause the inconsistency. Since the inconsistency is originated by the sentence, the HST-based debugging method can determine which part of the ontology is contradicted by the incorrect sentence (i.e., the explanation).

Horridge et al.'s approach has been incorporated into popular DL reasoners, such as Pellet [24] and HermiT [26].

5.3. Multiple Sentence Analysis

In our previous approaches for semantic error detection, we analyzed individual sentences of the text. Single sentence analysis is based on the notion that a sentence is the smallest linguistic unit from which an IE system can extract information. However, because sentences are usually used to construct paragraphs and documents to express more complex ideas, they are dependent. Although not all sentences of the same document are semantically *connected*, it is very likely that sets of sentences refer to the same concepts and relationships. Let us consider the following example:

$$\begin{array}{ll} \text{Ontology} & Planet \sqsubseteq \neg Dwarf_Planet \\ \\ \text{Axiom 1} & Planet(Pluto) \\ \text{Axiom 2} & Dwarf_Planet(Pluto) \end{array}$$

From the domain ontology, we only know that a *Planet* cannot be *Dwarf_Planet*. If we state that *Pluto* is a *Planet* (Axiom 1), we cannot label it as a semantically correct or incorrect statement. The same occurs with Axiom 2. In other words, if we apply any of our previous approaches to determine the semantic correctness of these two axioms, we would only discover that both axioms are unknown. However, it is clear that, given the domain ontology, these axioms together would make a document semantically incorrect.

In order to identify all possible semantic errors in a text, we need to consider that sentences are not independent of each other, i.e., semantic errors can occur by combining two or more sentences. As in the example, these semantic errors become evident only when analyzing set of sentences as a whole and not as a series of independent sentences.

5.3.1. Analyzing All Sentences Simultaneously

Although it is possible that the multiple sentence semantic error affect all sentences of a text, it is more likely that a set of sentences can be domain-inconsistent. But, because a set of semantically erroneous sentences can be formed with parts of any section of the text (e.g, domain inconsistency between sentences from different paragraphs), determining which set of sentences needs to be analyzed together becomes a difficult issue.

A simple approach would be to analyze all the sentences of a text together. This approach would avoid the complex task of determining which sentences need to be consider as set to be analyzed. It also avoids the problematic of missing a set of semantically incorrect sentences by splitting them into different analysis sets.

However, by considering all sentences at a time, we loose the information that consistency checking can give in the single sentence analysis for the *online reasoning*. Consistency checking can only determine the consistency of the ontology and the set of sentences. In the case of the single sentence analysis, we could determine if a sentence is semantically incorrect. On the other hand, if a set of sentences are inconsistent against the ontology, that means at least one sentence is semantically incorrect. We also cannot differentiate between semantically correct and unknown sentences since both types are consistent with the domain ontology.

It can be argued that we could reduce the error detection problem to ontology debugging (e.g., apply a method such as Horridge et al. [32]). However, if we consider that the number of sentences to be analyzed could be large (analyzing a large document), this approach becomes unpractical. Methods such as Horridge et al. [32], Schlobach and Cornet [34], and Schlobach et al. [35] need to perform multiple consistency checking. We can easily see that this approach becomes unpractical when

considering that consistency checking has an exponential complexity in DL, and the size of the ontology plus the extracted statements is significantly large.

5.3.2. Analyzing an Incremental Set of Sentences

Alternatively, instead of analyzing all sentences at the same time, we can consider a subset of sentences, making the method more practical. However, under a subset approach, it is possible to partition the set of sentences in a way that could eliminate the actual semantic errors.

An option to analyze groups of statements with overlooking error is by incrementally analyzing the set of statements. Iteratively, we add sentences into the ontology, and we perform consistency checking. 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 in which the sentences are being added to the ontology for analysis. For example, we produce the set $S = s_1, ..., s_i, ..., s_j, ..., s_n$ (with i much smaller than j) of extraction from sentences of a text. 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 iterations 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 checking, it can reduce the complexity when trying to find the origin of the inconsistency.

The weakness of this approach is that it can easily degrades into the approach of analyzing all sentences simultaneously. As we iterate, the number of sentences to

analyze will lead us into not determining which sentences are semantically correct, or which sentences from the large set are semantically incorrect.

5.3.3. Reduce Sentence Set

We proposed that the single sentence analysis can provide insight into which sentences need to be considered for multiple sentence analysis, and which sentences do not need to be considered. To identify multiple sentence semantical errors, we need to determine which sentences can provide new information into the analysis.

We propose that the sentences that do not provide new information can be remove from the analysis process without losing content, i.e., the reduction of the set of sentences still leads to a complete analysis. Our reduction is based on *cut elimination* over entailed elements. Cut elimination is the central inference rule in Sequent Calculus.

$$\frac{\Gamma \vdash \Delta, A \qquad A, \Sigma \vdash \Delta}{\Gamma, \Sigma \vdash \Pi, \Delta}. \qquad \text{(Equation 5.1.)}$$

As seen in Equation 5.1., *cut-elimination* mainly express that if we can entail a logical formula A from a set of formulas Γ, we do not need A to entail other elements (e.g., Δ) from Γ since the information of A is already contain in Γ.

Based on *cut-elimination*, we could remove two types of sentences without affecting the completeness of our analysis approach: semantically correct sentences and semantically incorrect sentences. Since semantically correct sentences are consequence of the domain, they do not provide any information that is not already contained in the domain ontology. Similarly, semantically incorrect sentences are *false consequence* of the domain, i.e., inconsistent with the domain ontology.

5.3.3.1. Determining Sentence Types

```
1  U set of unknown sentences while  i ≤ n do
2      if O ⊭ s_i then
3          if O ⊭ ¬s_i then
4              if O ∪ U ∪ s_i ⊨ ⊥ then
5                  U ∪ s_i is incorrect
6              else
7                  s_i is unknown and added to the set of unknown sentences U
8              end
9          else
10             s_i is incorrect
11         end
12     end
13     s_i is correct
14 end
```

Algorithm 2: *Online reasoning* approach for semantic error detection in multiple sentence analysis.

As mentioned, because sentence type can allow us to determine which sentence needs to be consider as part of a set of sentence for analysis, multiple sentence analysis for online reasoning semantic error detection provides a generalized approach of our reasoning-based approach.

