Improving open information extraction for semantic web tasks

Cheikh Hito Kacfah Emani, Catarina Ferreira da Silva, Bruno Fies, Parisa Ghodous

Improving Open Information Extraction for Semantic Web Tasks

Cheikh Kacfah Emani[1,2], Catarina Ferreira Da Silva[2], Bruno Fiès[1], and Parisa Ghodous[2]

[1] CSTB, 290 route des Lucioles, BP 209, 06904 Sophia Antipolis, France
{cheikh.kacfah, bruno.fies}@cstb.fr
[2] Université Lyon 1, LIRIS, CNRS, UMR5205, F-69622, France
{cheikh.kacfah-emani, catarina.ferreira, parisa.ghodous}@univ-lyon1.fr

Abstract. Open Information Extraction (OIE) aims to automatically identify all the possible assertions within a sentence. Results of this task are usually a set of triples (subject, predicate, object). In this paper, we first present what OIE is and how it can be improved when we work in a given domain of knowledge. Using a corpus made up of sentences in building engineering construction, we obtain an improvement of more than 18%. Next, we show how OIE can be used at a base of a high-level semantic web task. Here we have applied OIE on formalisation of natural language definitions. We test this formalisation task on a corpus of sentences defining concepts found in the pizza ontology. At this stage, 70.27% of our 37 sentences-corpus are fully rewritten in `OWL DL`.

1 Introduction

In recent years, researchers have tackled the problem of Open Information Extraction in different manner: from machine learning [11] to the exploitation of sentence structure [10], [3]. This last type of approaches obtains the best results. Unfortunately, their OIE-tools (exploiting grammatical dependencies [10] and syntactic tree [3]) sometimes output incorrect tuples. These wrong extractions are mainly due to parsing errors. Indeed, these approaches take advantage of the syntactic tree or grammatical dependencies provided by a parser. Consequently, a good way to improve Open Information Extraction is to handle parsing errors before the extraction stage itself. To achieve this goal, we have decided to *handle multi-word expressions* (MWE). A MWE is a phrase, made up of a set of words, which has a precise meaning and is unbreakable. "MWE-errors" represent more than 45% of parsing errors. We propose an algorithm to shorten multi-word expressions (Sect. 3). We evaluate the algorithm on sentences targeting the domain of building engineering construction, and show how it outperforms existing approaches (Sect. 5.1).

Now that we provide a tool which has the ability to split a complex sentence into a set of simple triples we can use it to accomplish more high level task. We illustrate this in an automatic Natural Language (NL) to *Web Ontology Language Description Logics* `OWL DL` [1] conversion process. Here NL sentences

are definitions of concepts found in an existing ontology. The goal is to help ontology designers to acquire automatically the OWL DL expression of a concept whose definition is expressed in natural language. For instance when defining a *Spiccy Pizza* through the sentence "*A spiccy pizza is any pizza that has a spicy topping*" the formalisation task aims to provide the following OWL DL expression:

$$\texttt{SpiccyPizza} \subseteq \texttt{Pizza}\ \textit{and}\ (\texttt{hasTopping}\ \textit{some}\ \texttt{SpiccyTopping})$$

This result is built from entities (Pizza, SpiccyPizza, SpiccyTopping and hasTopping) of the pizza ontology[1]. On the contrary of approaches like [16] [23,24], we do not create new entities from scratch. Our approach has the advantage to avoid the proliferation of entities and OWL DL expressions by taking advantage of the current state of the ontology. To obtain the formal expression of a sentence, we first extract all the assertions it contains using OIE. Next the challenge is to rewrite each triple as an OWL expression. Finally, all the expressions need to be recombined to obtain the final expression intended by the original definition. A preliminary evaluation of this approach has been made on a corpus of definitions of the pizza ontology. This corpus has 37 definitions from which 26 are fully and correctly formalised by our tool.

The key contributions of this work include:

- A betterment of Open Information Extraction (OIE) when taking into account domain terminology (Sect. 3)
- The proposal of a straightforward approach to provide a formal expression, in OWL DL, of a natural language definition (Sect. 4.1). This approach does not require any learning or external resource.
- The proposal of a formalisation approach which avoids an anarchic growing of new entities (Sect. 4.3). Indeed, the final expression, aligned with an existing domain ontology, uses *only* entities found within this ontology.
- A formalisation process which takes into consideration all the piece of information appearing in sentences (Sect. 4.2). This is done by taking advantage of OIE to ensure the grabbing of all the pieces of information.
- An original approach which merges all the triple extract from the sentence to obtain a single and coherent expression (Sect. 4.4).

The rest of the paper is organised as follows: first, a state of the art on both the identification the formal expression of concepts and Open Information Extraction (Sect. 2). Next, details about our approach to improve OIE when aware of domain terminology (Sect. 3). Then we expose our approach to obtain automatically OWL DL expressions from NL definitions, taking advantage of the enhanced OIE method we propose (Sect. 4). Next, we evaluate the two main problems we tackled (Sect. 5). Finally, we analyse the preliminary results we obtain (Sect. 6).

[1] http://www.cs.ox.ac.uk/isg/ontologies/UID/00793.owl

2 Related work

Some tasks in the Semantic Web community take as input a sentence and need to deal with *all the pieces of information* contained within it. It is the case for example of Question Answering (QA). To be able to answer a question, a QA system needs to be able to decode all the chunks of information held by the sentence. This inescapable need of identifying all what is said by the sentence can be achieved by OIE and it is what we apply to provide the formal expression of NL definitions. To the best of our knowledge, it is the first approach where definitions in natural language are automatically converted in their corresponding formal expressions containing entities completely disambiguated. Unlike approaches presented in [16] and [7], we do not learn the desired expression from the ontology or the knowledge base. In this work, we derive it from the concept's definition. Compared with some approaches working with NL sentences, we do not exploit partial information like in [25] where the authors focus on subsumption and thus exploit the "*is a*" fragment of the definition. Indeed, we are able to identify cardinality and value restrictions and we take in consideration all the evidences available within the sentence. In the attempt to identify more complex `OWL` restrictions, the interesting work of Tsatsaronis and colleagues [20] aims to provide the exact property, within a given ontology, which links two previously labelled concepts. Unlike our approach, a label denoting a class in the targeted ontology, first needs to be manually assigned to concepts and at the end of the process one does not have a complete formal expression reflecting the idea of the input sentence. In addition, our approach is part of a process which avoids the proliferation of new entities like in [23,24] where new ones are proposed to formalise a sentence. In our approach, we find the most suitable entities in the existing domain ontology able to transcript the idea conveyed by the NL definition. We are able to enrich lightweight ontologies with high logical specifications. It enables to obtain complex formal ontologies which are thus suitable for powerful reasoning (subsumption detection and consistency checking).

