NALDO: From natural language definitions to OWL expressions

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

NALDO: From Natural Language Definitions to OWL Expressions

Cheikh Kacfah Emani[a,b], Catarina Ferreira Da Silva[b,*], Bruno Fiès[a], Parisa Ghodous[b]

[a] *CSTB 290 Route des Lucioles, 06904 Sophia Antipolis Cedex, France*
[b] *Univ Lyon, Université Claude Bernard Lyon 1, CNRS UMR5205, LIRIS, F-69100, Villeurbanne, France*

Abstract

Domain ontologies are pivotal for Semantic Web applications. The richness of an ontology goes in hand with its usefulness and efficiency. Unfortunately, manually enriching an ontology is very time-consuming. In this paper, we propose to enrich an ontology automatically by obtaining logical expressions of concepts. We present NALDO, a novel approach that provides an `OWL DL` (Web Ontology Language Description Logics) expression of a concept from two inputs: (1) the natural language definition of the concept and (2) an ontology describing the domain of this concept. NALDO uses as much as possible entities provided by the domain ontology, however it can suggest, when needed, new entities. The expressiveness of expressions provided by NALDO covers value and cardinality restrictions, subsumption and equivalence. We evaluate our approach against the definitions and the corresponding ontologies of the BEAUFORD benchmark. Our results show that NALDO is able to perform the correct identification of formal entities with an F1-measure up to 0.79.

Keywords: Ontologies, Natural language definitions, Ontology enrichment, OWL DL, Semantic web

*Corresponding author
Email address: `catarina.ferreira@univ-lyon1.fr` (Catarina Ferreira Da Silva)

Preprint submitted to Data & Knowledge Engineering. *June 9, 2019*
Final version is available at https://doi.org/10.1016/j.datak.2019.06.002

1. Introduction

The usefulness of domain ontologies in various fields is well established. Indeed, they are a pivotal piece when dealing with heterogeneity or complex semantic issues [1, 2, 3, 4]. Building ontologies is a difficult and time-consuming task. It usually requires to combine the knowledge of domain experts with the skills of ontology engineers into a single effort. We believe that this bottleneck currently constitutes a real obstacle for the adoption of semantic technologies into practice. In order to solve this problem, it is worth considering to apply automated techniques to build ontology resources from existing data or at least to assist ontology engineers and domain expert by semi-automatic ways.

Therefore, numerous ontology learning tools have been developed aiming at the automatic or semi-automatic construction of ontologies from structured or unstructured sources. The current state of the art approaches are able to generate lightweight ontology such as [5] and [6]. Some other works are able to generate more expressive ontologies from existing knowledge base (KB) like in [7] and [8], or from natural language definitions like in [9] or [10]. We also have approaches that are able to generate ontologies from sentences written in a given controlled natural language. Among all these approaches only the works described in [7] and [8] propose *formal definition for existing concepts*. Indeed, having a formal definition for concepts in a given ontology facilitates consistency checking and the automatic evaluation of individuals in a KB with regards to that given ontology. However the work in [7] and [8] rely on existing KB having many instances. We argue that, for many domains, it is not guaranteed to have such KB. Approaches like [5] and [6] generate lightweight ontologies, so the inference capabilities of these ontologies are limited. Other approaches, like [9] and [10], do not enhance an *existing ontology* which means that they produce entities in an uncontrolled manner and are not able to find equivalent concepts expressed using different phrasings.

In this work, we propose NALDO, an approach to automatically formalize natural language (NL) definitions of concepts of existing ontologies. We thus enrich these ontologies using the formal definitions that we obtain. For a given ontology, the formal definitions from NALDO reuse foremost the entities found in that ontology, thus NALDO limits the uncontrolled creation of new entities. For instance, in the domain circumscribed by the *Vertebrate Skeletal Anatomy Ontology* (VSAO)[1], NALDO converts the NL definition $\mathcal{S}_1$ "*A cell space is an anatomical space that is part of a healthy cell*" into the expression (using entities of VSAO):

$$\texttt{CellSpace} \sqsubseteq \texttt{AnatomicalSpace} \sqcap \exists \texttt{partOf.Cell}$$

NALDO is consequently an approach that helps ontology engineers and domain experts to enrich their current ontologies with expressive definitions of the concepts of these ontologies, and in practice there are many lightweight ontologies for which a set of acknowledged NL definitions exist. It is the case of OBO foundry set of ontologies (http://www.obofoundry.org/) and BioPortal ontologies (http://bioportal.bioontology.org/).