The reduction of sentence occurs, as seen in Algorithm 2, by not including semantically correct and semantically incorrect sentences for the following iteration of the process. As it can be seen in line 4 in Algorithm 2, we only evaluate the consistency

between unknown sentences. Since unknown sentences contain information that is not in the ontology, we need to consider them for following iterations.

5.3.3.2. Proof of Completeness of the Analysis

Let us consider the set of extracted relations $S = s_1, ..., s_n$, s_i is an extracted sentence with $i \in [1, n]$, S' is a subset of extracted relations that have already been analyzed ($S' \subsetneq S$), and the domain ontology $\mathcal{O}$.

- s_i: Let us assume that s_i is a correct sentence, i.e., $\mathcal{O} \cup S' \vDash s_i$ is true.

  Then for s_{i+1}: Because $\mathcal{O} \cup S'$ can entail s_i (previous axiom is true), we do not need s_i to determine if s_{i+1} is a logical implication from the domain and the previous sentences. Then through cut elimination, $\mathcal{O} \cup S' \cup s_i \vDash s_{i+1}$ can be reduced to $\mathcal{O} \cup S' \vDash s_{i+1}$.

- s_i: let us assume that s_i is an incorrect sentence, i.e., $\mathcal{O} \cup S' \vDash \neg s_i$ is true.

  Then for s_{i+1}: Similarly to the case of s_i being a correct sentence, we do not need $\neg s_i$ to determine if s_{i+1} is a logical implication from the domain and the previous sentences. Then through cut elimination, $\mathcal{O} \cup S' \cup s_i \vDash s_{i+1}$ can be reduced to $\mathcal{O} \cup S' \vDash s_{i+1}$

- s_i: finally, let us assume that s_i is an unknown sentence.

  $\mathcal{O} \cup S' \vDash s_i$ and $\mathcal{O} \cup S' \vDash \neg s_i$ are false. If $\mathcal{O} \cup S'$ cannot entail s_i (previous axiom is true), then we cannot remove s_i for the analysis of s_{i+1}.

We can see that S' contains all sentence that have been labeled as semantically unknown because if we determine that a sentence is semantically correct (or incorrect), we do not need to consider it for the following analysis.

5.4. Evaluation

We have evaluated both the single sentence analysis and the multiple sentence analysis of our proposed *online reasoning* approach for semantic error detection. Following sections provide details of datasets, ontologies, and comparison methods used for each type of analysis.

5.4.1. Evaluating Single Sentence Analysis

We have evaluated our online reasoning approach for single sentence analysis against the Ecosystems dataset presented in Section 4.3.1. Following sections provide details regarding the data itself, the ontology constructed, and the metrics used for evaluation.

5.4.1.1. Overview of the Dataset and Ontology

As mentioned in Section 4.3.1., the Ecosystem dataset is a subset of 18 summaries from the study by Sohlberg et al. [82] regarding electronic strategies (eStrategies) for reading comprehension. The summaries of the Ecosystem dataset are oral summaries manually transcribed that range from a pair of sentences to 60 sentences. The summaries are based on a single, 3 pages in length, article which provides a introduction to the topic of Ecosystems. We have preprocessed to resolve anaphoras (e.g., pronouns) and on correcting misspellings.

On the other hand, the ontology used for this evaluation is based on the same article used by the students for summarization. The construction of the ontology is constrained to explicit facts from the domain knowledge defined by the article, and does not include facts from the entire domain of Ecosystems.

Element type	Original	Extended
Concepts	45	55
Relationships	28	30
Axioms	7	224

TABLE 5.1. Comparison between statistical information about the original ontology presented in Section 4.3.1. to evaluate the *precomputed* approach and the extended ontology used to evaluate our proposed *online reasoning* approach.

Although the ontology used in our present approach is similar to the one used in Section 4.3.1., there is a significant difference in the number of axioms of each ontology (Table 5.1). In order to determine incorrectness based on logic contradiction, the ontology for the present evaluation incorporates a large set of constraints, such as disjointness between classes, and strictly defines domain and range for each property.

5.4.1.2. Evaluation Metrics and Comparison Methods

To obtain a better understanding of how well our *online reasoning* method performs, we are comparing the performance of our method against two comparison methods. Just as in Chapters III and IV, we will use precision, recall, and F1 as metrics since our goal is to evaluate how complete is the analysis of these methods.

The first method is our previous *precomputed* approach defined in Chapter IV, which is, to the best of our knowledge, the only ontology-based semantic error detection method. As previously mentioned (Chapter IV), our previous *precomputed* approach defines domain inconsistent axioms by violating ontological constraints. These domain-inconsistent axioms are encoded into extraction patterns that can detect semantically incorrect sentences before the extraction process begins (i.e., precomputed approach).

For comparison, we have used the same set of rules manually defined before. We created the extraction rules by using the domain ontology and considering the content

documents. Because it is possible to generate a large set of inconsistent axioms from the domain ontology, we use the content of the four documents to limit the number of extraction rules that need to be generated. This led to 31 extraction rules to identify correct sentences, 16 extraction rules to identify incorrect sentences, and five extraction rules to identify incomplete sentences.

The second comparison method is a variation to our *online reasoning* approach that replace the IE process with *manual extraction*. This variation can provide us with insight of how the mapping and reasoning steps perform when analyzing correctness. Because currently available IE implementations are not 100% accurate, the overall performance of error detection might be affected by the IE process. The use of *manual extractions* can lead to an overall performance depending on directly the performance of the mapping and reasoning steps of our approach. We have constructed a data set formed by binary relationships manually extracted from the 18 summaries. These manually extracted relationships are then analyzed by our approach to determine their correctness.

For the mapping step, we use the same dictionaries for both proposed approach (i.e., automatic extraction) and the *manual extraction* method. The dictionaries were constructed by observing extracted relationships from 40 sentences taken from four of the 18 summaries.

