*Article*

JIS

Journal of Information Science
2016, Vol. 42(6) 798–820

DOI: 10.1177/0165551515610989
jis.sagepub.com
SAGE

# A hybrid ontology-based information extraction system

**Fernando Gutierrez**
University of Oregon, USA

**Dejing Dou**
University of Oregon, USA

**Stephen Fickas**
University of Oregon, USA

**Daya Wimalasuriya**
University of Moratuwa, Sri Lanka

**Hui Zong**
University of Virginia, USA

## Abstract

Information Extraction is the process of automatically obtaining knowledge from plain text. Because of the ambiguity of written natural language, Information Extraction is a difficult task. Ontology-based Information Extraction (OBIE) reduces this complexity by including contextual information in the form of a domain ontology. The ontology provides guidance to the extraction process by providing concepts and relationships about the domain. However, OBIE systems have not been widely adopted because of the difficulties in deployment and maintenance. The Ontology-based Components for Information Extraction (OBCIE) architecture has been proposed as a form to encourage the adoption of OBIE by promoting reusability through modularity. In this paper, we propose two orthogonal extensions to OBCIE that allow the construction of hybrid OBIE systems with higher extraction accuracy and a new functionality. The first extension utilizes OBCIE modularity to integrate different types of implementation into one extraction system, producing 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. The second extension is a novel ontology-based error detection mechanism. Following a heuristic approach, we can identify sentences that are logically inconsistent with the domain ontology. Because the implementation strategy for the extraction of a concept is independent of the functionality of the extraction, we can design a hybrid OBIE system with concepts utilizing different implementation strategies for extracting correct or incorrect sentences. Our evaluation shows that, in the implementation extension, our proposed method is more accurate in terms of correctness and completeness of the extraction. Moreover, our error detection method can identify incorrect statements with a high accuracy.

## Keywords

Ensemble learning; error detection; information extraction; machine learning; ontology

**Corresponding author:**
Fernando Gutierrez, Department of Computer and Information Science, University of Oregon, 1585 E 13th Ave, Eugene, OR 97403, USA.
Email: fernando@cs.uoregon.edu

## 1. Introduction

Information Extraction (IE) is the process of automatically transforming natural language text into structured information (e.g. relational databases) [1], by identifying semantic relevant elements such as entities and relationships. However, because of the inherent ambiguity of natural language (e.g. 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 using a domain ontology. An ontology is an explicit specification of a shared conceptualization that represents knowledge through concepts, relationships and individuals [2]. These concepts and properties guide the extraction process in OBIE [3, 4]. However, OBIE can introduce new problems to the extraction process. Creating and maintaining an ontology 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) [5, 6], although some cases require application-specific ontologies [7]. 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 [8]. OBCIE aims to encourage re-usability by modelling 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. *IE 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. concept), making extractions based 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 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).

In this paper, we present two orthogonal extensions to OBCIE. We first extend OBCIE by considering the implementation strategy as a defining characteristic of an information extractor. Independent of the ontological component it represents, an information extractor can be implemented as an *extraction rule* or by applying *machine learning* methods [8]. 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 [9] and Conditional Random Fields [10]. Under this approach, the information extraction process is transformed into a supervised learning task where classification methods and probabilistic models try to identify which elements of a sentence are part of the sought information [8]. 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 have proposed a *hybrid* OBIE system, 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 minimal amount of 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. In other words, for each concept and functionality, we select the implementation that commits fewer extraction errors. In the case of the integration strategy, we propose to integrate the outputs of both implementations under the ensemble learning schema of *stacking*. A top-level classifier is trained with the outputs produced by both implementations of information extractors.

The second extension we propose to the OBCIE architecture is *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. We have extended the OBCIE architecture to detect errors in text. Despite text being a rich source of information (e.g. to identify contradicting statements [11, 12]), text itself is not sufficient to determine the correctness of its content. In order to overcome this limitation, we have proposed an ontology-based mechanism to determine the correctness of document content. Based on ontology debugging, which is the area of research that identifies the origin of inconsistency in an ontology [13], and ontological constraints (e.g. disjointness between concepts), we have proposed an approach that creates axioms that are inconsistent

with respect to the domain ontology. These *domain-inconsistent* axioms are encoded as information extractors that will identify incorrect text. This leads to information extractors being dependent not only on the ontological element they represent, but also on the type of extraction they can perform (i.e. to extract correct or incorrect statements).

Our proposed extensions to OBCIE are based on a new expanded definition of information extractors. We propose that an information extractor encodes a specific ontological element, for a defined functionality, under a specific implementation strategy. In OBCIE, information extractors are self-contained so that the extraction process can be modified with minimal impact on the OBIE system. Our new definition of information extractor needs to keep its modularity to apply to OBCIE. These orthogonal dimensions can be specified for each information extractor independently from the rest. Information extractors in OBCIE contain implementation elements in the extractor by default. On the other hand, the functionality of an information extractor refers to the logical relation between the extractor and the domain, and not between information extractors. With this new characterization, we can have information extractors of different natures in the same OBIE system (i.e. a hybrid OBIE), which allows our proposed extensions to be integrated to OBCIE.

We have evaluated correctness (i.e. precision) and completeness (i.e. recall) for both *selection* and *integration* strategies in order to determine their performance impact in an OBIE system. We have used a synthetic data set generated from real students' answers to a cell biology final exam question. We have compared our method against a single implementation IE and with information extractors with different implementation that do not have any combination strategy (i.e. hybrid OBIE) [14]. Our evaluation experiments show that our proposed strategies have a higher precision than the methods with which we compared them. In the case of recall, both proposed strategies outperform the single- and the multiple-implementations IE in most cases. Integration strategy seems to be more accurate when considering extraction functionality on correct statements, while the selection strategy performs slightly better when extracting from incorrect statements. In any case, extracting from incorrect statements is prone to more errors than when extracting from correct statements.

The remainder of this paper is organized as follows. We provide a brief review of OBCIE in Section 2. We introduce some related works on implementations of IE and error detection in text in Section 3. In Section 4, we present our proposed extensions to OBCIE architecture. We report our experimental setting in Section 5 and the results in Section 6. We conclude the paper by summarizing our contributions and discussing the future work in Section 7.

## 2. Background

### 2.1. Ontology-based Information Extraction

As mentioned, OBIE uses a formal representation of the domain (i.e. ontology) to guide the extraction process [3]. Because of this guidance in the extraction process, OBIE systems have mostly been implemented following a supervised approach [8], that is, labelled data for machine learning or handcrafted patterns, in the case of extraction rules. Given the manual labelling of data and the handcrafted extraction patterns, OBIE can produce very accurate extractions. Some OBIE systems follow a semi-supervised approach to data labelling, where known relationships from a knowledge base help determine training data [15–18].

OBIE allows the possibility of Semantic Annotation because it can connect text with a domain ontology through the extraction process. Semantic Annotation adds meta-data information (e.g. concepts) to text [4]. Nebhi [19] proposed an OBIE system for disambiguating Twitter messages. By combining concepts from Freebase and extraction rules based on dependency trees, Nebhi's approach determines the meaning (and context) of entities mentioned in the messages. Nebhi [20] improved the accuracy of the disambiguation process by replacing the pattern-based approach with a classifier, using Support Vector Machine.

### 2.2. Ontology-based Components for Information Extraction

As mentioned, OBCIE architecture 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 by 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.

In order to provide a clearer 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 1), which interact with the elements of our hybrid approach.

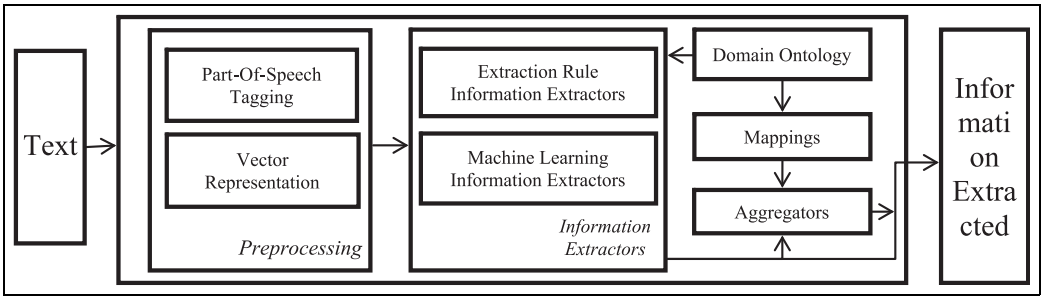


**Figure 1.** Ontology-based Components for Information Extraction.

*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. 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-speec*h 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*. 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].