As mentioned in the above paragraph, OIE can help us to identify all the pieces of information within definitions, hence we need to use the most accurate OIE-tool. During the recent years, many systems were developed to perform OIE. It is the case of `ReVerb` [11], `OLLIE` [17], `ClausIE` [10] and CSD-IE [3,4]. `ReVerb` by means of efficient heuristics, focused on *incoherent* and *uninformative* triples. Unfortunately, relations extracted by `ReVerb` were necessarily verb-based. This is the main reason why `OLLIE`, developed by the same group of researchers, was provided. In addition to be able to identify non verb-driven facts, `OLLIE` aims to provide the *context/condition*, if existing, in which the extracted fact can *be considered true*. These two previous tools are machine learning-based. The most recent approaches do not need any additional resource. They only exploit result of a standard parser. `ClausIE` uses grammatical typed dependencies and `CSD-IE` the syntactic tree of the input sentence. These two tools dissect each piece of the result they get from the parsing tool. Consequently, if a dependency is wrong or a sub-tree is incorrectly labelled in the syntactic tree these OIE-tools may provide inaccurate extraction. This is why we propose to make some

preprocessing operations before OIE itself. The details of these tasks are given in the next section.

3 Handle Multi-Word Expressions to Improve Open Information Extraction

Recent approaches of OIE, for instance `ClausIE` [10] or `CSD-IE` [3,4], *only* use syntactic pieces of information to identify informative triples from a sentence. Consequently, errors in analysis of sentences by Natural Language Processing (NLP) tools lead to major incorrect extractions in Open Information Extraction. In a sample set of sentences selected from various regulatory texts in the field of building engineering (see Sect. 5.1 for more details about this corpus), the percentage of errors due to MWE is 46.15% using `CSD-IE`. To handle problems caused by MWE is thus a relevant way to improve result of IE. Our solution to improve the quality of results of OIE-tools is to handle MWE by means of this three-step approach: (*i*) the *detection* of MWEs within the sentences, (*ii*) their *shortening* and then the information extraction process is done within each resulting triple containing a shortened-MWE, (*iii*) this shortened-MWE is expanded to get back to its original form. This process is illustrated by Fig. 1 and detailed in the following subsections.

Step 1 - Detection of Multi-Word Expressions

For us, a MWE is every phrase which the meaning will be modified (even become meaningless) by the addition or the deletion of any of its word. Consequently, a *domain term*, an *idiomatic expression*, a *phrasal verb*, a *named entity*, a *formula*, a *quotation* etc. is a MWE. These types of MWE make us foresee that MWE are more easily and *reliably* identifiable in a given domain. One can have also domain-independent terms that are not related to the field of study but are frequently found in the corpus. It is the case of operators (example: less than, less than or equal to, as much as), idiomatic expressions (example: "Loose your head", "Jump in feet first"), units of measurements, etc. So, a set of MWE in a precise domain can be made up of the *terminology of the field* and *frequent terms*. This last category of terms can be obtained by means of existing statistical methods and the help of human experts. At this stage we identify the MWE present in the original sentence. We thus have a *list of possible MWE* in our corpus (see Sect. 5.1 for more details).

Step 2 - Compression of a Multi-Word Expression

The reason why precision of OIE-tools is affected by MWE is that the latter is considered by the former to be *non atomic*. Hence, to limit potential hazardous fragmentation of expressions, we propose to extract information from a new version of sentences where each MWE will have been replaced by a *shortened version*. So, now the question is: how do we get this short version of MWE?

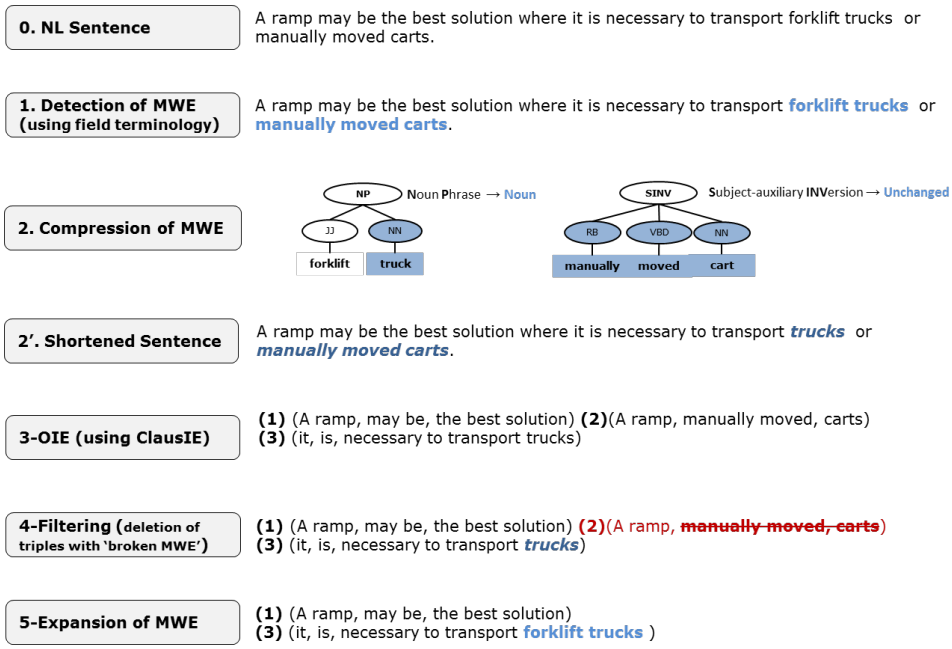

Fig. 1. An end-to-end example of Open Information Extraction when handling multi-word expressions.

When trying to answer this question, we must have in mind that the shortened sentences must always be semantically and syntactically correct to be appropriately handled by OIE-tools. We propose the following steps for shortening a MWE (using its syntactic parse tree):

1. if the MWE is a *clause* (list of labels for clauses is available in [6]) or a *verb phrase*, *there is no shortening*;
2. else, if the MWE is a noun phrase, the first token *labelled noun* is considered to be the shortened version of the MWE;
3. else, we take the string provided by the smallest phrase[2] within the tree.

Let us note that some MWE will be short enough so that they will remain the same after the shortening. Although such MWE (like any other MWE) is considered to be atomic, this is important to have in mind because, if an OIE-tool breaks a MWE, the resulting triple will be incorrect.

After this stage, we now perform OIE itself, which is the **third step** (Fig. 1). This OIE is done by using existing OIE-systems. Consequently, the following steps come after OIE and take as input results of OIE, i.e triples.

[2] An exhaustive list of labels for phrases is available in the Penn Treebank [6].

Step 4 - Filtering of Open Information Extraction Results

Earlier in this work, we have pointed out a set of things which degrades precision of OIE-tools. We have focused on the problematic role caused by multi-word expressions. Now, we use the only characteristic of MWE to finalise our OIE-process. Indeed, a MWE is *unbreakable*. Consequently, when a triple contains only a fragment of a MWE, it is considered as incorrect. This filtering is done before the expansion stage, so the MWE are in their "shortened" form.