5.4.1.3. Results

From Table 5.2, we can say that in the case of *online reasoning* approach, it is possible to determine with high precision the semantic correctness of a sentence with respect to the domain by logic reasoning. However, there is a significant amount of sentences that, although contained in the domain, are considered to be unrelated to

Sentence	Automatic Extraction	Manual Extraction	Precomputed approach
Correct	100%	100%	91.9%
Incorrect	100%	100%	97.4%
Unknown	89.5%	74.71%	-

Sentence	Automatic Extraction	Manual Extraction	Precomputed approach
Correct	40.9%	80.23%	83.3%
Incorrect	41.3%	88.63%	88.6%
Unknown	100%	100%	-

Sentence	Automatic Extraction	Manual Extraction	Precomputed approach
Correct	58.1%	89.0%	87.4%
Incorrect	58.4%	93.97%	92.8%
Unknown	94.4%	74.71%	-

TABLE 5.2. Precision (top), recall (center), and F1 measure (bottom) for the proposed method (automatic and manually extraction) and for the *precomputed* approach [1].

the domain. There are a significant amount of cases where the IE process extracted phrases as entities. Although this is not strictly incorrect, most of these phrases represented something more than only a domain concept. This leads to a lower recall. On the other hand, although not all semantically correct and incorrect sentences were captured, the sentences that were labeled as correct are all semantically correct sentences. The same goes with the semantically incorrect sentences.

The perfect precision (i.e., 100%) obtained by both *online reasoning* and the *manual extraction* approaches in the case of semantically correct and incorrect sentences might seem unrealistic. However, it is the natural outcome given the underlying method used in the process (i.e., reasoning). If one sentence was labeled as correct when it was actually incorrect, it would mean that reasoning process used to determine the label the sentence is not accurate. However, as previously mentioned,

we are using a DL reasoner (i.e., HermiT) which is sound and complete. So, once the semantic elements of a sentence are mapped to the ontology, the reasoner can accurately determine if it contradicts the domain ontology or not.

In the case of *manually extracted* relations, we can observe an increment in the recall with respect to the *online reasoning* approach, with the same level of precision. This result indicates that the quality of the extraction process has a significant effect in the detection of correctness, it is not the only factor affecting the recall of correct and incorrect sentences. In the case of *manual extractions*, the error in determining the correctness of a sentence can be explained by the mapping between extractions and ontology. The correct (and incorrect) sentences that were labeled as incomplete are cases where the mapping procedure failed to connect extraction entities with ontological concepts.

When compared with our previous approach, precomputed error detection, both our proposed automatic extraction and manual extraction methods are more accurate when identifying incorrect sentences. On the other hand, because our previous approach seeks specific pre-defined patterns in the text, it has a higher recall. However, the *precomputed error* has higher deployment conditions (i.e., overhead) since the extraction rules need to be created by domain and ontology experts.

5.4.2. Evaluating Multiple Sentence Analysis

We have also evaluated our *online reasoning* approach for multiple sentence analysis. However, because multiple sentence analysis is a new approach to semantic error detection, rather than evaluating the method, we provide some observations from the execution of this new approach over two synthetic datasets.

5.4.2.1. Synthetic Datasets

Currently there is not datasets for semantic errors on multiple sentences. For this evaluation, we have generated two synthetic dataset that contains multiple sentence semantical errors.

Ecosystem Dataset. We have used this dataset for the evaluation of both *precomputed* semantic error detection (Section 4.3.1.) and the *online reasoning* semantic error detection for single sentence analysis (Section 5.4.). It consists of 18 oral student summaries that have been manually transcribed. The length of the summaries can vary significantly, from 2 to 60 sentence.

For multiple sentence, we have used the same ontology defined in Section 5.4.1.1. It is based on the introduction article read by the students participating in the study. The ontology has explicitly defined all logical constraints that are usually left undefined, such as disjointness between sibling concepts, and defined domain and range for properties.

We have introduce into the summaries 20 sentences that, by them selfs, are semantically unknown. However, when these sentences are analyzed in a set, they are semantically incorrect. We have randomly added these sentence into 10 of the 18 summaries.

Wikipedia's County Dataset. It consists of 570 articles from Wikipedia regarding counties of the United States. These articles vary significantly in length, with some articles containing less than 10 sentence, while others containing more than 60.

Element type	Number of element
Concepts	15
Relationships	6
Axioms	40

TABLE 5.3. Statistical information about the ontology.

We have designed an ontology following patterns described in previous sections (e.g., Section 3.3.2.2.). Because the counties' articles had a very limited number of share topics (e.g., origin of the name of the county), the ontology is small in comparison to other ontologies used for evaluation of semantic error (Figure 5.2). However, it still has a large number of constraints (Table 5.3).

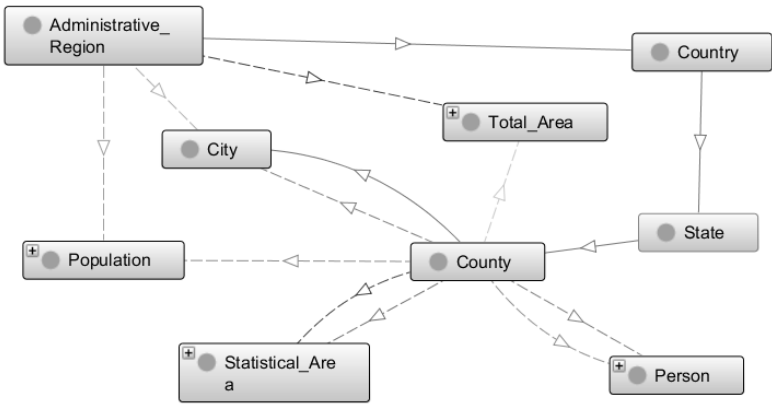

FIGURE 5.2. Graphical representation of a section of the County ontology

The synthetic error introduce into the article is based on characteristics of the dataset. There 41 cases where two or more counties, from different state, sharing the same name. The semantic error is introduce by adding sentences from one county to another county that has the same name. Because of constraints such as a county can have on seat and it can belong to one state, the inclusion of a sentence indicating another seat (or state) than the one in the article creates domain-inconsistency across

multiple sentences. It is very likely that, at some point, these types of semantic error might have occurred before the content was verified by Wikipedia editors.

5.4.2.2. Results

As mentioned, because semantic error detection over multiple sentence is a new approach, there are no comparison methods. However, we can still get some insight from the performance of the method.

In the case of the Ecosystem dataset, the results are mostly a reflection the performance of the single sentence analysis. If the sentence was extracted and mapped correctly to the ontology, the multiple sentence analysis method would accurately identify the semantically incorrect sentences (90%). When the transformation from text to logic clause fails, the sentences are labeled as unknown.