*2.2.3. 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 $x_s \epsilon X^{\Delta}$, where $X^{\Delta}$ corresponds to a set of sentences from the domain $\Delta$. Let us also consider a concept $c$ from the ontology $O^{\Delta}$ of the same domain $\Delta$. An information extractor $e^c$ will determine the connection between sentence $x_s$ and concept $c$ by resolving the sentence's semantic content $y_s^c$ in the form:

$$e^c(x_s) = y_s^c \tag{1}$$

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 $x_s$ 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 $e^c$ 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 $R_c$ (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 [8]. Based on regular expressions, *extraction rules* capture information by identifying specific elements in the text based on lexical elements (i.e. keywords), syntactical elements (e.g. noun phrases) or both. For example, it is very likely that, in a sentence, the word followed by 'Corporation' is the name of a corporation. *Extraction rules* are, in most cases, simple to design, and they have relatively good performance. However, since they are based on specific cues crafted manually, extraction rules are difficult to generalize [22, 23]. Information extractors can also be based on *machine learning* methods, such as Naive Bayes or Conditional Random Fields. The information extraction process is transformed into a supervised learning task, where classification methods and probabilistic models try to identify which elements of a sentence are part of the sought information [8].

In OBCIE, an information extractor component must contain most (if not all) of 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 terms 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.

*2.2.4. 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].

## 3. Related works

We extend OBCIE through the redefinition of *information extractor*. We have realized that an information extractor is characterized not only by the ontological element it encodes (e.g. concept) but also by the implementation strategy it uses. In most cases, the selection of an implementation strategy in OBIE is a guideline that is applied to all information extractors. We argue that by considering multiple implementations for an information extractor, we can improve the performance of the OBIE system. This improvement can be obtained through the selection of the most accurate implementation or by combining the output of different implementations. An information extractor is also defined by the function it performs: extracting correct or incorrect information (i.e. error detection in text) [7]. The following reviews the most relevant research according to implementation and functionality of an information extractor.

### 3.1. Implementation of information extractors

In general, information extractors can be implemented under two main strategies [2, 8]: as *extraction rules* or based on *machine learning* methods.

*3.1.1. Extraction rules.* Extraction rules capture information by identifying specific syntactic and lexical elements in a text, such as keywords, *part-of-speech* labels and other syntactic structures. Because they are handcrafted from known examples, extraction rules can be very accurate. However, their extraction process can be incomplete (i.e. it can overlook relevant entities in the text) as a result of not generalizing well to unseen examples. Although extraction rules can be defined following regular expression, languages like *SystemT*'s Annotation Query Language (AQL) [24] and *GATE*'s Java Annotation Patterns Engine (JAPE) [25] have been created to specify extraction patterns. These specially designed languages allow the creation of complex extraction rules through the manipulation of annotations.

One of the most well-known sets of extraction rules is *Hearst's extraction patterns* [26]. Hearst has identified a small set of specific linguistic structures (combination of lexical and syntactical elements) that represent a *hyponymy* relationship between two or more entities. A hyponymy relation between two entities $NP_0$ and $NP_1$ refers to membership relations in the form $NP_0$ *is a (kind of)* $NP_1$. For example, if the extraction pattern

$$NP_0 \text{ such as } [NP_1,\ NP_2,\ \ldots, (\text{and}|\text{or})NP_n] \tag{2}$$

is applied to the sentence 'Cetaceans such as whales and dolphins, are marine mammals', it leads to the extraction of the relations *hyponym(whales, cetaceans)* and *hyponym(dolphins, cetaceans)*.

Under the label Open Information Extraction, which is an unsupervised and domain-independent IE, systems like *KnowItAll* [27, 28] and *TextRunner* [10] have extended Hearst's original set of extraction patterns to consider other types of relations. These extensions come from generalizing lexical elements in the extraction patterns to syntactical elements, such as in the case of the extraction rule $NP_0$ *Verb* $NP_1$. In their case study, Banko and Etzioni [10] found that this extended set of extraction rules can cover up to 95% of all binary relationships from their text corpus sample. Although *TextRunner* can extract a large portion of the relationships that are presented in the text, it can produce an erroneous extraction by not identifying correctly the entities that are part of the relation. To reduce these erroneous extractions, *ReVerb* [29] proposes a refinement of the extraction patterns by better defining the syntactical structure that represents the relationship. This refinement can significantly reduce the number of erroneous extractions, allowing *ReVerb* to produce more accurate extractions.

While the set of general extraction rules used by previously mentioned systems should allow the extraction of most types of relationships (from 85% [24] to 95% [10] of all binary relationships), it is possible to extend it by discovering more robust extraction models and patterns. From the initial extraction, it is possible to extend the extraction strategy following an approach similar to a semi-supervised extraction. The initial set of extracted relations is used to learn new extraction patterns [26, 30] or an extraction model [10]. In the case of *OLLIE* [30], the new extraction patterns are templates. From a set of high confidence relations extracted by *ReVerb, OLLIE* analyses the dependency of the extractions to discover more general patterns. By including dependency parsing, *OLLIE* can manage complex relationships, defined by verb phrase structure, between complex entities. This approach leads to higher coverage of the extraction patterns without losing accuracy.

Although most of the previously mentioned systems are domain-independent IE, extraction rules can easily be adapted for OBIE. This adaptation can be obtained by specializing the extraction rule. For example, if, as in the aforementioned example of Hearst's extraction patterns, we replace $NP_0$ with *Cetaceans*, the extraction rule will identify the relations *hyponym(whales, cetaceans)* and *hyponym(dolphins, cetaceans)*. However, no relationships will be identified if this new extraction rule is applied to the sentence 'Motor vehicles such as automobiles, and motorcycles'. If we add the fact that hyponymy is roughly equivalent to the ontological relation between a concept and its super concept, the new rule identifies only subclasses of the concept *cetaceans*.

*3.1.2. Machine learning-based extraction.* An information extractor can also be based on machine learning methods. Under this approach, classification methods and probabilistic models try to identify which elements of a sentence are part of the sought-after information. However, machine learning techniques are *data driven*, so the performance of these methods depends on the quality and quantity of the data used for the training. This training data can come from instances in a knowledge base or from labelled sentences.

Machine learning methods can use, as training data, tuples from a knowledge base, such as FreeBase [31] or Wikipedia's *infoboxes* [15, 16]. Under this approach, we can find systems such as *Snowball* [17], *Distant Supervision* (DS) [31] and the system by Snow et al. [32]. This knowledge base approach intends to identify sentences that make reference to elements of the knowledge base (Figure 2). If a sentence contains a pair of entities that are known to be related (i.e. in a tuple of a knowledge base), the sentence most likely represents the relationship. This relatedness assumption does not always hold, since it is possible for a pair of entities to have sentences representing different relationships [28]. However, it is expected that 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. Before using the sentences for training by any machine learning methods, they are generalized from their textual form into a set of linguistic features, which are presented as a vector. Most systems consider features such as *part-of-speech* [15, 16] and syntactic dependencies [18, 31]. Other types of features used by these systems include named entities [17, 18, 31], words in the sentence [15, 16, 32] and *in-sentence* word location (e.g. middle of the sentence) [15, 16]. Usually, an IE system will use a combination of these features, for example, *Kylin* uses part-of-speech and *in-sentence* location. Once the selected sentences have been transformed, the machine learning method is trained with the transformed sentences. In the case of *Snowball*, this task is mostly reduced to evaluating 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 example sentences. For *Kylin* [15, 16] and DS [31], this task consists of applying a machine learning technique. *Kylin* uses Conditional Random Fields to learn a sequence model from the sentence. *Kylin* uses a large set of features, such as the actual words from the sentence, part-of-speech, whether the word is in the first or second half of the sentence, and also the output of the sentence and document classifiers. In the case of DS, the system uses a multi-class logistic classifier. The output is a relation name and a confidence score.