Step 5 - Expansion of a Multi-Word Expression

After the OIE has been done from the shortened version of the sentence, and te set of facts has been filtered, we now have to reconcile the remaining facts with the original (long) sentence. We then look into the list of the extracted facts to replace shorten version of MWE by their initial long form. This is the aim of this step.

4 Automatic Acquisition of Axioms from Natural Language Definitions via Open Information Extraction

In this section, we present our approach to formalise NL definitions (Sect. 4.1). The first step of this approach uses straightforwardly the improvements of our OIE method (Sect. 4.2). From OIE, we obtain a set of triples which are then formalised (Sect. 4.3). Then, we recombine all the formalised triple to obtain a single formal expression (Sect. 4.4). Finally, we provide a full example to illustrate our approach.

4.1 Overall Approach

Taking as input a sentence $\mathcal{S}$, a domain ontology $\mathcal{O}$, our approach aims to provide automatically the expression of the defined concept within $\mathcal{S}$ w.r.t to $\mathcal{O}$. We assume that the sentence follows the Aristotelian definition pattern [5]. In simple terms our definitions are made up of defined concept that is put in relation to one or more general concepts then, optionally, various precisions are added to these general concepts. Our formalisation approach is summarised as follows:

1. Structural Sentence Decomposition of $\mathcal{S}$ via OIE.
   The result of this step is a set of triples $\{\tau_i\}$ and their organisation (sentence structure) Σ
2. For each triple τ_i
   (a) Identify and decode non domain terms
   (b) Identify possible concept C_s and C_o in $\mathcal{O}$ using respectively the subject-part and the object-part of the triple.
   This identification is done by a string matching algorithm and we keep only concepts with the highest similarity score.

   (c) Identify the list $\{p_l\}$ of properties of $\mathcal{O}$ whose domain and range are *simultaneously compatible* with C_s and C_o
   (d) Rank $\{p_l\}$ according to the matching score between each of its elements and the whole triple and select the top property
   (e) Rewrite formally ($C \subseteq DL(\tau_i)$)
3. Link the triples as suggested in Σ

The rest of this section details the steps of our approach.

4.2 Structural Sentence Decomposition

In our approach, we take advantage of Structural Sentence Decomposition using Open Information Extraction (OIE). This notion is exposed in details in [15]. Basically, when OIE is done, facts are *simply* extracted without any piece of information on how they are organised and linked. For instance, from the sentence "*If a building is intended to host the public then it should have two escapes or three main entrances*" we do not only have these three facts, triples $Fact_1$ <*A building, is intended, to host the public*>, $Fact_2$ <*it, should have, two escapes*> and $Fact_3$ <*it, should have, three main entrances*>. Indeed, it is crucial to output how all these assertions are related as shown here: `If`($Fact_1$) `then` ($Fact_2$ `or` $Fact_3$). Since this organisation reflects the *structure* of the sentence, we call it *Sentence Structure* and we name it "*Decomposition*".
Here we use OIE with improvements bring by the handling of MWE as described earlier in this paper. A wrong information may lead to a wrong formalisation result. This is why we choose to deal with our improved OIE approach, despite some preprocessing steps before extraction itself. For the OIE step itself, we have chosen `CSD-IE` of Bast and Haussman [3]. First, let us mention that `ClausIE` [10] and `CSD-IE` are, to our knowledge, the best current OIE-tools. Secondly, characteristics of `CSD-IE` in comparison with `ClausIE` make the former more suitable for us than the latter. Indeed, `CSD-IE` was designed to provide triples with some quality aspects, mainly:

- *minimality*: triples should be small enough so that a more fine extraction cannot be made from them. In addition, in `CSD-IE`, *coverage* is a main concern. It means that there is an effort to make appear in the whole set of resulting triples each word of the original sentence at least once.
- *accuracy*: of course all systems, and thus `ClausIE`, are expected to be accurate. But, in `CSD-IE`, there is a given heuristic which has a worth for us: the predicate-part of their triples must only contain words which "belong to the verb". This is a guarantee of having a sort of *format* for resulting the triples. It will allow us to find string which may lead to a possible concept (Step 2.(b) of our approach in 4.1) *only* in the subject and object parts of the triple. We thus assume that verbs' (i.e predicates of our minimal triple) contribution to the identification of a concept is negligible.

4.3 Processing of each Triple

In accordance with the content of the previous section, our triples here are supposed to be minimal. Consequently, each of these triple will lead to a "*minimal restriction*" or a class subsumption. In `OWL DL` we have two main kinds of restrictions: *value* and *cardinality* constraints[3]. These restrictions in their *minimal* form *strictly* follow the templates presented below:

- Value constraints
  - `owl:allValuesFrom`: $< property >$ only $< Class >$
  - `owl:someValuesFrom`: $< property >$ some $< Class >$
  - `owl:hasValue`: $< property >$ value $< Class >$
- Cardinality constraints
  - `owl:maxCardinality`: $< property >$ max $< integer > < Class >$
  - `owl:minCardinality`: $< property >$ min $< integer > < Class >$
  - `owl:cardinality`: $< property >$ exactly $< integer > < Class >$

In these templates, words between $' <'$ and $' >'$ represent *slots* to be filled. The next paragraphs present elements related to the identification of the correct restriction and then its filling.

Handling of non Domain Terms. When we look at our set of templates, we have some non domain terms i.e terms not related to our domain ontology $\mathcal{O}$. These terms are `OWL` *key terms*. This operation consists of identifying the right non domain elements when taking as input a triple. As shown in existing works, mainly in *Question Answering* systems [21,22], they can be handled through a lexicon of non-domain terms. In this lexical resource, we have a set of NL expressions (e.g: at least, higher than, uniquely, only etc.) and their formal equivalents, i.e. the exact `OWL` key term to which they refer and thus *the right template*. In this work, we have essentially taken advantage of existing lexicon of TBSL [21].

Concepts Identification. In our approach, we uppermost identify concepts instead of property. The reason is that, in one hand, we have some cases where there is not any hint to directly identify a property from the verb of the phrase. It mainly happens when the verb is a variation of the auxiliaries *be* or *have* (e.g: <A wonderful pizza, has, a topping of tomato>). On the other hand, a given predicate can be expressed using different NL expressions (e.g: A pizza *has/is made up of/contains* tomato). Here, we find a concept C_s using the subject-part and another one C_o using the object-part of the triple. C_o and C_s are found within $\mathcal{O}$.
Given an input string s and a domain ontology, we compare s with each *label* of all classes and individuals of the ontology[4]. For each comparison, we have

[3] `http://www.w3.org/TR/owl-ref/#Restriction`
[4] With a large ontology, such comparison must take advantage of an index for the sake of scalability.

a matching score and the class with the highest score is considered to be the concept (or individual) denoted by s. The challenge here is to have an *appropriate* metric for string alignment. Since in this task we are dealing with multi-token strings, we expect the following characteristics for the matching metric:

- the position of each token in corresponding string
- the similarity between each token of the two strings
- the editing distance (deletion, replacement, insertion) between the two strings

In the literature, we have found a string similarity metric which fulfil all these features and which is called `Liuppa` [18]. It has been developed for ontology alignment at the terminological level. To handle the "token-level" of string comparison, designers of `Liuppa` first propose to rewrite each input string as a "*string of symbols*" as illustrate by Tab. 1.