One of the mapping issues occurred because of a negation in the sentence. Although information extraction system can handle negation in most cases, it is not clear to which element in the ontology it should map. Because most DL languages cannot handle complex negation of concepts, we have negation mostly used in ontologies to define disjointness between concepts. Let us consider the concept $Carnivore$ from the Ecosystem ontology, which is disjoint with a set of concepts. It is unclear if the statement $no_Carnivore$ refers to all of the concepts that are disjoint to $Carnivore$, or it refers to a specific concept like $Herbivore$.

CHAPTER VI

CONCLUSION

As automatic processing of written natural language progresses, while processes, such as IE, moves to sources where the quality of text content cannot be guaranteed, it seems reasonable to identify mechanisms that can help to coup with this lack of quality. In this dissertation, we have explored how to overcome these difficulties in IE by combining mechanisms of different nature. We have focused on two orthogonal issues that affect IE: accuracy of extraction and semantic correctness of extraction. The present dissertation, which consists of three parts, presents three different approaches to improve accuracy and tackle semantic correctness.

In the first part of the dissertation, we proposed a hybrid implementation approach for OBIE, which leads to a more accurate extraction process. It considers the use of combined information extractors with different implementations. By using both implementations (extraction rules and machine learning-based extractors), it is possible to obtain higher accuracy in the extraction process. We offer a selection strategy and an integration strategy to combine information extractors with different implementations. The selection strategy determines the most accurate set of information extractors by determining which implementation commits fewer extraction errors. The integration strategy uses the ensemble method of stacking to combine the outputs of both implementations. Stacking trains a classifier from the outputs of the underlying methods (i.e. information extractors) to produce a more accurate extraction. The evaluation of our proposed approach shows a clear improvement in accuracy, providing an overall balance between precision and recall of the extracted information.

In the second part of the dissertation, we proposed a semantic error detection method based on traditional Ontology-based Information Extraction, where semantic errors are *precomputed*. Because an ontology only represents domain facts, this approach requires a mechanism to create (or generate) axioms that are incorrect with respect to the domain (i.e., domain-inconsistent axioms). These domain-inconsistent axioms are encoded into information extractors that are applied to the text. The information extractors were implemented as pattern-based rules, and as machine learning based extractors in order to determine the most suitable method for identifying incorrectness. Our approach to semantic error detection shows that it is possible to integrate this new functionality without affecting traditional extraction (i.e., semantically correct information). We can also see that it is possible to obtain accurate extractions in spite of the inherent complexity of identifying semantic error.

In the third and final part of my dissertation, we proposed a semantic error detection method based on *reasoning*. Under this approach, the text sentence needs to be transformed from written natural language into a logic like representation, such as IE extracted tuples. With the text in a logic form plus the domain ontology, we apply ontology debugging methods, through reasoning, to determine the type of sentence and the origin of the error. In contrast to the precomputed semantic error, where the origin of the incorrectness is known because of the generation mechanism, this reason based approach requires an explicit methods to determine the origin (i.e., explanation) of the semantic error. We extended this reasoning-based method to analyze a text as a whole and not as a set of independent sentences. This extension has led to a generalized approach to error detection, which will allow analysis of both single and multiple sentences. The evaluation of our proposed reasoning-based approach showed that, although dependent on the quality of the extraction by the underlying IE system,

such method can produce an accurate and very complete extraction, identifying single and multiple sentence semantic errors.

6.1. Future Work

There are some aspect of the previous work that we believe can be extended into the following work:

1. **Hybrid Implementation** From our work in hybrid implementation, there are a few pending goals that we would like to analyze in more details.

   (a) *Alternative combination strategies.* We would like to see if there are alternative strategies that would allow a more accurate combination of information extractors, such as the *constraint coupling* approach by Carlson et al. [44] (logic constraints to improve accuracy) or the *multiple OBIE* approach by Wimalasuriya and Dou [12].

   For example, we want to see whether combining information extractors of the same concept but different functionality can lead to a more accurate extraction. A simple approach is to use an information extractor with one functionality as a preprocessor for the other functionality. Preliminary work shows that is possible to reach improvements under this *functional preprocessing* approach around 10%.

   (b) *Alternative implementation approaches.* In our proposed hybrid implementation, rule-based and machine learning-based information extractors are combined to improve accuracy of the extraction. We would like to see if other methods, such graph model-based IE [58, 83–85] or more sophisticated rule-based extractors (based on JAPE [52] or AQL [51]), can

be combined into our hybrid approach to improve the extraction accuracy even further.

2. **Hybrid Semantic Error Detection** From our work in semantic error detection in text, we have identified three goals that require improvement.

   (a) *Specialized IE strategy for semantic error detection.* Our online reasoning approach, discussed in Chapter V, uses an unsupervised extraction strategy to produce the most complete set of extractions (of relationships) as possible. However, because unsupervised IE focus in the extraction of entities rather than concepts (e.g., non-verb mediated relationship between concepts), it can lead to unrecognizable extractions that might not be possible to map to the ontology. We believe that this situation could be solved by domain-aware methods such as current approaches to semi-supervised IE [58, 83–85], or Named Entity Linking (domain-based approach to Named Entity Recognition) [84, 86].

   (b) *Improve mapping between extraction and ontology.* The mapping method offered in this work is a simple and direct approach to the problem. However, we need better mechanisms to define mappings between the vocabulary of the text and the vocabulary of the ontology, specially when consider larger document sets. We believe that this aspect of our method can be automated by the inclusion of linguistic tools such *WordNet* [66], or logic consistency [87].

   (c) *Explanation method for semantic error detection.* Although current ontology debugging methods can provide tentative solutions to this problem, they have both different focus and different parameters to find the

origin of inconsistency. As mentioned in Section 2.1.3., because in ontology debugging the origin of the inconsistency is not known, a search mechanism must be defined as part of the debugging process, which might no always work. In the case semantic error detection the origin of the inconsistency is the ontological axioms that are affected the analyzed text. We believe that use of a selection function, such as the on used by Schlobach et al. [35], would lead to a more reliable and efficient method for inconsistency explanation.