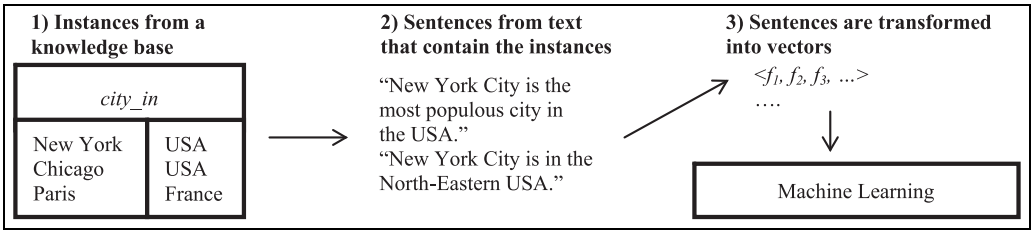


**Figure 2.** Examples from a knowledge base or a database (1) are used to gather sentences (2) representing a specific relationship. These sentences are then transformed into some general representation (3) and used as training for a machine learning method.

The training data set for a machine learning-based information extractor can also come from labelled sentences [7, 8]. In essence, this approach is very similar to the one used by systems that are based on knowledge instances. Sentences are transformed by generalizing features and selecting the most relevant. Then the entities are extracted following a two-phase classification approach. The first classifier determines if a given sentence contains the instances sought. If the sentence does contain the relational instance, then it is passed to a sequence model to extract the relationship. Our machine learning-based information extractors also follow this approach [7]. Using labelled sentences as training data differs, however, from using knowledge base training sets by removing the assumption that all sentences that mention the same entities represent the same relationship. In general, the removal of this assumption can lead to more accurate information extractors. By not having this assumption, sentences that might have been considered positive examples for the information extractor are no longer included, or they are used as negative examples.

*3.1.3. Hybrid implementations of IE.* As mentioned, for most IE systems the selection of an implementation strategy is a design guideline that is extended to all information extractors of the system. However, there are some IE systems that will combine both implementation strategies into a hybrid extraction mechanism. A hybrid approach to IE is to use rule-based extraction to generate training sets for a machine learning-based extractor. *TextRunner* follows this approach by using a small set of rules, similar to Hearst extraction patterns [26], to generate a labelled set. A Naive Bayes extractor [9] and Conditional Random Fields extractor [10] use this labelled data for training the actual extractor, which is applied to the text. The system described by Yakushiji et al. [33] follows a similar approach, with dependency-based extraction rules to capture instances of protein interaction. The instances are later used to train a Support Vector Machine classifier.

Another hybrid approach to IE is to evaluate the quality of the extractors. Systems such as *OLLIE* [29] and *Snowball* [17] use a confidence measure to promote rule-based extractors that are more accurate. Both systems generate large sets of rule-based extractors based on in text elements such as named entities and dependency trees. While Snowball uses a set of weights to determine the overall confidence of each rule-based extractor, *OLLIE* associates a probability to each extractor that represents its precision (i.e. correctness of the extraction). If an extractor has a high confidence value, then it is included into the system.

Although these hybrid methods might seem similar to our proposed approach because they incorporate both types of implementation, they differ in how the extraction system applies the implementations. As mentioned, these systems use the implementation in a *sequential* fashion, that is, a rule-based extractor provides a training set for the machine learning method, which *then* is finally applied to the text. On the other hand, our hybrid method *simultaneously* (or, in *parallel*) extracts information with both types of implementations.

## 3.2. Error detection in text

Research in IE has mainly been used for data documents with curated content (i.e. peer-reviewed scientific documents) [4, 26, 34]. This approach has led to the assumption that the information contained in the documents is correct. In other words, the information obtained from an information extractor is semantically correct. However, even in these specific cases of curated documents, correctness cannot be guaranteed [35, 36]. Considering that current research in IE is moving towards domains where quality of content is not controllable, such as the Internet [10, 26, 37], we need to consider the presence of incorrect statements in our data set.

By incorrect statement, we refer to a natural language statement that is either false or contradicts the domain knowledge. Most efforts toward determining semantic errors in text (i.e. error extraction functionality) have indirectly addressed the problem as a divergence from correctness, or as the presence of a contradiction.

*3.2.1. Error as divergence.* The evaluation of the correctness of a text is the main goal for the education-related research area of *automatic text evaluation*. Because summaries and essays enhance long-term retention of information [38], it is an ideal tool when compared with other evaluation mechanisms (e.g. fill-in the blanks) [39]. If performed manually, the feedback produced by the text evaluation process would take too long to reach the student. This delay in feedback significantly reduces the effectiveness of using summaries and essays as an evaluation tool. Advancements in Natural Language Processing (NLP) have lead to three main approaches in automatic text grading. The first approach is based on the identification of coincident words and n-grams [40]. n-Gram co-occurrence methods have been shown to be adequate for automatic text grading, especially when evaluating characteristics such as the fluency of the text. The second approach uses NLP techniques, with the most popular being Latent Semantic Analysis (LSA) [41, 42]. LSA treats the text as a matrix of word frequencies and applies Singular Value Decomposition to the matrix to find an underlying semantic space. Based on the distance between the vector representations of the student's document and a gold standard (i.e. correct text) in this semantic space, the similarity of the documents is estimated, which can then be transformed into a grade. LSA has been shown to be quite accurate when compared with human grading. Finally, the third approach is based on Information Extraction (IE) [43, 44], which intends to capture the underlying semantics of the text. Because automatic text grading needs to determine the meaning (i.e. semantics) behind the student's writing, IE is a natural choice for analysing text. It uses NLP tools to identify relevant semantic elements from the text, such as concepts, individuals and relationships. These semantic elements can provide a structural representation of the information in the text.

All of the previously mentioned methods share a common characteristic: they do not have an effective way of determining what is incorrect in a text. These methods can only determine what is less correct, based on how dissimilar they are to the gold standard or based on missing elements. Methods such as n-gram co-occurrence and IE-based can identify specific elements in the text from previously known patterns (e.g. n-gram methods) or expected domain elements (e.g. IE regular expression patterns). Therefore, to find incorrect text, these methods would require a reference of incorrectness, such as text with incorrect statements or incorrect facts of domain knowledge. However, because the incorrectness of a statement can be originated by many different factors, this reference of incorrectness would need to be very large to provide useful coverage of content, which would be impractical. In the case of LSA-based methods, determining if a text is incorrect is even more difficult because the correctness of the text is measured regarding its similarity to a gold standard. It is possible to argue that a low similarity is an indication of incorrectness. However, even a correct text can obtain a low similarity with respect to the gold standard if it is written in an unexpected fashion or contains more information [45].

*3.2.2. Error as contradiction.* Another alternative to capture incorrectness in text is by identifying logical contradiction. Since we have defined incorrect text as a false or contradicting statement, it is reasonable to consider logic as a mechanism to identify it. Through logic, the truth value of a statement (i.e. a statement is true or false) can be determined from a set of facts. Logic consequence in text is studied by *textual entailment*, which intends to determine if a body of text is a logical consequence of another body of text [46]. However, this research area is mostly focused on determining logical consequence between a pair of statements [45] rather than the truth value of a statement.

Through logic, we can also determine if a statement is a contradiction. Contradiction in text is studied by *Contradiction detection* [11], which is a special case of textual entailment. *Contradiction detection* tries to identify pairs of sentences that are very *unlikely* to be true at the same time. It can use syntactic and lexical elements of the text [11], or background statistical knowledge [12] to determine if a pair of sentences contradicts each other. However, with only information from the text itself to support the validity of the pair of contradicting statements, *contradiction detection* cannot determine with certainty which of the statements is false.

## 4. Methodology

In this paper, we present two extensions to the OBCIE architecture that lead to a hybrid approach to the extraction process. The first extension increments the accuracy of the OBIE process by combining information extractors with different implementation strategies. The second extension adds the functionality to identify *domain-incorrect* elements in the text. We use a heuristic approach that generates domain inconsistent statements that we then use to create information extractors that can identify incorrect text.

In order to integrate these extensions 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 [8, 16] it extracts. We have extended the definition of information extractors by considering two new fundamental and

orthogonal aspects (i.e. dimensions): function performed by the information extractor [7] and implementation of the information extractor [14]:

$$e_i^{cf}(x_s) \tag{3}$$

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 ($i$). These new dimensions (functionality and implementation) allow us to include information extractors of different concepts, using different implementations and performing different functionalities, into the same extraction system, that is, 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 function. 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. In the case of functionality, the correctness of a statement is based on its logical relation (e.g. logic contradiction) with the domain and is not affected by other statements. On the other hand, this new characterization allows relations to be established between information extractors. For a concept and functionality, there is one information extractor based on machine learning, and another based on extraction rules.