Table 1. From token to symbols as performed by `Liuppa` [18]. This operation takes as input two strings (here S_1 = *toppings of tomato* and S_2 = *tomato topping*) and rewrite them as a set of symbols ($S_1 = \alpha_1\alpha_2\alpha_3$ and $S_2 = \alpha_3\alpha_1$). Each α_i represents two tokens similar over a given threshold and is seen as a character in the alphabet $\{\alpha_1,\ \alpha_2,\ \alpha_3\}$

		Token t	**Most similar token** t'	**score**(t,t')	**Symbol**
S_1	t_1	toppings			α_1
	t_2	of			α_2
	t_3	tomato			α_3
S_2	t_4	tomato	t_3 = tomato	1.0	α_3
	t_5	topping	t_1 = toppings	0.975	α_1

We see that this replacement uses an existing string matching metric. After a set of experiments on our corpus, the `Jaro-Winkler` metrics [26] with a threshold equals to 0.85 gives the best results. The second and last step of matching performed by `Liuppa` is to provide the similarity between the two "*string of symbols*" (for the example exposed in Tab. 1, in this step one computes the similarity between $\alpha_1\alpha_2\alpha_3$ and $\alpha_3\alpha_1$). Experimentally, we have chosen the `Jaccard` [12] metrics for this second matching. Indeed it is suitable for set-based similarity. Because there is a lost of the order of the symbols when using `Jaccard`, we relax `Liuppa(`S_1`,`S_2`)` with `Jaro-Winkler (`S_1`,`S_2`)`. The final similarity measure is given by:

$$Sim(S_1, S_2) = 0.75 \times \texttt{Liuppa(}S_1\texttt{,}S_2\texttt{)} + 0.25 \times \texttt{Jaro-Winkler (}S_1\texttt{,}S_2\texttt{)} \quad (1)$$

Properties Identification Now we have possible concepts C_o and C_s, we have to provide the property which links them. Since we are linking out input triple with the domain ontology $\mathcal{O}$, this property should be found in the set of properties of $\mathcal{O}$. For this reason, we should first identify the set of properties

$\{p_l\}$ in the schema of $\mathcal{O}$ which are *compatible* both with C_o and C_s. A property p_l is compatible with C_o and C_s if and only if:

$$Domain(p_l) \cap Hierarchy(C_s) \neq \emptyset \text{ and } Range(p_l) \cap Hierarchy(C_o) \neq \emptyset \quad (2)$$

In equation 2, $Domain(p_l)$ and $Range(p_l)$ are respectively the range and the domain of p_l and $Hierarchy(x) = \{c, x \texttt{ rdfs:subClassOf}^* \ c\}$. The star (*) here denotes the *property path operator* for an arbitrary path of a length equals to 0 or more [5]. It allows us to select all the super-classes of x including x itself. $\{p_l\}$ can be obtained using the following `SPARQL` query:

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX :<http://example.com/onto#>
SELECT DISTINCT ?p
WHERE  {:Cs rdfs:subClassOf* ?Dp . ?p rdfs:domain ?Dp.
        :Co rdfs:subClassOf* ?Rp . ?p rdfs:range ?Rp }
```

In this query `:Co` and `:Cs` are respectively URIs for the concepts C_o and C_s previously identified. When `:Co` or `:Cs` denotes an individual, the predicate `rdf:type` replaces `rdfs:subClassOf*` in the above query.

Selecting the Top property From the set $\{p_l\}$ of possible properties, we now need to rank them. This ranking will be done based on the strings provided by our input triple. Let us remind that using the subject-part and the object-part of the triple, we have identified the corresponding concepts C_s and C_o in the domain ontology $\mathcal{O}$. As first mentioned in Sect. 4.1 sketching our approach, we perform a matching between labels of each member of $\{p_l\}$ and the whole triple. The matching score is obtained using the following formula:

$$score(p_l, \tau_i) = \alpha.sim(p_l, pred(\tau_i)) + \beta.(score(p_l, subj(\tau_i)) + score(p_l, obj(\tau_i))) \quad (3)$$

In equation 3, α and β are two scalars used to weight the similarity between the current property and the predicate-part of the triple on one hand and the same property and the rest of the triple on the other hand. The relation between them is $\alpha + \mathbf{2}.\beta = \mathbf{1}$. We thus give more importance to the predicate than to the rest of the triple for property ranking. $subj(\tau_i)$ and $obj(\tau_i)$ represent the subject-part and the object-part of the triple τ_i. In this equation, sim represents the string similarity measure expressed in equation 1. In practice we have chosen $\alpha = 0.5$ and $\beta = 0.25$.

In brief, we assume that the predicate of a triple is pivotal for property identification. But in some cases, predicates are not informative enough. It is thus important to use the other part of the triple (subject and object).

Formal Rewriting of Triple In the step 2.a of our approach, we have already identified the template of the restriction (and thus `OWL` key terms) and potential values. Now, using the concepts (C_o and C_s) and the property p_r, we just have to fill the selected template. We call $\mathcal{R}(\tau_i)$ (Restriction from τ_i) the output of this step.

[5] `http://www.w3.org/TR/sparql11-query/#propertypath-arbitrary-length`

4.4 Triples Join

Now that we have built our set of minimal restrictions, it is time to put them together to obtain the expression of our defined concept as stated by the input definition. Let us remind that our triples (still in NL) $(\tau_i)_{i=1}^{n}$ were obtained using a "Structural" Information Extraction approach, meaning that we know exactly how all these n triples are linked. As illustrated in Sect. 4.2, these links can be expressed by subordinate conjunctions (*if, then, while*, etc.), conjunctive adverbs (*unless, otherwise,* etc.) or by coordinating conjunctions (*and, or*). In this paper we will focus on cases where triples are linked by coordinating conjunctions (we discuss about other types of links in Sect. 5.2). We therefore notice that our triples are linked by *logical operators*. For us, negation is handled here at the *level of triple*. For instance, from the definition "*A vegetarian pizza is any pizza that does not have fish topping.*" we extract τ_1:-(A vegetarian pizza, is, any pizza) and τ_2:-(any pizza, does not have, fish topping). The structure of the sentence is $\Sigma = \tau_1$ `and` τ_2. We see that expressing τ_2 by (`not` (any pizza, has, fish topping)) does not have any impact on Σ. Consequently, the negation operator is not a possible link in Σ.

In this final step, we aim to answer the question raised by the following example.

We have the current input data:

- τ_1:-(A vegetarian pizza, is, any pizza)
  and thus $\mathcal{R}(\tau_1) =$ `:VegetarianPizza` $\subseteq$ `:Pizza`
- τ_2:-(any pizza, does not have, fish topping)
  and $\mathcal{R}(\tau_2) =$ (not (`:Pizza :hasTopping` some`:FishTopping`))
- $\Sigma = \tau_1$ and τ_2
  and thus $\mathcal{R}(\Sigma) = \mathcal{R}(\tau_1)$ and $\mathcal{R}(\tau_2)$

How can we obtain:

$\mathcal{R}(\mathcal{S}) =$ `:VegetarianPizza` $\subseteq$ (`:Pizza` and (not(`:hasTopping` some `:FishTopping`))

When we look at the expected result $\mathcal{R}(\mathcal{S})$, the task to perform seems to be a *join*, like in relational databases. Thus the name of *triples join*. In the above example, the *common attribute* for the join is the concept `:Pizza`. To obtain $\mathcal{R}(\mathcal{S})$, we will take advantage of join and boolean expressions simplification methods. In $\mathcal{R}(\Sigma)$ we already have some boolean operators (links between triples). Now, in the light of our approach to obtain them, for the sake of the current task, each $\mathcal{R}(\tau_i)$ is simply rewritten $C_s^i\ r_i\ C_o^i$. In this expression r_i stands for the restriction parameters[6].

For two triples τ_i and τ_j we perform simplification using these formulae:

$$\left(C_s^i\, r_i\, C_o^i\right) * \left(C_s^i\, r_j\, C_o^j\right) = C_s^i \subseteq \left(r_i\, C_o^i * r_j\, C_o^j\right) \tag{4}$$

[6] r_i is the subsumption or the set of elements of a more complex restriction (URI of the restriction property, `OWL` keywords for the type of the restriction, etc.) as explained in the introduction of Sect. 4.3

$$\left(C_s^i\, r_i\, C_o^i\right) * \left(C_o^i\, r_j\, C_o^j\right) = C_s^i \subseteq \left(r_i\, (C_o^i * r_j\, C_o^j)\right) \tag{5}$$

In these formulae, the operator $*$ represents a logical operator (*and* or *or*). Moreover, as it is the case for join in relation database, a "*common*" operand to τ_i and τ_j is needed (in the following numbered list, we suppose $* = \texttt{and}$[7]):

1. within 4, we have $C_s^i = C_s^j$ (the two restrictions concerned the same subject). The first member of this equation means "instances of C_s^i are individuals of the anonymous class $r_i\, C_o^i$ `and` instances of C_s^i are individuals of the anonymous class $r_j\, C_o^j$". Consequently, instances of C_s^i are member (subsumption) of both $r_i\, C_o^i$ `and` $r_j\, C_o^j$. In this case the join is similar to a *factorisation* of an arithmetic expression and we will refer to this first case by that name.
2. $C_s^j\, r_j\, C_o^j$ means "instances of C_s^j are individuals of the anonymous class $r_j\, C_o^j$". Now, suppose $C_o^i = C_s^j$ (equation 5). It implies that the restriction $r_j\, C_o^j$ has to be applied to C_o^i. This additional restriction for instances of C_o^i thus needs to be inserted in $\tau_i = (C_s^i\, r_i\, C_o^i)$. We call this equation where C_o^i is refined *refinement*.

To evaluate the simplification of a complex expression, we use the algorithm 1 below. In this algorithm we suppose the existence of the directed graph $G(V, E)$ thus defined:

- V is made up of all the set of distinct concepts found in $(\tau_i)_{i=1}^n$
- a directed edge (e_p, e_q) belongs to E if there exists a triple $\tau_i = (C_s^i\, r_i\, C_o^i)$ in $(\tau_i)_{i=1}^n$ with $C_s^i = e_p$ and $C_o^i = e_q$.

G is a dependency graph where a concept depends of another one if the former (in object position) helps to provide more information about the latter (in subject position). In G some vertices do not have any incoming edge and are element of what we call *Root*. Moreover, we can know at any time what is the triple which has lead to the edge $\left(C_s^i, C_o^i\right)$ in G (this information is kept during the built of G). We will use the function `triple(`C_s`,`C_o`,`G`)` to denote this triple.