APPENDIX A

SEQUENT CALCULUS

Our proposed reduction is based on applying *sequent calculus* inference rules to the analyzed sentence. *Sequent calculus* (Gentzen1934) is a logical argumentation style that applies derivation rules to a sequence of sequents (i.e., logical expression). The idea is that we apply inference rules to $A_1, ..., A_n \vdash B_1, ..., B_m$, deriving a set of C_l. For each C_l, we want to obtain

$$\frac{}{C_l \vdash C_l}(I).$$

Inference rules in sequent calculus are group by the side of $\vdash$ they affect, and if they apply to operators (logic rules) or to formulas (structural rules). The central rules of sequent calculus is *cut-elimination*:

$$\frac{\Gamma \vdash \Delta, A \qquad A, \Sigma \vdash \Delta}{\Gamma, \Sigma \vdash \Pi, \Delta}.$$

The following are a subset of the inference rules:

$$\frac{\Gamma, A \vdash \Delta}{\Gamma, A \wedge B \vdash \Delta}(\wedge L_1) \qquad \frac{\Gamma, B \vdash \Delta}{\Gamma, A \wedge B \vdash \Delta}(\wedge L_2)$$

$$\frac{\Gamma \vdash A, \Delta}{\Gamma, \neg A \vdash \Delta}(\neg L) \qquad \frac{\Gamma, A \vdash \Delta}{\Gamma \vdash \neg A, \Delta}(\neg R)$$

$$\frac{\Gamma \vdash \Delta, A \quad \Sigma, B \vdash \Pi}{\Gamma, \Sigma, A \to B \vdash \Delta, \Pi}(\to L) \qquad \frac{\Gamma, A \vdash B, \Delta}{\Gamma \vdash A \to B, \Delta}(\to R\ )$$

$$\frac{\Gamma_1, A, B, \Gamma_2 \vdash \Delta}{\Gamma_1, B, A, \Gamma_2 \vdash \Delta}(PL) \qquad \frac{\Gamma \vdash \Delta_1, A, B, \Delta_2}{\Gamma \vdash \Delta_1, B, A, \Delta_2}(PR)$$

In the previous inference rules, A and B are first-order predicate logic formulas, Γ, Δ, Σ and Π are sets of formulas (that can be empty).

APPENDIX B

INCONSISTENCY AS COMPLEMENT ENTAILMENT

We can restate a incorrect sentence as its complement being entailed by the domain as $\mathcal{O} \vDash \neg\omega_i$. In order to use this redefinition, we need to demonstrate its equivalence with the original definition of error. In other words, we need to prove that $\mathcal{O} \wedge \omega \vDash \bot \longrightarrow \mathcal{O} \vDash \neg\omega$. This equivalence can be proved through sequent calculus (Appendix A). First, we will use the relation between absurdity and negation typically used in sequent calculus $\neg A \longleftrightarrow A \rightarrow \bot$. This transform the original expression into:

$$\mathcal{O} \wedge \omega \vDash \bot \longrightarrow \mathcal{O} \vDash \neg\omega$$

Second, we will consider for simplicity that $\mathcal{O}$ is a set of one element (e.g., $\mathcal{O} = \{\neg\omega\}$), or as the element itself. It can be easily seen that the following proof can be extended $\mathcal{O}$ with multiple concepts and properties.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\omega \vdash \omega}(I)}{\vdash \neg\omega, \omega}(\neg R) \quad \dfrac{}{\mathcal{O} \vdash \mathcal{O}}(I)}{\vdash \omega, \neg\omega}(PR)}{\mathcal{O} \vdash \mathcal{O} \wedge \omega, \neg\omega}(\wedge R)}{\mathcal{O}, \neg(\mathcal{O} \wedge \omega) \vdash \neg\omega}(\neg L)}{\neg(\mathcal{O} \wedge \omega), \mathcal{O} \vdash \neg\omega}(PL)}{\neg(\mathcal{O} \wedge \omega) \vdash \mathcal{O} \rightarrow \neg\omega}(\rightarrow R)}{\vdash \neg(\mathcal{O} \wedge \omega) \rightarrow (\mathcal{O} \rightarrow \neg\omega)}(\rightarrow R)$$

REFERENCES CITED

[1] Fernando Gutierrez, Dejing Dou, Steven Fickas, and Gina 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 '12, pages 1722–1726, 2012.

[2] Daniel Jurafsky and James H. Martin. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics and Speech Recognition*. Prentice Hall, second edition, February 2008. ISBN 013122798X. URL `http://www.worldcat.org/isbn/013122798X`.

[3] Tom Mitchell, Terry Russell, Peter Broomhead, and Nicola Aldridge. Towards robust computerised marking of free-text responses. In *Proceedings of the 6th CAA Conference*. Loughborough University, 2002. URL `http://hdl.handle.net/2134/1884`.

[4] 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.

[5] Kamel Nebhi. Ontology-based information extraction for french newspaper articles. In Birte Glimm and Antonio Krger, editors, *KI 2012: Advances in Artificial Intelligence*, volume 7526 of *Lecture Notes in Computer Science*, pages 237–240. Springer Berlin Heidelberg, 2012. ISBN 978-3-642-33346-0.

[6] Kamel Nebhi. Named entity disambiguation using freebase and syntactic parsing. In *Workshop on Linked Data for Information Extraction (LD4IE)*, 2013.

[7] Peter Geibel, Martin Trautwein, Hebun Erdur, Lothar Zimmermann, Kati Jegzentis, Michaela Bengner, ChristianHans Nolte, and Thomas Tolxdorff. Ontology-based information extraction: Identifying eligible patients for clinical trials in neurology. *Journal on Data Semantics*, 4(2):133–147, 2015. ISSN 1861-2032.

[8] Thomas R. Gruber. A translation approach to portable ontology specifications. *Knowl. Acquis.*, 5(2):199–220, June 1993. ISSN 1042-8143. doi: 10.1006/knac.1993.1008. URL `http://dx.doi.org/10.1006/knac.1993.1008`.

[9] Daya C. Wimalasuriya and Dejing Dou. Ontology-based information extraction: An introduction and a survey of current approaches. *Journal of Information Science*, 36:306–323, June 2010. ISSN 0165-5515. doi: http://dx.doi.org/10.1177/0165551509360123. URL `http://dx.doi.org/10.1177/0165551509360123`.

[10] Padmini Srinivasan and Xin Ying Qiu. Go for gene documents. *BMC Bioinformatics*, 2007.

[11] Fernando Gutierrez, Daya C. Wimalasuriya, and Dejing 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)*, volume 7046 of *Lecture Notes in Computer Science*, pages 31–32, 2011.