### 4.1. 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 [30], and the features and restrictions the extraction process as whole might have [15, 17]. From the information extractor $e_i^{cf}(x_s)$, 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:

$$e_i^{cf}(x_s) \approx y_s^{cf} \tag{4}$$

Further, as 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 [7, 8, 15–17, 47]. This behaviour 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 an 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, that is, a hybrid OBIE. We explore the impact that a hybrid OBIE can have when extracting information as part of an evaluation system [7]. We found that improvements were observable even when choosing an arbitrary *configuration*, for example, 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, for example, selecting the worst implementation strategy for each concept [7].

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.

*4.1.1. 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 and functionality.

At the beginning of Section 4.1, we have defined the output of an information extractor as an approximation to the semantic content $y_s^{cf}$ of sentence $x_s$ with respect to concept $c$ and functionality $f$. An implementation $e_i^{cf}(x_s)$ will produce an accurate extraction if its difference from 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 and functionality, we estimate the error across all sentences as:

$$E_i^{cf}(S) = \sum\nolimits_{s \in S} \left| e_i^{cf}(x_s) - y_s^{cf} \right| \tag{5}$$

where $E$ is the accumulated error over the set of sentences $S$, of implementation $i$, and concept $c$ of the domain ontology $O^{\Delta}$. We will consider the output of the information extraction to be $e_i^{cf}(x_s) \in \{0, 1\}$, and the semantic content of the sentence to be $y_s^{cf} \in \{0, 1\}$. So, when there is no difference between the information extractor's output and the semantic content of the sentences (i.e. when $e_i^{cf}(x_s) - y_s^{cf} = 0$), 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 $y_s^{cf} = R_{cf}(a, b)$. 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 $y_s^{cf} \in \{0, 1\}$.

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 $E_i^{cf}$:

$$I^c(S) = \underset{i}{\operatorname{argmin}} (E_i^{cf}(S)) \tag{6}$$

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

To extend the selection of information extractors to all concepts, we pick the information extractors for each concept and functionality with implementation $I(c, f, S)$:

$$\cup_{\forall c \in O^{\Delta}} e_{I(c,f,S)}^{cf}(x_s) \tag{7}$$

The implementation that has the minimum number of errors will be selected as part of the OBIE. This selection leads to having a hybrid OBIE system because, for concepts $c, c' \in O^{\Delta}$ , 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 (and functionality) level, the error of each information extractor is minimized, which leads to a smaller total error.

*4.1.2. Integration strategy.* The 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) [6]. 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 case wherein it is difficult to select one type of implementation because the 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 [48]. 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 votes with different importances (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 determine which of the methods is more important (i.e. a 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* is 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). In other words, the set of outputs of the bottom level classifiers creates 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 [49]. The input for the top-level classifier will be:

$$e^{cf}_{ML}(x_s), e^{cf}_{ER}(x_s), y^{cf}_s \tag{8}$$

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 $c$ and function $f$. 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.

## 4.2. Ontology-based error detection

The second proposed extension to OBCIE corresponds to the inclusion of *error detection* based on a domain ontology. We propose that an information extractor cannot perform extractions based only on a specific concept $c$, but also on a domain-related correctness *functionality* $f$. In traditional information extraction, the information extractors perform the functionality of extracting statements that are correct with respect to the domain knowledge. In contrast, error detection is the functionality of an information extractor to identify domain-incorrect statements. In other words, an information extractor $e^{cf}(x_s) = y^{cf}_s$ is bound not only to a specific concept $c$, but also to the type of functionality $f$.

We have been the first group to propose the use of domain knowledge as a mechanism to determine the correctness and incorrectness of text content, by applying OBIE for autonomous text evaluation [7, 50]. 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 statement itself is not sufficient to conclude if it is incorrect. We need to know *facts* from the domain to verify if the statement from the text is false or not. The domain knowledge, represented through an ontology, can provide us with the frame of *what is correct* of the domain. Therefore, combining this *correct knowledge frame* with logic, we resolve the correctness (or incorrectness) of the text's content. For each concept and property of the ontology, we define an information extractor to identify correct instances, and we define an information extractor to detect incorrect instances.

We also have determined types of functions an information extractor could have: extraction of correct statements and extraction of incorrect statements. Next, we provide a more detailed discussion of each function.

*4.2.1. Extracting correct statements.* Originally, when we proposed ontology-based error detection, we defined a statement as correct if it was consistent with the domain ontology [7]. However, consistency is not a sufficient guarantee of correctness with respect to the domain ontology. A statement does not need to be part of the domain to be consistent with it. Further, if a statement is completely unrelated to the domain, it is more likely that the statement will not violate any

constraint of the domain. We need a more precise definition of a correct statement that takes into account whether the extraction is part of the ontology. In this work, we use a more strict definition of domain correctness: a statement is correct if it is a logical consequence of the domain (i.e. entailment) [50]. In other words, a *correct statement* from a text is the logical consequence of the concepts and properties that define the domain. Therefore, to extract correct statements, we use axioms that are explicitly stated in the ontology, or that can be inferred from the ontology. For example, we have as part of our domain knowledge the following axioms:

$$K_1 = \{\forall xy(Professor(x) \wedge Teaches(x,y)) \rightarrow (UnderGradStudent(y) \vee GradStudent(y)), \\ \forall x(UnderGradStudent(x) \vee GradStudent(x)) \rightarrow Student(x)\} \tag{9}$$

From $K_1$, we know that if a *Professor* teaches a person, it must be an *undergraduate* or a *graduate* student and that both *undergraduate* and *graduate* are *Students*. These axioms can be encoded into information extractors to identify correct statements. The information extractors can be implemented as extraction rules in the form of regular expression patterns (e.g. Perl regular expression):

$$\$\_ = \sim /(\mathrm{P}|\mathrm{p})\mathrm{rof}(\backslash.|\backslash\mathrm{w}+)\,(\backslash\mathrm{w}+)\ \mathrm{teach}(|\mathrm{es})/, (\backslash\mathrm{w}+)/ \tag{10}$$

$$\$\_ = \sim /(\mathrm{Phd}|\mathrm{master}|\mathrm{college})\ \mathrm{student}\ (\backslash\mathrm{w}+)/ \tag{11}$$

In essence, this is the traditional definition of OBIE: extract elements of the domain ontology from the text. However, this definition of correctness also considers the implicit elements of the ontology: entailed elements of the ontology. From the original two axioms of $K_1$, we can infer a third axiom:

$$\frac{\forall xy(Professor(x) \wedge Teaches(x,y)) \rightarrow (UnderGradStudent(y) \vee GradStudent(y)) \\ \forall x(UnderGradStudent(x) \vee GradStudent(x)) \rightarrow Student(x)}{\forall xy(Professor(x) \wedge Teaches(x,y)) \rightarrow Student(y)}, \tag{12}$$

which tells us that, if a *Professor* teaches a person, it must be a *Student*. This new axiom can also be encoded as an information extractor to identify correct statements:

$$\$\_ = \sim /(\mathrm{P}|\mathrm{p})\mathrm{rof}(\backslash.|\backslash\mathrm{w}+)\,(\backslash\mathrm{w}+)\,\mathrm{teach}(|\mathrm{es})\,((\mathrm{Phd}|\mathrm{master}|\mathrm{college})\ \mathrm{student}|)\,(\backslash\mathrm{w}+)/ \tag{13}$$

*4.2.2. Extracting incorrect statements.* A natural consequence of the definition of correct statement is the definition of incorrect statement. We consider a statement to be incorrect if it is a logical contradiction (i.e. inconsistency) of some aspects of the domain ontology. 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. [51]. 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 and they 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. For example, let us consider the following two axioms from our domain knowledge $K$:

$$K_2 = \{\forall xy(Professor(x) \wedge Teaches(x,y)) \rightarrow Student(y), \forall x Professor(x) \leftrightarrow \neg Student(x)\} \tag{14}$$

From the first axiom, we obtain that *Professor* cannot teach a *non-Student* ($\forall xy(Professor(x) \wedge Teaches(x,y) \wedge \neg Student(x)) \rightarrow \bot$). From the second axiom, we know that a *Professor* is not a *Student*, and a *Student* cannot be a *Professor*. By combining these two axioms, we can construct an axiom that is inconsistent with our domain knowledge:

$$\forall xy(Professor(x) \wedge Teaches(x,y)) \rightarrow Professor(y) \tag{15}$$

The axiom by itself is not inconsistent. The contradiction occurs when adding this new axiom to our domain knowledge (i.e. $K_2$). In other words, the new axiom is domain-inconsistent. To identify incorrect statements, we use these *domain-*

*inconsistent* axioms, encoded into information extractors, such as pattern-based extraction rule (e.g. Perl regular expression):

$$\$\_ = \sim /(\mathrm{P}|\mathrm{p})\mathrm{rof}(\backslash.|\backslash\mathrm{w}+)(\backslash\mathrm{w}+)\ \mathrm{teach}(|\mathrm{es})/(\mathrm{P}|\mathrm{p})\mathrm{rof}(\backslash.|\backslash\mathrm{w}+)(\backslash\mathrm{w}+)/ \quad (16)$$

It must be mentioned that *domain-inconsistent* axioms can also be implemented as machine learning-based information extractors. However, there is no need to modify any machine learning technique for the purpose of error extraction. Because machine learning-based information extractors learn a model based on training data, incorrect statements can be labelled into the training data set.