The *recursive* function `FactorizationRec` depicted below "factorises" an input concept C. Before performing factorization itself (`Factorization`) the object-concepts in triples having C in subject position are firstly factorised and then refined (`Refinement`) in the aforementioned triples.
We call `FactorizationRec` for each element of *Root*. In practise this set is usually a *singleton* containing the defined concepts. It is due to the Aristotelian form of the definition considered here.

[7] Only for better understanding. The choice of `or` would not have changed anything.

input : C, Σ, G
output: A triple $C\, r\, c_o$ which is the *factorisation* of all the triples having the concept C in subject position `// Equation 4`

$C_{objs} \leftarrow G.children(C)$;
if $|C_{objs}| == 0$ **then** return ε;
`// It means there is not any triple with` C `as subject, thus no factorization possible`
else
  BoolLinks $\leftarrow$ `BooleanLinks` (Σ, C_{objs}) `// ''Children'' of` C `(i.e` C_{objs}`) and their links (boolean operators) in` Σ `(A directed edge in` G `corresponds to a triple in` Σ`)`
  for *each* C_o *in* C_{objs} **do**
    $\tau \leftarrow$ `Triple` (C, C_o, G);
    $\tau' \leftarrow$ `FactorizationRec` (C_o, Σ, G);
    BoolLinks $\leftarrow$ `Refinement` (τ, τ', BoolLinks);`// Equation 5`
  end
  return `Factorization` (C, BoolLinks);
end

Algorithm 1: Recursive factorization (`FactorizationRec`) algorithm

4.5 An End-to-End Example

In this section we provide a full example of the formalisation of a NL definition.

Inputs: the defined concept "*American Pizza*" and its definition "*An american pizza is a pizza which has toppings of pepperoni, mozzarella and tomato.*"

1. **Decomposition**
   - τ_1 (An american pizza, is, a pizza)
   - τ_2 (a pizza, has, toppings of pepperoni)
   - τ_3 (a pizza, has, toppings of mozzarella)
   - τ_4 (a pizza, has, toppings of tomato)
   - $\Sigma = (\tau_1)$ and τ_2 and τ_3 and τ_4
2. (a) **Identification and decoding of non domain terms**
   - τ_1 An american pizza $\underbrace{\textbf{is a}}_{\subseteq}$ pizza
   - No identification in the triples $\tau_2 - \tau_4$

   (b) **Concepts identification**
   - $(C_s, C_o)_{\tau_1}$ = (`pizza:AmericanPizza`, `pizza:Pizza`)
   - $(C_s, C_o)_{\tau_2}$ = (`pizza:Pizza`, `pizza:PepperonniSausageTopping`)
   - $(C_s, C_o)_{\tau_3}$ = (`pizza:Pizza`, `pizza:MozzarellaTopping`)
   - $(C_s, C_o)_{\tau_4}$ = (`pizza:Pizza`, `pizza:TomatoTopping`)

   (c) **Properties identification**
   - No identification in τ_1 since the predicate part of the triple has lead to a subsumption relation
   - In $\tau_2 - \tau_4$ we have the same set of properties { `pizza:hasIngredient`, `pizza:isIngredientOf`, `pizza:hasTopping`}

(d) **Ranking**
With the presence of words *has* and *toppings* the ordered list of properties (for triples $\tau_2 - \tau_4$) is (`pizza:hasTopping`, `pizza:hasIngredient`, `pizza:isIngredientOf`).
Therefore the top property in this case is `pizza:hasTopping`.

(e) **Formal rewriting** (when there is not any hint on the type of the restriction, we choose the *some values* restriction)
- $\tau_1 \rightarrow$ `AmericanPizza` $\subseteq$ `Pizza`
- $\tau_2 \rightarrow$ `Pizza hasTopping some PepperonniSausageTopping`
- $\tau_3 \rightarrow$ `Pizza hasTopping some MozzarellaTopping`
- $\tau_4 \rightarrow$ `Pizza hasTopping some TomatoTopping`

3. **Final linking w.r.t** Σ
The dependency graph here is depicted by Fig. 2. The only root element (node without incoming edge) of this graph is the concept `AmericanPizza`. To factorize it, we will first need to do so with `Pizza` using $\tau_2 - \tau_4$. We obtain for the factorization of `Pizza`:

`Pizza` $\subseteq$ ((`hasTopping` some `PepperonniSausageTopping`) and (`hasTopping` some `MozzarellaTopping`) and (`hasTopping` some `TomatoTopping`))

Now, by taking the above triple ($C_s^j =$ `Pizza`, $r_j = '\subseteq'$, and $C_o^j =$ ((`hasTopping` some `PepperonniSausageTopping`) ...)) which adds more precision to the concept `Pizza`, we refine the same concept present in τ_1 and we obtain:

`AmericanPizza` $\subseteq$ (`Pizza` and ((`hasTopping` some `PepperonniSausageTopping`) and (`hasTopping` some `MozzarellaTopping`) and (`hasTopping` some `TomatoTopping`))

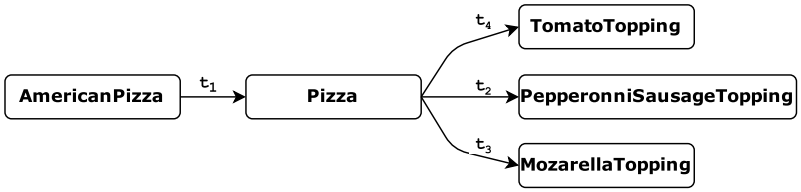

Fig. 2. Dependency graph from the triples provide in the example in section 4.5

5 Overall Evaluation

The work we present in this paper tackle two main aspects. The first aspect is about the improvement of OIE using domain knowledge and the second one is an approach to provide the `OWL DL` expression of definitions in natural language. These two major tasks are evaluated into two following subsections. A proof of concept of the approach is available at `http://tinyurl.com/ozkbmf3`.

5.1 Multi-Word Expressions Handling for Open Information Extraction

After making some statistics on the factors which lead to incorrect Information Extraction, we have decided to tackle the multi-word expressions-problem. The first step of the approach we propose is to identify them in the input sentence. Be able to perform such identification implies to have a list of possible MWE. That is why we hypothesize that the sentence describes the realities of a specific field of interest. Actually, to know that we are working in a specific domain implies to have a good idea of the terminology of this domain. For our evaluation, we have taken the list of terms (labels of the concepts in the field) as the set of our MWE. These terms have been obtained through a *key terms extraction process*. We have taken advantage of existing tools (`Alchemy` [8] in our case) to carry out this extraction. For this preliminary evaluation, our corpus is made up of 50 random sentences from documents about *fire safety* [8], *energy efficiency* [9] and *accessibility* [2]. Our list of MWE consists of result provided by `Alchemy` without terms containing a proper noun. Moreover, we have added units of measurement.

To perform OIE itself after preprocessing tasks, we have used `ClausIE` of Del Corro and Gemulla [10]. In addition, we compare our results to `CSD-IE` of Bast and Haussmann [3] and to the "original" version of `ClausIE`. Results are presented by Tab. 2.

Table 2. Results of the primary evaluation of OIE using `ClausIE` (by handling MWE -thus called `ClausIE-MWE`) and `CSD-IE` (without any preprocessing) on a corpus built from law in building engineering construction.

Tools	**#extractions**	**#extractions-correct**	**#extractions-incorrect**	**#errors-due-to-MWE**
`CSD-IE`	218	127 (58,26%)	91 (41.74%)	42 (46.15%)
`ClausIE`	315	201 (63.80%)	114 (36.20%)	60 (52.63%)
`ClausIE-MWE`	165	135 (81.81%)	30 (18.89%)	9 (30%)

`CSD-IE` performance in *this domain-specific* corpus (58.26%) is less good than in "open datasets" like the Wikipedia (70.0%) and New York Times dataset (71.5%) [3]. The same remark can be made to `ClausIE`. But by handling the MWE-problem, we obtain 81.81% of correct extractions (18.19% of errors). We still have a certain number of errors. Some of these errors are caused by our handling of MWE as discussed in Sect. 6.1 and others errors come from OIE-tools we use at the extraction step itself.

[8] `http://www.alchemyapi.com/`

5.2 Automatic Formalisation Process of Definitions