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

[13] Daya C. Wimalasuriya and Dejing 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.

[14] Thomas K. Landauer, Darrell Laham, and Peter W. Foltz. Learning human-like knowledge by singular value decomposition: a progress report. In *Proceedings of the 1997 Conference on Advances in Neural Information Processing Systems 10*, NIPS '97, pages 45–51, 1998. ISBN 0-262-10076-2. URL `http://dl.acm.org/citation.cfm?id=302528.302546`.

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

[16] Alan Ritter, Doug Downey, Stephen Soderland, and Oren 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)*, pages 11–20, 2008. URL `http://dl.acm.org/citation.cfm?id=1613715.1613718`.

[17] Michele Banko, Michael J. Cafarella, Stephen Soderland, Matt Broadhead, and Oren Etzioni. Open information extraction from the web. In *20th International Joint Conference on Artificial intelligence (IJCAI)*, IJCAI'07, pages 2670–2676, San Francisco, CA, USA, 2007. Morgan Kaufmann Publishers Inc. URL `http://dl.acm.org/citation.cfm?id=1625275.1625705`.

[18] Michele Banko and Oren Etzioni. The tradeoffs between open and traditional relation extraction. In *Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (ACL-HLT)*, pages 28–36, 2008. URL `http://www.aclweb.org/anthology/P/P08/P08-1004`.

[19] Marie-Catherine de Marneffe, Anna N. Rafferty, and Christopher D. Manning. Finding contradictions in text. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1039–1047, 2008. ISBN 978-1-932432-04-6.

[20] Giorgos Flouris, Dimitris Manakanatas, Haridimos Kondylakis, Dimitris Plexousakis, and Grigoris Antoniou. Ontology change: Classification and survey. *The Knowledge Engineering Review*, 23(02):117–152, 2008. doi: 10.1017/S0269888908001367. URL `http://dx.doi.org/10.1017/S0269888908001367`.

[21] Sean Bechhofer, Frank van Harmelen, Jim Hendler, Ian Horrocks, Deborah L. McGuinness ant Peter F. Patel-Schneider, and Lynn Andrea Stein. OWL Web Ontology Language. `http://www.w3.org/TR/owl-ref/`.

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

[23] Volker Haarslev and Ralf Müller. Racer system description. In Rajeev Goré, Alexander Leitsch, and Tobias Nipkow, editors, *Automated Reasoning*, volume 2083 of *Lecture Notes in Computer Science*, pages 701–705. Springer Berlin / Heidelberg, 2001. URL `http://dx.doi.org/10.1007/3-540-45744-5`. 10.1007/3-540-45744-5_59.

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

[25] Dmitry Tsarkov and Ian Horrocks. Fact++ description logic reasoner: System description. In *Proceedings of the Third International Joint Conference on Automated Reasoning (IJCAR)*, pages 292–297, 2006. ISBN 3-540-37187-7, 978-3-540-37187-8. doi: 10.1007/11814771. URL `http://dx.doi.org/10.1007/11814771`.

[26] Boris Motik, Rob Shearer, and Ian Horrocks. Hypertableau Reasoning for Description Logics. *Journal of Artificial Intelligence Research*, 36:165–228, 2009.

[27] Ian Horrocks and Peter Patel-Schneider. Reducing owl entailment to description logic satisfiability. In *Web Semantics: Science, Services and Agents on the World Wide Web*, pages 345–357. 2004. ISBN 978-3-540-20362-9. URL http://dx.doi.org/10.1007/978-3-540-39718-2.

[28] Peter Haase and Johanna Völker. Ontology learning and reasoning – dealing with uncertainty and inconsistency. In Paulo Cesar Costa, Claudia D'Amato, Nicola Fanizzi, Kathryn B. Laskey, Kenneth J. Laskey, Thomas Lukasiewicz, Matthias Nickles, and Michael Pool, editors, *Uncertainty Reasoning for the Semantic Web I*, pages 366–384. 2008. ISBN 978-3-540-89764-4. doi: http://dx.doi.org/10.1007/978-3-540-89765-1. URL http://dx.doi.org/10.1007/978-3-540-89765-1.

[29] Giorgos Flouris, Zhisheng Huang, Jeff Z. Pan, Dimitris Plexousakis, and Holger Wache. Inconsistencies, negations and changes in ontologies. In *Proceeding of the 21st National Conference on Artificial Intelligence*, page 6. AAAI, 2006.

[30] Xi Deng, Volker Haarslev, and Nematollaah 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. ISBN 978-3-540-72666-1. doi: http://dx.doi.org/10.1007/978-3-540-72667-8. URL http://dx.doi.org/10.1007/978-3-540-72667-8.

[31] Peter Haase and Ljiljana Stojanovic. Consistent evolution of owl ontologies. In *Proceedings of the 2nd European Semantic Web Conference (ESWC)*, pages 182–197, 2005.

[32] Matthew Horridge, Bijan Parsia, and Ulrike Sattler. Explaining inconsistencies in owl ontologies. In *Scalable Uncertainty Management*, pages 124–137, 2009.

[33] Zhisheng Huang, Frank van Harmelen, and Annette ten Teije. Reasoning with inconsistent ontologies. In *Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence*, IJCAI '05, pages 454–459, 2005.

[34] Stefan Schlobach and Ronald Cornet. Non-standard reasoning services for the debugging of description logic terminologies. In *Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence (IJCAI)*, pages 355–362, 2003.

[35] Stefan Schlobach, Zhisheng Huang, Ronald Cornet, and Frank van Harmelen. Debugging incoherent terminologies. *Journal of Automated Reasoning*, 39(3): 317–349, 2007. URL http://dx.doi.org/10.1007/s10817-007-9076-z.

[36] Bijan Parsia, Evren Sirin, and Aditya 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. ISBN 1-59593-046-9. doi: 10.1145/1060745.1060837. URL http://doi.acm.org/10.1145/1060745.1060837.

[37] Aditya Kalyanpur, Bijan Parsia, Matthew Horridge, and Evren 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. ISBN 3-540-76297-3, 978-3-540-76297-3. URL http://dl.acm.org/citation.cfm?id=1785162.1785183.