## 5. Experiments

In the following section, we provide details regarding the evaluation of our proposed hybrid OBIE system. We describe the construction of the domain ontology and the generation of the synthetic data set used in the evaluation. Then we explain the process of creating the information extractors with different implementations and different functionalities. Finally, we present the comparison methods and metrics used to evaluate our proposed hybrid OBIE system.

### 5.1. 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 is:

> 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.

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

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

It can be argued that an incorrect answer can be formed by correct sentences if these sentences mention concepts and relationships that are unrelated to the question. In the present work, we do not intend to determine the correctness of the text as whole (complete answer). We will only focus on the correctness of each sentence independently. For this reason, 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.

### 5.2. Domain ontology

Currently, there are a large number of biology-related ontologies that are available. As mentioned in Section 2.1, 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

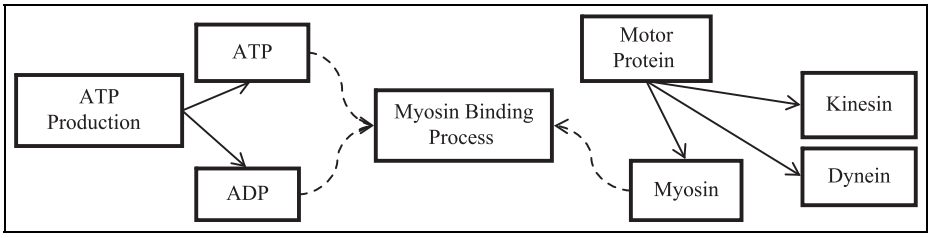


**Figure 3.** Graphical representation of a section of the ontology development for this work. In the figure, continuous lines represent IS_A relationships between concepts (e.g. Myosin *is a* Motor Protein) while segmented lines represent property relationships between concepts (e.g. Myosin *has a* Myosin Binding Process).

**Table 1.** Statistical information about the ontology

| Element type | Number of elements |
|---|---|
| Concepts | 17 |
| Relationships | 10 |
| Subclass relationships | 3 |

Thesaurus) that contain the concepts (e.g. *myosin* and *ATP*) which are required to analyse the students' answers. However, these ontologies do not offer all of the necessary relationships that are required to analyse 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).

For this work, we have designed an application-driven ontology (Figure 3). Although we could have opted to extend one of the available ontologies with the relationships needed to analyse 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 sufficient domain knowledge to analyse the arguments of the answer, that is, 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, for example, *Myosin Binding Process* in Figure 3. These intermediary concepts have led the ontology having a rather *sparse* structure, with few concepts in an *ISA* (i.e. subclass) relationship (Table 1).

## 5.3. 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 evaluate our functionality extension, 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.

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

**Table 2.** Number of template sentences for each concept given its functionality

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

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 2). 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.

From the original data set, we have observed that 22% of the students answered the question correctly and provided a correct justification. From the rest, 60% answered the question incorrectly, and only 3.8% of all the students' answers were completely incorrect. With these numbers in mind, we have created data sets containing 1000 instances with error levels of 20, 30, 40, 50, 60, 70, 80 and 90%.

## 5.4. Preparation of information extractors

In the following section, we provide details of the design of the information extractors for the study case. These details should provide insight on how we created the information extractors under different implementation strategies, for different functionalities.

*5.4.1. Functionality details.* As mentioned in Section 4.2.1, the information extractors for the correct functionality are obtained by selecting relevant axioms from the ontology. These axioms represent relationships between the four main concepts of the ontology:

$$\frac{\begin{array}{c}\forall xy\, ETC(x) \wedge break(x) \rightarrow ATP(y) \wedge reduce(y) \\ \forall xy\, ATP(x) \wedge reduce(x) \rightarrow ADP(y) \wedge increase(y) \\ \forall xy\, ATP(x) \wedge reduce(x) \wedge ADP(y) \wedge increase(y) \rightarrow ATP - ADP\_exchange(x, y) \\ \forall xyz\, ATP - ADP\_exchange(x, y) \wedge Myosin(z) \rightarrow stuck(z)\end{array}}{\forall xy\, ETC(x) \wedge break(x) \wedge Myosin(y) \rightarrow stuck(y)}. \tag{17}$$

The last logic clause corresponds to the correct answer, which is a logical consequence of the relationships between the concepts of the domain.

In the case of the information extractors for incorrect functionality, we generate *domain-inconsistent* axioms from the ontology (Section 4.2.2). Based on our understanding of the types of errors that could appear in the text, we can follow a broad range of strategies to the generation of inconsistent axioms [7, 14]. This approach for error detection has the advantage that it can target a specific set of errors. In this work, inconsistent axioms were generated from *ontological properties* (e.g. A *has* B) rather than *taxonomical relationships* (e.g. A *is a subclass of* B). For example, if we state that Myosin will *fall* instead of *staying stuck* (i.e. $\forall x\, fall(x) \leftrightarrow \neg stuck(x)$), we obtain the an incorrect answer, which is also inconsistent with the domain:

$$\frac{\begin{array}{c}\forall xy\, ETC(x) \wedge break(x) \wedge Myosin(y) \rightarrow stuck(y) \\ \forall x\, fall(x) \leftrightarrow \neg stuck(x)\end{array}}{\forall xy\, ETC(x) \wedge break(x) \wedge Myosin(y) \wedge fall(y) \rightarrow \bot} \quad (18)$$

*5.4.2. Implementation details.* In general, the creation of individual information extractors mostly follows the same considerations for single implementation (i.e. traditional OBIE), multiple implementations [14] and our proposed combination strategies.

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 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 and functionality is approximately 200 instances. This allows a good insight into instances that could be expected for each concept and functionality while still being manageable. The following extraction rule identifies the consequence of the *breakdown* of the *electron transfer chain* (ETC):

$$\$\_ = \sim /(\text{It}|\text{Myosin}).+(((\text{stay}|\text{get})\text{stuck})|(\text{bind}))/. \quad (19)$$

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:

$$\$\_ = \sim /(\text{ETC}|\text{electron transport chain}).+(\text{stops}|\text{loss}|\text{less}|\text{required}).+(\text{ATP})/. \quad (20)$$

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 incorporated the two-phase extraction approach previously described (Section 3.1.2), which is also seen in *Kylin* [15, 16] and in the study case of OBCIE [8]. 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)[52]. 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 Naive Bayes classifier, which is a popular option for text classification because of its simplicity and good general performance [53]. 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 [54]) 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 [33] or Maximum Entropy [15, 16]. 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 [8–10, 15, 16].

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.

## 5.5. Evaluation metrics

To evaluate and compare our proposed hybrid approach, we will use the metrics of precision, recall and $F_1$ measure. Precision indicates the correctness of the extraction, while recall indicates the completeness. The *F1* measure provides the harmonic mean between precision and recall.

$$\text{Precision} = \frac{\text{Correctly extracted}}{\text{All extractions}} \tag{21}$$

$$\text{Recall} = \frac{\text{Correctly extracted}}{\text{All instances}} \tag{22}$$

$$F_1 \text{ measure} = \frac{2 * \text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}} \tag{23}$$

These traditional evaluation metrics do not consider the semantic relation between domain elements when evaluating the correctness and completeness of the extraction process [3]. An extraction (or label) is either correct or incorrect. Metrics such as Balanced Distance Metric [55] and Learning Accuracy [56] 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.

However, the ontology used in the evaluation of our hybrid approach is mostly *flat* (very limited hierarchical relationships). This situation leads to the extraction process to be *hit-or-miss*: the information extractor correctly identifies a relationship or not. Since our study case does not need to consider possible semantic relatedness between extraction and correct labelling, we will use as evaluation metrics precision, recall and $F_1$ measure.