Test Material. To the best of our knowledge, there is not any standard benchmark for automatic formalisation of NL definitions. Consequently we have made preliminary test on a well-known ontology from the SW community: the pizza ontology. The pizza ontology is very suitable for our task since it matches well our hypothesis: we have and existing schema with a given hierarchy, we also have a set of properties with their domain and range well defined, we have a set of new concepts and their definition to add to this pre-existant schema. We have taken the version 1.5 of the pizza ontology and gather their definitions (as stated in the official documentation [13]) when not provided in the `rdfs:comment` annotation of concepts. Since we have estimated that the 13 definitions we had after this operation were not enough for a real evaluation, we have gather more definitions on the Web. Indeed, there are many `:NamedPizza` is this ontology with no NL definition or comment. We look for possible definitions in Wikipedia and various restaurants sites[9]. In addition, these definitions are challenging because they do not follow the "classical" pattern denoted by **[Concept]** <**is a/are a**> **[Definition]** like in [23] or [19]. Indeed, the defined term is not always the introducer of the sentence (e.g: *"Any pizza that has a spicy topping* ***is a spicy pizza****"*) or can be done using another verb (e.g: *"A high calorie pizza* ***will be defined to be*** *any pizza ... "*). Moreover, since the pizza ontology example has been designed to illustrate the whole set of possible OWL restrictions, we have of course all these possibles restrictions expressed in NL definitions. At this stage we have a set of 37 definitions.

Results. The first step of our approach brings us to perform an Open Information Extraction (OIE). Of course, this operation does not always provide accurate results. An evaluation of OIE when taking into account domain knowledge (and thus multi-word expressions - MWE) is described in [14]. There, the accuracy is 81.81% when using `ClausIE` for the extraction task itself. In this case, the domain knowledge (i.e the terminology) taken as input was the set of labels of our entities. By taking into account this list of domain terms, we improve OIE from 75.2% (when using directly `CSD-IE`) to 85.12% as shown by Table 3.

For the extraction of restrictions itself, validation was done manually. Over the 37 expressions provided by the formalisation of our sentences, 26 (70.27%) were correct (i.e both triples in `OWL DL` and the final expression of the defined concept). From the 121 triples extracted (still in NL), 103 (85.12%) were accurate. When formalising these triples, we get a precision of 79.61% (82 triples). We remark that our matching process handle correctly *noise*[10] in NL expressions. These numbers are summarised in Table 4.

[9] For example `http://www.pizzaexpress.com/our-food/our-restaurant-menu/mains/`, `http://www.nutritionrank.com/`, etc.

[10] Concepts' tokens are usually surrounded by adjectives, adverbs, prepositions, etc.

Table 3. Results of the evaluation of OIE when using `CSD-IE` (by handling MWE -thus called `CSD-IE-MWE`) and `CSD-IE` (without any preprocessing) on our set of definitions.

Tools	**#Triples**	**#Triples-correct**	**#Triples-incorrect**
`CSD-IE`	125	94 (75.2%)	31 (24.8%)
`CSD-IE-MWE`	121	103 (85.12%)	18 (14.88%)

Table 4. Results of the formalisation process

#Definitions	**#Correct-Definitions-OWL**	**#Triples-correct**	**#Triples.-correct-OWL**
37	26 (70.27%)	103	82 (79.61%)

6 Overall Discussion

Preliminary evaluations we have performed both on improving OIE and automatic formalisation of definitions highlight some interesting points. They are discussed in the next two subsections.

6.1 About Open Information Extraction Improvement

We have seen that handle MWE, in a given domain, helps to improve OIE on sentences of that domain. However we see that our method to handle MWE has to be improved. Indeed 30% of remaining errors after the shortening of MWE are due to this shortening operation. Indeed:

- When we choose the first noun of a *noun phrase*-MWE to replace this MWE it is not always its suitable representative. In practice, some nouns can sometimes be *tagged as verb* and thus a *potential predicate* (e.g: "fire" in the expression *fire extinguisher*, "means" in the term *means of access*, etc.) and it can cause wrong extractions. Consequently, when we have more than a noun in a noun phrase, we must have more criteria to choose the representative.
- In some sentences, parsers correctly identify the prepositional modifiers of all verbs, nouns, adverbs, etc. Consequently the *presence of MWE is a priori not a problem* for OIE-systems. Unfortunately, the deletion of prepositions (found for example in a *noun phrase*-MWE) during the shortening may lead to parsing errors. Indeed, parsers will try to identify new relations which may be wrong leading to incorrect extractions as illustrated below:
  1. `Original sentence :` "A stair is a fixed means of access."
  2. `Shortened version :` "A stair is a fixed means."
  3. `OIE : CSD-IE`→(A stair, means, is) & `ClausIE`→(A stair, a fixed means)

One of the possible solutions to avoid the shortening of MWE to escape from their multi-word problem is to replace a MWE by a *synonym*. Ideally, this synonym should have fewer words (eventually a single word) than the original MWE. Such synonyms could be found in Linked Open Data or in lexical databases like Wordnet, etc.

6.2 Analysis of our Automatic Formalisation Approach

In this section, we analyse the key steps of our approach in the light of the evaluation we have performed. In addition we highlight possible perspectives to this work.

Improving OIE. By using the compression of multi-word expressions we make a step in the direction of a more precise OIE-tool. But another weakness of OIE tools is enumeration detection. In a definition it is very common to enumerate items. The challenge for OIE-tools is to be able to decode correctly these items and to put them in the right place in a minimal <subject, predicate, object> representation. This task goes directly in hand with the identification of the structure of the sentence.

Triple Formalisation. We have made some (implicit) hypothesis for this step and they are discussed here.

1. In our approach we assume that each triple correspond to a restriction. In practice, it is not always the case. Some triples are not informative. For instance, for the sentence *"Sicilian pizza is a pizza prepared in a manner that originated in Italy."* we have the following output <A pizza, is prepared, in a manner> which does not have a correct formal expression w.r.t the pizza ontology. Such cases must be handled.
2. We do not take into account the fact that the concepts C_s and C_o could play opposite roles w.r.t the ontology. For instance, let us take the triple <Tomato, can be found, in Diavollo Pizza>. The restriction expected here is `DiavolloPizza` $\subseteq$ (`hasTopping` *some* `TomatoTopping`). Therefore, in the query to identify compatibles properties (equation 2) we may take this issue into account. Moreover, we may have a domain ontology where there are only properties from C_o to C_s.
3. We have performed a concept-driven approach (identification of concepts first and then their compatible properties). In some cases where the identification of the property (by means of a string matching) would have been correct, the concepts are wrongly identified thus leading to an incorrect property identification. We thus have to find hints, for each specific triple whether to have a property or concept-driven approach.