[38] Raymond Reiter. A theory of diagnosis from first principles. *Artificial Intelligence*, 32(1):57 – 95, 1987. ISSN 0004-3702. doi: http://dx.doi.org/10.1016/0004-3702(87)90062-2. URL http://www.sciencedirect.com/science/article/pii/0004370287900622.

[39] Eugene Agichtein and Luis 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. doi: 10.1145/336597.336644. URL http://dx.doi.org/10.1145/336597.336644.

[40] Mike Mintz, Steven Bills, Rion Snow, and Dan 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*, ACL '09, pages 1003–1011, 2009. ISBN 978-1-932432-46-6.

[41] Fei Wu and Daniel S. Weld. Automatically refining the wikipedia infobox ontology. In *Proceedings of the 17th International Conference on World Wide Web (WWW)*, pages 635–644, 2008. ISBN 978-1-60558-085-2. doi: 10.1145/1367497.1367583. URL http://doi.acm.org/10.1145/1367497.1367583.

[42] Burr 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. URL http://dl.acm.org/citation.cfm?id=1567594.1567618.

[43] Alexander Maedche and Steffen Staab. Ontology learning for the semantic web. *IEEE Intelligent Systems*, 16(2):72–79, March 2001. ISSN 1541-1672. doi: 10.1109/5254.920602. URL http://dx.doi.org/10.1109/5254.920602.

[44] Andrew Carlson, Justin Betteridge, Estevam R. Hruschka, and Tom 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. ISBN 978-1-932432-38-1. URL `http://rtw.ml.cmu.edu/papers/cbl-sslnlp09.pdf`.

[45] Eyal Oren, Knud M?ller, Simon Scerri, Siegfried Handschuh, and Michael Sintek. What are semantic annotations? Technical report, DERI Galway, 2006. URL `http://www.siegfried-handschuh.net/pub/2006/whatissemannot2006.pdf`.

[46] Anthony Fader, Stephen Soderland, and Oren Etzioni. Identifying relations for open information extraction. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1535–1545, 2011. ISBN 978-1-937284-11-4. URL `http://dl.acm.org/citation.cfm?id=2145432.2145596`.

[47] Mausam, Michael Schmitz, Stephen Soderland, Robert Bart, and Oren Etzioni. Open language learning for information extraction. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, EMNLP-CoNLL'12, pages 523–534, 2012. ISBN 978-1-937284-43-5.

[48] Fei Wu and Daniel S. Weld. Open information extraction using wikipedia. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 118–127, 2010. URL `http://dl.acm.org/citation.cfm?id=1858681.1858694`.

[49] Daniel S. Weld, Raphael Hoffmann, and Fei Wu. Using wikipedia to bootstrap open information extraction. *SIGMOD Rec.*, 37(4):62–68, March 2009. ISSN 0163-5808. doi: 10.1145/1519103.1519113. URL `http://doi.acm.org/10.1145/1519103.1519113`.

[50] 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. URL `http://journals.plos.org/plosbiology/article?id=10.1371/journal.pbio.0020309`.

[51] Rajasekar Krishnamurthy, Yunyao Li, Sriram Raghavan, Frederick Reiss, Shivakumar Vaithyanathan, and Huaiyu Zhu. Systemt: a system for declarative information extraction. *SIGMOD Record*, 37(4):7–13, 2008.

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

[53] Fernando Gutierrez, Dejing Dou, Steven Fickas, Adam Martini, and Hui 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 359–364, 2013.

[54] Fei Wu, Raphael Hoffmann, and Daniel 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)*, pages 731–739, 2008. ISBN 978-1-60558-193-4. doi: 10.1145/1401890.1401978. URL `http://doi.acm.org/10.1145/1401890.1401978`.

[55] Fei Wu and Daniel S. Weld. Autonomously semantifying wikipedia. In *Proceedings of the 16th ACM Conference on Information and Knowledge Management (CIKM)*, pages 41–50, 2007. ISBN 978-1-59593-803-9. doi: 10.1145/1321440.1321449. URL `http://doi.acm.org/10.1145/1321440.1321449`.

[56] Rion Snow, Daniel Jurafsky, and Andrew Y. Ng. Learning syntactic patterns for automatic hypernym discovery. In *Advances in Neural Information Processing Systems (NIPS 2004)*, November 2004. URL `http://ilpubs.stanford.edu:8090/665/`. This is a draft version from the NIPS preproceedings; the final version will be published by April 2005.

[57] Andre Blessing and Hinrich Schütze. Crosslingual distant supervision for extracting relations of different complexity. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management (CIKM)*, pages 1123–1132, 2012. ISBN 978-1-4503-1156-4. doi: 10.1145/2396761.2398411. URL `http://doi.acm.org/10.1145/2396761.2398411`.

[58] Sebastian Riedel, Limin Yao, and Andrew 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)*, pages 148–163, 2010. ISBN 3-642-15938-9, 978-3-642-15938-1. URL `http://dl.acm.org/citation.cfm?id=1889788.1889799`.

[59] Marti A. Hearst. Automatic Acquisition of Hyponyms from Large Text Corpora. In *Proceedings 14th Conference on Computational Linguistics (COLING)*, pages 539–545, 1992.

[60] Oren Etzioni, Michael Cafarella, Doug Downey, Stanley Kok, Ana-Maria Popescu, Tal Shaked, Stephen Soderland, Daniel S. Weld, and Alexander Yates. Web-scale information extraction in knowitall: (preliminary results). In *13th International Conference on World Wide Web (WWW)*, pages 100–110, 2004. ISBN 1-58113-844-X. doi: 10.1145/988672.988687. URL http://doi.acm.org/10.1145/988672.988687.

[61] Oren Etzioni, Michael Cafarella, Doug Downey, Ana-Maria Popescu, Tal Shaked, Stephen Soderland, Daniel S. Weld, and Alexander Yates. Unsupervised named-entity extraction from the web: an experimental study. *Artif. Intell.*, 165(1):91–134, June 2005. ISSN 0004-3702. doi: 10.1016/j.artint.2005.03.001. URL http://dx.doi.org/10.1016/j.artint.2005.03.001.

[62] Doug Downey, Oren Etzioni, and Stephen Soderland. A probabilistic model of redundancy in information extraction. In *19th International Joint Conference on Artificial intelligence (IJCAI)*, IJCAI'05, pages 1034–1041, San Francisco, CA, USA, 2005. Morgan Kaufmann Publishers Inc. URL http://dl.acm.org/citation.cfm?id=1642293.1642459.