### 5.6. Comparison methods

In this work, we have proposed two extensions to the OBCIE architecture: the first improves the accuracy of the extraction process, while the second adds a new functionality to the extraction process. To evaluate the first extension, we will consider other strategies for comparison. To determine the impact of our combination strategies, we need to compare against currently used methods. We will compare 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. 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 16 information extractors by combining all four possible concepts (*Myosin, ETC, ATP* and *ADP*), two implementations (i.e. machine learning and extraction rules), and two functions (i.e. extracting correct and incorrect statements).

Because error detection is a new functionality for IE, there is no other method for comparison. For this reason, we will present evaluation metrics (precision, recall and $F_1$ 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.

## 6. Results and discussion

In the following section, we present and discuss the results of the evaluation of our proposed combination 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 (and functionality). With these three values (best, average and worst), it is possible to obtain a reasonable understanding of the performance behaviour of a configuration. The results are presented in detail with respect to the amount of errors in the data set, which provides an insight into how errors can affect the extraction process. We also provide a general view of the experimental results, which allows a more accessible comparison between methods. For clarity, we have separated our results based on the extraction functionality (i.e. extracting correct statements and error extraction). This separation allows visualizing the effect that an implementation can have over functionality, and it also permits an easier comparison between functionalities.

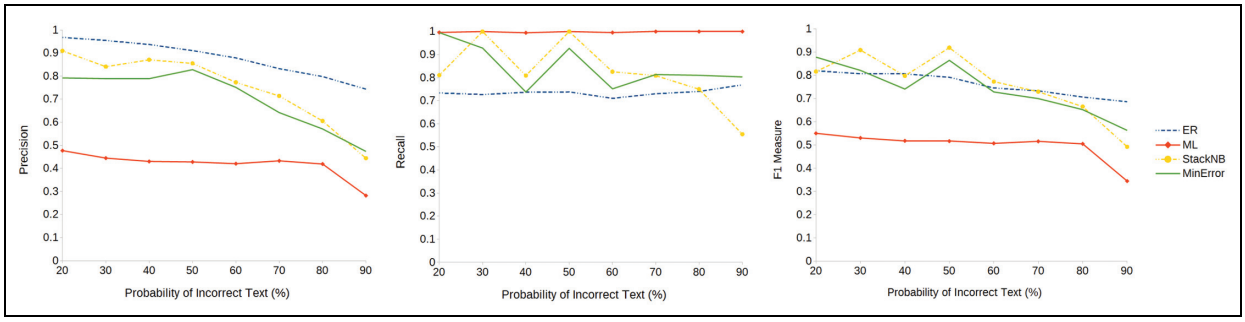


**Figure 4.** Precision, recall and $F_1$ measure for information extractors under different levels of error in text, with single implementation (ER and ML), and multiple implementations with our proposed combination strategies (*MinError* and *StackNB*) with the functionality of extracting correct statements.

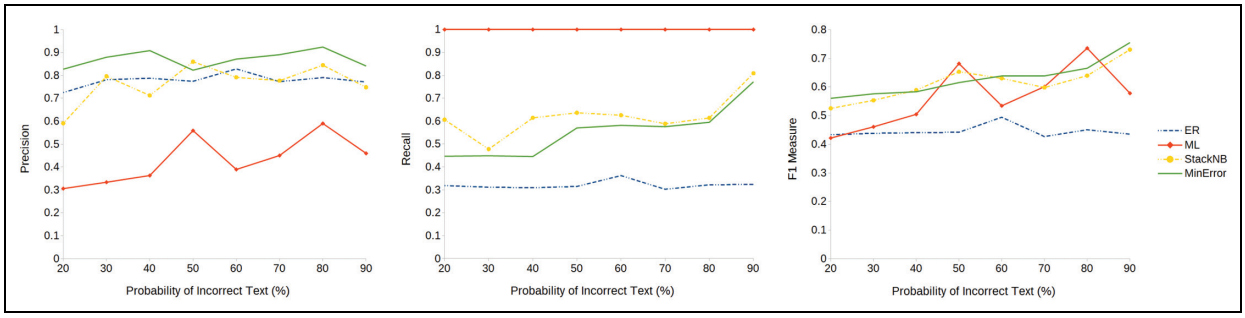


**Figure 5.** Precision, recall and $F_1$ measure for information extractors under different levels of error in text, with single implementation (ER and ML), and multiple implementations with our proposed combination strategies (*MinError* and *StackNB*) with the functionality of extracting incorrect statements.

## 6.1. Error level and extraction

First we compare our proposed combination methods by considering the quantity of incorrect statements there are in the text. As previously mentioned, we have generated data sets with different levels of error: 90, 80, 70, 60, 50, 40, 30 and 20% of error. We have applied our proposed methods and those methods used for comparison to the eight data sets. The results are presented in Figure 4 for correct statement extraction and in Figure 5 for error extraction.

In general, as the number of errors increases (higher probability of error in an answer), the precision of all the methods is reduced. This trend can be explained by the similarity between correct and incorrect sentences of the same concept, and by the amount of training data. As the amount of examples of correct statements of a concept is reduced, it is more likely that the generalization of machine learning models and the extraction rule patterns will see errors as correct. This trend applies inversely for extraction of errors: as the error level increases in the data set, the extraction of errors becomes more precise. In contrast, the completeness of the extraction seems not to be affected by the level of error.

We can observe that, in the case of functionality of extracting correct statements, the extraction rule implementation (ER) can produce a more precise extraction, while the machine learning-based implementation (ML) can produce a more complete extraction. Although *StackNB* and *MinError* have lower precision than ER, and lower recall than ML, they can produce a more balanced extraction, which leads to the higher F1 measure. On the other hand, when considering the functionality of extracting incorrect statements, we can observe that *MinError* produces the most precise extraction, with *StackNB* and ER close in performance. In the case of recall, ML produces a complete extraction, with a significant difference from the rest of the implementation strategies. In term of $F_1$ measure for error extraction, *MinError* is slightly better than *StackNB* in most cases.

It must be mentioned that, although ML has a perfect recall, the actual extraction has a significant number of errors. This situation occurs because ML produces an over-generalized extraction model.

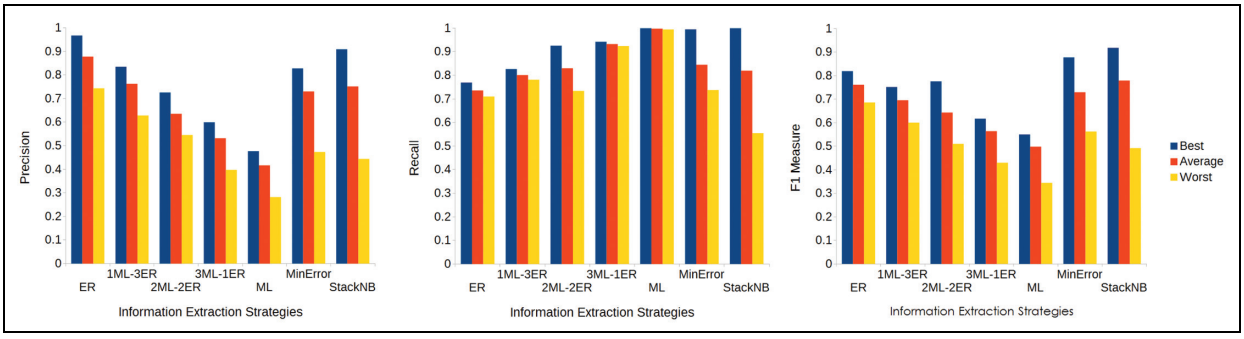


**Figure 6.** Precision, recall and $F_1$ measure for information extractors with single implementation (ER and ML), multiple implementations without a combination strategy (1ML-3ER, 2ML-2ER and 3ML-1ER), and multiple implementations with our proposed combination strategies (*MinError* and *StackNB*) with the functionality of extracting correct statements.