4. Another assumption made in this approach is the fact that C_s and C_o are directly linked. In other words, there is a *one length path* between them. Since we are in a perspective where the formalisation must be done taking into account the complexity of the schema of the ontology, we are not in principle aware of the existence of such a direct link. Thus, we must be able to handle paths made of many edges between concepts.

5. We have focused on object properties-based restrictions. But in practice we may find some restrictions based on data properties. We have in mind mainly the `owl:dataSomeValuesFrom` restriction which can be found for instance in the definition of a *High Calorie Pizza*:
   $\subseteq$ `Pizza` *and* (`hasCalorificContentValue` *some* integer[>= 400])

Going beyond Definitions. As early mentioned in section 4.4, definitions can be done through sentences with a more complex structure than boolean connections between facts (e.g: "***If*** *a pizza is made in Italy* ***then*** *it is an Italian pizza*"). If we are able to join correctly the triples provided by complex NL definitions, it will open the way for a more general task on NL sentences: we mainly think about automatic formalisation of rules.

7 Conclusion

Open Information Extraction is an NLP task which aims to identify all the atomic facts within sentences. In this work we propose an approach to improve current OIE-systems overall performance with domain knowledge. This betterment is done without modifying the *code* of existing tools by handling multi-word expressions (domain terms) at the entry point (detection and shortening before extraction-itself) and the exit point (re-expansion of shortened expressions). This manner to deal with MWE allows us to improve OIE from more than 18%. The ability of OIE to split a sentence into atomic and informative phrases is proposed to formalise automatically NL definitions in `OWL DL`. This formalisation process is straightforward and does not require any learning. In addition to subsumption detection, we are able to identify more complex `OWL DL` restrictions: cardinality and value restrictions. This formalisation task has been evaluated on a corpus made of sentences defining concepts of the pizza ontology. From the 37 sentences considered in the evaluation, 26 have been fully and accurately converted in an `OWL DL` expression. In the future, we must extend the set of possible restrictions actually considered in our approach. Moreover, we must enlarge the size of evaluation corpora. It may lead to some adjustments of some heuristics or weights for string similarity measures we used.

References

1. OWL Web Ontology Language Guide (feb 2004), `http://www.w3.org/TR/owl-guide/`
2. American with Disabilities Act (ADA): 2010 ADA Standards for Accessible Design (sep 2010), `http://www.fire.tas.gov.au/userfiles/stuartp/file/Publications/FireSafetyInBuildings.pdf`
3. Bast, H., Haussmann, E.: Open information extraction via contextual sentence decomposition. In: Semantic Computing (ICSC), 2013 IEEE Seventh International Conference on. pp. 154–159. IEEE Computer Society (2013)
4. Bast, H., Haussmann, E.: More informative open information extraction via simple inference. In: Advances in Information Retrieval. Lecture Notes in Computer Science, vol. 8416, pp. 585–590. Springer International Publishing (2014)

5. Berg, J.: Aristotle's theory of definition. ATTI del Convegno Internazionale di Storia della Logica pp. 19–30 (1982)
6. Bies, A., Ferguson, M., Katz, K., MacIntyre, R., Tredinnick, V., Kim, G., Marcinkiewicz, M.A., Schasberger, B.: Bracketing guidelines for treebank II Style Penn Treebank project. University of Pennsylvania 97 (1995)
7. Bühmann, L., Fleischhacker, D., Lehmann, J., Melo, A., Völker, J.: Inductive lexical learning of class expressions. In: Knowledge Engineering and Knowledge Management, pp. 42–53. Springer (2014)
8. Building Safety Unit Tasmania Fire Service: Fire Safety in Buildings (aug 2002), http://www.fire.tas.gov.au/userfiles/stuartp/file/Publications/FireSafetyInBuildings.pdf, obligaitions of owners and occupiers
9. California Energy Commission: 2008 Building Energy Efficiency Standards (2008), http://www.energy.ca.gov/2008publications/CEC-400-2008-001/CEC-400-2008-001-CMF.PDF, for residential and nonresidential buildings
10. Del Corro, L., Gemulla, R.: Clausie: clause-based open information extraction. In: Proceedings of the 22nd international conference on World Wide Web. pp. 355–366. WWW '13, International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, Switzerland (2013)
11. Fader, A., Soderland, S., Etzioni, O.: Identifying relations for open information extraction. In: Proceedings of the Conference on Empirical Methods in Natural Language Processing. pp. 1535–1545. EMNLP '11, Association for Computational Linguistics, Stroudsburg, PA, USA (2011)
12. Hadjieleftheriou, M., Srivastava, D.: Weighted set-based string similarity. IEEE Data Eng. Bull. 33(1), 25–36 (2010)
13. Horridge, M., Jupp, S., Moulton, G., Rector, A., Stevens, R., Wroe, C.: A Practical Guide To Building OWL Ontologies Using Protégé 4 and CO-ODE Tools Edition1. 2. The University of Manchester (2009)
14. Kacfah Emani, C.H., Ferreira Da Silva, C., Fis, B., Ghodous, P.: Improving Open Information Extraction using Domain Knowledge. In: Surfacing the Deep and the Social Web (SDSW), co-located with The 13th ISWC (Oct 2014)
15. Kacfah Emani, C.H., Ferreira Da Silva, C., Fis, B., Ghodous, P., Khosrowshahi, F.: Structural Sentence Decomposition via Open Information Extraction. In: 18th International Conference Information Visualisation (IV2014) (Jul 2014)
16. Lehmann, J., Auer, S., Bühmann, L., Tramp, S.: Class expression learning for ontology engineering. Web Semantics: Science, Services and Agents on the World Wide Web 9(1), 71–81 (2011)
17. Mausam, Schmitz, M., Bart, R., Soderland, S., Etzioni, O.: Open language learning for information extraction. In: EMNLP-CoNLL. pp. 523–534. Association for Computational Linguistics (2012)
18. Nguyen, V.T., Sallaberry, C., Gaio, M.: Mesure de la similarité entre termes et labels de concepts ontologiques. In: Conférence en recherche d'information et applications. pp. 415–430 (2013)
19. Sayah, K.: Automated Norm Extraction from Legal Texts. Master's thesis, Utrecht University (aug 2004)
20. Tsatsaronis, G., Petrova, A., Kissa, M., Ma, Y., Distel, F., Baader, F., Schroeder, M.: Learning formal definitions for biomedical concepts. In: OWLED (2013)
21. Unger, C., Bühmann, L., Lehmann, J., Ngonga Ngomo, A.C., Gerber, D., Cimiano, P.: Template-based question answering over RDF data. In: Proceedings of the 21st International Conference on World Wide Web. pp. 639–648. WWW '12, ACM, New York, NY, USA (2012)

22. Unger, C., Cimiano, P.: Pythia: Compositional meaning construction for ontology-based question answering on the semantic web. In: Natural Language Processing and Information Systems, Lecture Notes in Computer Science, vol. 6716, pp. 153–160. Springer, Heidelberg (2011)
23. Völker, J., Hitzler, P., Cimiano, P.: Acquisition of OWL DL axioms from lexical resources. In: The Semantic Web: Research and Applications, pp. 670–685. Springer (2007)
24. Völker, J., Rudolph, S.: Lexico-logical acquisition of OWL DL axioms. In: Formal Concept Analysis, pp. 62–77. Springer (2008)
25. Wächter, T., Schroeder, M.: Semi-automated ontology generation within obo-edit. Bioinformatics 26(12), i88–i96 (2010)
26. Winkler, W.E.: The state of record linkage and current research problems. Tech. rep., Statistical Research Division, U.S. Census Bureau (1999)