[63] Fernando Gutierrez, Dejing Dou, Steven Fickas, Daya Wimalasuriya, and Hui Zong. A hybrid ontology-based information extraction system. *Journal of Information Science*, 2015. doi: 10.1177/0165551515610989. Accepted.

[64] TG Dietterich. Ensemble methods in machine learning. In *International workshop on multiple classifier systems*, pages 1–15, 2000.

[65] J. Sill, G. Takacs, L. Mackey, and D. Lin. Feature-weighted linear stacking. *ArXiv e-prints*, November 2009.

[66] George A. Miller. Wordnet: A lexical database for english. *Communications of the ACM*, 38:39–41, 1995.

[67] Andrew McCallum, Ronald Rosenfeld, Tom M. Mitchell, and Andrew Y. Ng. Improving text classification by shrinkage in a hierarchy of classes. In *Proceedings of the Fifteenth International Conference on Machine Learning*, ICML '98, pages 359–367, San Francisco, CA, USA, 1998. Morgan Kaufmann Publishers Inc. ISBN 1-55860-556-8. URL http://dl.acm.org/citation.cfm?id=645527.657461.

[68] John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, pages 282–289, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc. ISBN 1-55860-778-1. URL http://dl.acm.org/citation.cfm?id=645530.655813.

[69] A Yakushiji, Y Miyao, T Ohta, Y Tateisi, and J Tsujii. Automatic construction of predicate-argument structure patterns for biomedical information extraction. In *ACL conference on empirical methods in natural language processing, Association for Computational Linguistics*, pages 284–292, 2006.

[70] Diana Maynard, Wim Peters, and Yaoyong Li. Metrics for evaluation of ontology-based information extraction. In *WWW 2006 Workshop on "Evaluation of Ontologies for the Web" (EON)*, Edinburgh, Scotland, UK, May 2006.

[71] Philipp Cimiano, Günter Ladwig, and Steffen Staab. Gimme the context: Context-driven automatic semantic annotation with c-pankow. In Tatsuya Hagino Allan Ellis, editor, *Proceedings of the 14th World Wide Web Conference*, pages 332 – 341, Chiba, Japan, May 2005. ACM Press.

[72] Hai Wang, Matthew Horridge, Alan Rector, Nick Drummond, and Julian 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] Oren Etzioni, Michele Banko, Stephen Soderland, and Daniel S. Weld. Open information extraction from the web. *Commun. ACM*, 51(12):68–74, December 2008. ISSN 0001-0782. doi: 10.1145/1409360.1409378. URL `http://doi.acm.org/10.1145/1409360.1409378`.

[74] McKay M. Sohlberg, Laurie Ehlhardt, Stephen Fickas, and Alistair Sutcliffe. A pilot study exploring electronic mail in users with acquired cognitive-linguistic impairments. *Brain Injury*, 17(7):609–629., 2003.

[75] Iraide Zipitria, Jon A. Elorriaga, Ana Arruarte Lasa, and Arantza Daz de Ilarraza Snchez. From human to automatic summary evaluation. In *Intelligent Tutoring Systems 2004*, pages 432–442, 2004.

[76] Peter W. Foltz, Darrell Laham, and Thomas K. Landauer. Automated essay scoring: Applications to educational technology. In Betty Collis and Ron Oliver, editors, *Proceedings of World Conference on Educational Multimedia, Hypermedia and Telecommunications 1999*, pages 939–944, Chesapeake, VA, 1999. AACE. URL `http://www.editlib.org/p/6607`.

[77] 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.

[78] Fernando Gutierrez, Dejing Dou, Steven Fickas, and Gina Griffiths. Online Reasoning for Ontology-based Error Detection in Text. In *Proceedings of the 13th International Conference on Ontologies, Databases and Application of Semantics (ODBASE)*, pages 562–579, 2014.

[79] Luciano Del Corro and Rainer Gemulla. Clausie: Clause-based open information extraction. In *Proceedings 22nd International Conference on World Wide Web (WWW)*, pages 355–366, 2013. ISBN 978-1-4503-2035-1. URL `http://dl.acm.org/citation.cfm?id=2488388.2488420`.

[80] Bo Liu, Jianqiang Li, and Yu 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.

[81] Johanna Vlker, Peter Haase, and Pascal Hitzler. Learning expressive ontologies. In *Proceedings of the 2008 conference on Ontology Learning and Population: Bridging the Gap between Text and Knowledge*, pages 45–69, Amsterdam, The Netherlands, The Netherlands, 2008. IOS Press. ISBN 978-1-58603-818-2. URL `http://dl.acm.org/citation.cfm?id=1563823.1563829`.

[82] McKay M. Sohlberg, Bonnie Todis, Stephen Fickas, and Laurie Ehlhardt. Analysis of e-mail produced by middle school students with disabilities using accessible interfaces: An exploratory study. *Topics in Language Disorders*, 31(4):1–20, 2011.

[83] Raphael Hoffmann, Congle Zhang, Xiao Ling, Luke Zettlemoyer, and Daniel S. Weld. Knowledge-based weak supervision for information extraction of overlapping relations. In *Proceedings 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (ACL-HLT)*, pages 541–550, 2011. ISBN 978-1-932432-87-9. URL `http://dl.acm.org/citation.cfm?id=2002472.2002541`.

[84] Mitchell Koch, John Gilmer, Stephen Soderland, and Daniel S Weld. Type-aware distantly supervised relation extraction with linked arguments. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014.

[85] Mihai Surdeanu, Julie Tibshirani, Ramesh Nallapati, and Christopher D. Manning. Multi-instance multi-label learning for relation extraction. In *Proceedings Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 455–465, 2012. URL `http://dl.acm.org/citation.cfm?id=2390948.2391003`.

[86] Ben Hachey, Will Radford, Joel Nothman, Matthew Honnibal, and James R. Curran. Evaluating entity linking with wikipedia. *Artificial Intelligence*, 194: 130–150, 2013. ISSN 0004-3702. doi: 10.1016/j.artint.2012.04.005. URL `http://dx.doi.org/10.1016/j.artint.2012.04.005`.

[87] John Philip McCrae. *Automatic extraction of logically consistent ontologies from text corpora*. PhD thesis, University of Advanced Studies, September 2009.