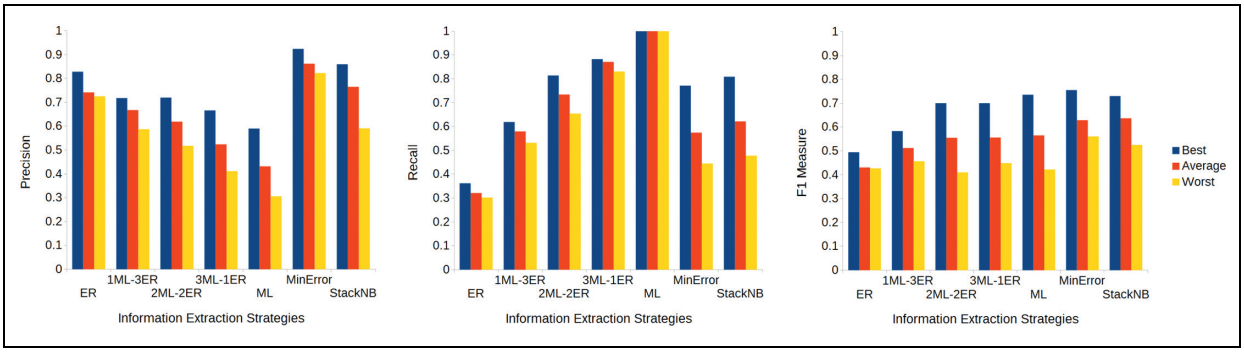


**Figure 7.** Precision, recall and F1 measure for information extractors with single implementation (ER and ML), multiple implementations without a combination strategy (1ML-3ER, 2ML-2ER and 3ML-1ER), and multiple implementation with our proposed combination strategies (*MinError* and *StackNB*) with the functionality of extracting incorrect statements.

## 6.2. Overall extraction accuracy

In order to provide better overview of the performance of the information extractor's functions (i.e. extracting correct statements and errors), we provide average precision, recall, and $F_1$ measure of each implementation strategy separated by functionality (Figures 6 and 7).

Both combined methods obtain, in general, better performance than the *pure* methods and the mixed methods that do not have any combination strategy. However, both combined strategies depend on the quality of the extraction performed 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. This issue can be also seen in both Figures 6 and 7. In the case of the correct statement extraction function, we see that, although the combined strategies outperform the other methods in the case of best performance, their average performance is close to (precision in Figure 4) if not worse than (recall in Figure 4) 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 behaviour affects the combined strategies in different ways when integrated with extraction rule performance. In the case of the error extraction function, we can see more clearly that the combined strategies perform better than the rest in terms of precision and $F_1$ measure. In the case of recall, similarly as for the correct statement extraction, machine learning dominates.

From Figures 6 and 7, it seems that our proposed combination strategies are sensitive to the performance of the underlying implementations. The effect seems to differ from correct statement to error extraction functionality. In the case of correct statement extraction, the performance of the worse implementation seems to dominate. On the other hand, our proposed combination strategies for error extraction produce a more average performance between the underlying implementations.

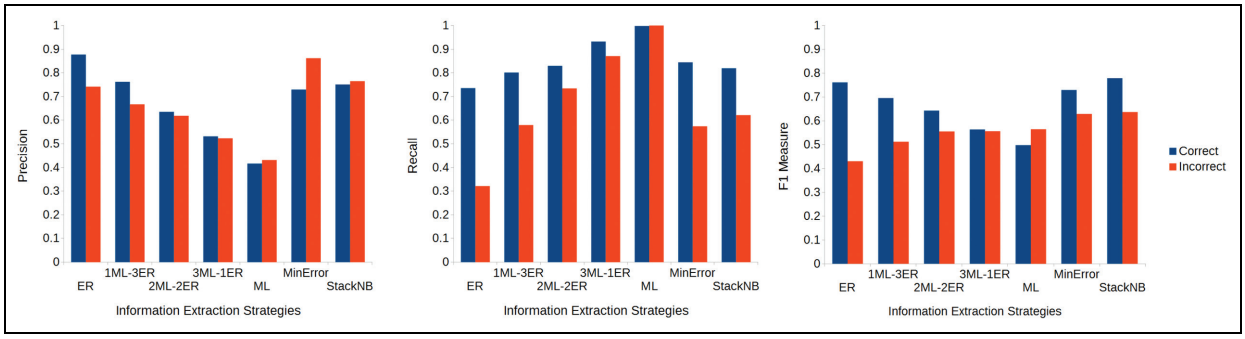


**Figure 8.** Comparison of correct statement extraction and error extraction functionality in terms of precision, recall and $F_1$ measure for information extractors with single implementation (ER and ML), multiple implementations without a combination strategy (1ML-3ER, 2ML-2ER, and 3ML-1ER), and multiple implementations with our proposed combination strategies (*MinError* and *StackNB*).

Finally, Figure 8 compares the average performance of each configuration given its functionality. In general, information extractors that extract correct statements have a higher precision, recall and $F_1$ 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 2 that there are 12 types of correct sentences for Myosin, while there 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 in the training set. This leads not only to the machine learning-based extractor having to consider a wider range of types, but also to not all available types being available enough or frequent enough in the training set to be considered relevant.

Figure 8 also shows that the integration strategy *StackNB* performs better for correct statement extraction, while *MinError* slightly outperforms the rest for the error extraction.

## 7. Conclusions

In the present work, we propose a hybrid OBIE framework through two extensions to the OBCIE architecture, which leads to a more accurate and complete extraction process. The first extension 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 second extension to OBCIE provides the system with the capability to detect errors in text. By identifying what axioms could be inconsistent with the domain ontology, we create information extractors that identify logic errors. The *domain-inconsistent* axioms are determined following the heuristic of violating ontological constraints.

We have applied our hybrid OBIE system to identify the correct and incorrect statements to a set of synthetic data sets with different levels of errors. Our hybrid system can identify both correct statements and errors with high and balanced precision and recall measures. Furthermore, we have found that the combination of information extractors that have different implementations can obtain a higher precision and recall than using only one type of implementation. We also found that error extraction is more complex than the extraction of correct statements, which leads to high variability in the performance of information extractors. The experimental results show that this variability can be reduced through the use of a hybrid configuration.

As future work, we would like to refine our hybrid approach. In the case of functionality, we would like to determine a more formal and efficient mechanism to define information extractors for errors. We believe that research in ontology debugging can provide us with insight into a method that could guarantee a complete analysis of the consistency of each sentence. In the case of combining strategies, we would like to see if there are alternative strategies that would allow a more accurate combination of information extractors, such as the use of meta-features for stacking. We would also like

to determine new extraction capabilities that can be obtained by combining information extractors that have different functions and implementations. For example, we want to see whether combining information extractors of the same concept but different functionality can lead to a more accurate extraction.

## Acknowledgements

We thank Adam Martini for his input in this work.

## Funding

This research is partially supported by the National Science Foundation grant IIS-1118050 and grant IIS-1013054. The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the NSF.

## References

[1] Jurafsky D and Martin JH. *Speech and language processing: An introduction to natural language processing*, 2nd edn. Englewood Cliffs, NJ: Prentice Hall, 2008, p. 725.
[2] Gruber TR. A translation approach to portable ontology specifications. *Knowledge Acquisition* 1993; 5(2): 199–220.
[3] Wimalasuriya DC and Dou D. Ontology-based information extraction: An introduction and a survey of current approaches. *Journal of Information Science* 2010; 36: 306–323.
[4] Srinivasan P and Qiu XY. GO for gene documents. *BMC Bioinformatics* 2007; 8(9): S3.
[5] Gutierrez F, Wimalasuriya DC and Dou D. Using information extractors with the Neural ElectroMagnetic Ontologies. In: *OTM international conference on ontologies, databases and application of semantics*, 2011, pp. 31–32.
[6] Wimalasuriya DC and Dou D. Using multiple ontologies in information extraction. In: *ACM conference on information and knowledge management*, 2009, pp. 235–244.
[7] Gutierrez F, Dou D, Fickas S and Griffiths G. Providing grades and feedback for student summaries by ontology-based information extraction. In: *ACM conference on information and knowledge management*, 2012, pp. 1722–1726.
[8] Wimalasuriya DC and Dou D. Components for information extraction: Ontology-based information extractors and generic platforms. In: *ACM conference on information and knowledge management*, 2010, pp. 9–18.
[9] Banko M, Cafarella MJ, Soderland S, Broadhead M and Etzioni O. Open information extraction from the web. In: *International joint conference on artificial intelligence*, 2007, pp. 2670–2676.
[10] Banko M and Etzioni O. The tradeoffs between open and traditional relation extraction. In: *Annual meeting of the association for computational linguistics with the human language technology*, 2008, pp. 28–36.
[11] de Marneffe MC, Rafferty AN and Manning CD. Finding contradictions in text. In: *Annual meeting of the Association for Computational Linguistics with the human language technology*, 2008, pp. 1039–1047.
[12] Ritter A, Downey D, Soderland S and Etzioni O. It's a contradiction – no, it's not: A case study using functional relations. In: *ACL conference on empirical methods in natural language processing*, 2008, pp. 11–20.
[13] Flouris G, Manakanatas D, Kondylakis H, Plexousakis D and Antoniou G. Ontology change: Classification and survey. *The Knowledge Engineering Review* 2008; 23(2): 117–152.
[14] Gutierrez F, Dou D, Fickas S, Martini A and Zong H. Hybrid ontology-based information extraction for automated text grading. In: *IEEE conference on machine learning and applications*, 2013, pp. 359–364.
[15] Wu F and Weld DS. Autonomously semantifying Wikipedia. In: *ACM conference on information and knowledge management*, 2007, pp. 41–50.
[16] Wu F and Weld DS. Automatically refining the Wikipedia infobox ontology. In: *International conference on World Wide Web*, 2008, pp. 635–644.
[17] Agichtein E and Gravano L. Snowball: Extracting relations from large plain-text collections. In: *ACM international conference on digital libraries*, 2000, pp. 85–94.
[18] Riedel S, Yao L and McCallum A. Modeling relations and their mentions without labeled text. In: *European conference on machine learning and knowledge discovery in databases*, 2010, pp. 148–163.
[19] Nebhi K. Ontology-based information extraction for french newspaper articles. In: *Annual german conference on artificial intelligence*, 2012, pp. 237–240.
[20] Nebhi K. Ontology-based information extraction from twitter. In: *Workshop on information extraction and entity analytics on social media*, 2012, pp. 17–22.
[21] Whetzel PL, Noy NF, Shah NH, Alexander PR, Nyulas C, Tudorache T and Musen MA. BioPortal: Enhanced functionality via new web services from the National Center for Biomedical Ontology to access and use ontologies in software applications. *Nucleic Acids Research* 2011; 39(suppl. 2): W541–W545.
[22] Muller H, Kenny E and Sternberg P. Textpresso: An ontology-based information retrieval and extraction system for biological literature. *PLoS Biology* 2004; 2(11): 1984–1998.

[23] Yildiz B and Miksch S. Motivating ontology-driven information extraction. In: *International conference on Semantic Web and digital libraries*, 2007, pp. 45–53.
[24] Krishnamurthy R, Li Y, Raghavan S, Reiss F, Vaithyanathan S and Zhu H. SystemT: A system for declarative information extraction. *SIGMOD Record* 2008; 37(4): 7–13.
[25] Cunningham H, Maynard D, Bontcheva K and Tablan V. GATE: A framework and graphical development environment for robust NLP tools and applications. In: *Anniversary meeting of the Association for Computational Linguistics*, 2002, pp. 168–175.
[26] Hearst MA. Automatic acquisition of hyponyms from large text corpora. In: *Conference on computational linguistics*, 1992. pp. 539–545.
[27] Etzioni O, Cafarella M, Downey D, Kok S, Popescu AM, Shaked T et al. Web-scale information extraction in KnowItAll (preliminary results). In: *International conference on World Wide Web*, 2004, pp. 100–110.
[28] Etzioni O, Cafarella M, Downey D, Popescu AM, Shaked T, Soderland S et al. Unsupervised named-entity extraction from the web: an experimental study. *Artificial Intelligence* 2005; 165(1): 91–134.
[29] Fader A, Soderland S and Etzioni O. Identifying relations for open information extraction. In: *ACL conference on empirical methods in natural language processing*, 2011, pp. 1535–1545.
[30] Mausam, Schmitz M, Soderland S, Bart R and Etzioni O. Open language learning for information extraction. In: *ACL conference on empirical methods in natural language processing and computational natural language learning*, 2012. pp. 523–534.
[31] Mintz M, Bills S, Snow R and Jurafsky D. Distant supervision for relation extraction without labeled data. In: *ACL conference on empirical methods in natural language processing*, 2009, pp. 1003–1011.
[32] Snow R, Jurafsky D and Ng AY. Learning syntactic patterns for automatic hypernym discovery. In: *Advances in neural information processing systems*, Vol. 17. Cambridge, MA: MIT Press, 2005, pp. 1297–1304.
[33] Yakushiji A, Miyao Y, Ohta T, Tateisi Y and Tsujii J. 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*, 2006, pp. 284–292.
[34] Swanson DR. Fish oil, Raynaud's syndrome, and undiscovered public knowledge. *Perspectives in Biology and Medicine* 1986; 30: 7–18.
[35] Lyons R. The spread of evidence-poor medicine via flawed social-network analysis. *Statistics, Politics, and Policy* 2011, 2(1): article 2.
[36] Ioannidis JPA. Why most published research findings are false. *PLoS Medicine* 2005, 2(8): e124.
[37] Etzioni O, Banko M, Soderland S and Weld DS. Open information extraction from the web. *Communications of the ACM* 2008, 51(12): 68–74.
[38] Karpicke JD and Roediger HL. The critical importance of retrieval for learning. *Science* 2008, 319(5865): 966–968.
[39] Whittington D and Hunt H. Approaches to the computerized assessment of free text responses. In: *Computer-assisted assessment conference*, 1999, pp. 207–219.
[40] Lin CY. Rouge: A package for automatic evaluation of summaries. In: *ACL workshop on text summarization branches out*, 2004, pp. 25–26.
[41] Foltz PW, Laham D and Landauer TK. Automated essay scoring: Applications to educational technology. In: *Conference on educational multimedia, hypermedia and telecommunications*, 1999, pp. 939–944.
[42] Franzke M and Streeter L. Building student summarization, writing and reading comprehension skills with guided practice and automated feedback. White paper from Pearson Knowledge Technologies, 2006.
[43] Brent E, Atkisson C and Green N. Time-shifted online collaboration: Creating teachable moments through automated grading. In: Juan AA (ed.) *Monitoring and assessment in online collaborative environments: Emergent computational technologies for e-learning support*. IGI Global, 2010, pp. 55–73.
[44] Mitchell T, Russell T, Broomhead P and Aldridge N. Towards robust computerized marking of free-text responses. In: *Computer-assisted assessment conference*, 2002, pp. 233–249 .
[45] Tatar D, Serban G, Mihis A and Mihalcea R. Textual entailment as a directional relation. *Journal of Research and Practice in Information Technology* 2009; 41(1): 53–64.
[46] Dagan I and Glickman O. Probabilistic textual entailment: Generic applied modeling of language variability. In: *PASCAL workshop on learning methods for text understanding and mining*, 2004.
[47] Carlson A, Betteridge J, Hruschka ER and Mitchell TM. Coupling semi-supervised learning of categories and relations. In: *NAACL HLT workshop on semi-supervised learning for natural language processing*, 2009, pp. 1–9.
[48] Dietterich TG. Ensemble methods in machine learning. In: *International workshop on multiple classifier systems*, 2000, pp. 1–15.
[49] Sill J, Takacs G, Mackey L and Lin D. Feature-weighted linear stacking, http://arxiv.org/abs/0911.0460 (2009, accessed 10 July 2014).
[50] Gutierrez F, Dou D, Fickas S and Griffiths G. Online Reasoning for Ontology-based Error Detection in Text. In: *OTM international conference on ontologies, databases and application of semantics*, 2014, pp. 562–579.
[51] Wang H, Horridge M, Rector A, Drummond N and Seidenberg J. Debugging OWL-DL ontologies: A heuristic approach. In: *International conference on the Semantic Web*, 2005, pp. 745–757.

[52] Miller G. Wordnet: A lexical database for English. *Communications of the ACM*, 1995, 38: 39–41.
[53] McCallum A and Nigam K. A comparison of event models for Naive Bayes text classification. In: *AAAI workshop on learning for text categorization*, 1998, pp. 41–48.
[54] Lafferty J, McCallum A and Pereira F. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In: *International conference on machine learning*, 2001, pp. 282–289.
[55] Maynard D, Peters W and Li Y. Metrics for Evaluation of Ontology-based Information Extraction. In: *WWW workshop on evaluation of ontologies for the Web*, 2006.
[56] Cimiano P, Ladwig G and Staab S. Gimme' the context: Context-driven automatic semantic annotation with C-PANKOW. In: *ACM international conference on World Wide Web*, 2005, pp. 332–341.