1.1. Issues related with the automatic formalization of natural language definitions

Getting automatically *a correct expression* from a given natural language (NL) definition, towards a domain ontology, requires to deal with many issues. By correct expression, we mean an expression which is accepted by domain experts without any change on the expression. We list the main issues to tackle in order to solve the automatic formalization of NL definitions problem. We illustrate each issue with several examples.

1.1.1. ($\mathcal{F}_1$) Identification of single definitions in a multiple definition sentence

In practice, a sentence may contain more than one definition. So, when one wants to enrich automatically an ontology using definitions in texts, a first challenge is to identify automatically these definitions. For instance, the sentence

[1] `http://svn.code.sf.net/p/phenoscape/code/tags/vocab-releases/VSAO/vsao.owl`

"*An american pizza is a pizza which has toppings of tomato and a lujuhman pizza is a pizza made with lamb*" contains two distinct definitions: (1) "*An american pizza is a pizza which has toppings of tomato*" and (2) "*A lujuhman pizza is a pizza made with lamb*"

1.1.2. ($\mathcal{F}_2$) Identification of the defined concept and its definition

A same definition can have multiple wordings. The challenge here is to identify the concept that is defined and the phrases that give the definition itself. For example, all the following sentences define the term *cell space*.

- A **cell space** is an anatomical space that is part of a healthy cell.
- An anatomical space that is part of a healthy cell is a **cell space**.
- If an anatomical space is part of a healthy cell then it is a **cell space**.

1.1.3. ($\mathcal{F}_3$)-($\mathcal{F}_4$) Entity linking

The formalization task requires to provide a formal expression of concepts using entities provided in the domain ontology. Hence, a challenge consists of linking the terms of the definition with the corresponding entities in the ontology. This linking can be

($\mathcal{F}_3$) "Basic", which means that there are some similarities between the surface form of the term of the definition and the one of the ontology's entity. For example, when the definition "*An american pizza is a pizza which has toppings of tomato*" is formalized, against the pizza ontology, as $\texttt{AmericanPizza} \sqsubseteq \texttt{Pizza} \sqcap \exists \texttt{hasTopping.TomatoTopping}$, we consider the linking of "*tomato*" to `TomatoTopping` as "basic". Formally, basic linking can be defined as: Given a string s, an ontology $\mathcal{O}$, identify entity $e \in \mathcal{O}$ so that s refers to e. In this task, s and labels of e share common stems[2] (considering also synonyms).

[2] Stem is the root or main part of a word, to which inflections or formative elements are added.

($\mathcal{F}_4$) "Strong", to denote linkings where there is no similarity between the term of the definition and its formal correspondence in the ontology. When the definition "*A lujuhman pizza is a pizza made with lamb*" is formalized, towards the pizza ontology, as `LujuhmanPizza` $\sqsubseteq$ `Pizza` $\sqcap$ $\exists$`hasTopping.MeatTopping`, we consider that the resolution of "*made*" as `hasTopping` and "*lamb*" as `MeatTopping`[3] are both "strong". Formally, strong linking can be defined as: Given a string s, an ontology $\mathcal{O}$, identify entity $e' \in \mathcal{O}$ so that s refers to a hypothetical entity e, not found in $\mathcal{O}$, but semantically related to e'.

1.1.4. ($\mathcal{F}_5$) Pruning

When we transform an NL definition of a concept into a formal expression, some terms of the definition remain unused. So, the formalization prunes these terms. Domain experts, who validate the formalization, assume that those terms are meaningless for this definition. For example, when the definition "*A cell space is an anatomical space that is part of a healthy cell*" is formalized as `CellSpace` $\sqsubseteq$ `AnatomicalSpace` $\sqcap$ $\exists$`partOf.Cell`, the term "*healthy*" is pruned. Formally, pruning can be defined as: Given a string s, an ontology $\mathcal{O}$, discard the words in s so that the intended matching of s to entity e is done correctly.

1.2. Contributions

NALDO presents the following innovative features:

- It is an approach to support ontology engineers in the creation of new OWL DL assertions, by suggesting automatically a formal expression of ontology's concepts using their NL definitions

[3]There is not any term which directly refers to "*lamb*" in the pizza ontology. Thus, since lamb is known as a kind of meat, it is formalized as `MeatTopping`

- It provides the `OWL DL` expression of a definition towards a given domain ontology
- It is domain-independent: we make no assumption on the domain ontology
- It deals with the issues ($\mathcal{F}_3$) and ($\mathcal{F}_5$). We do not focus on issue ($\mathcal{F}_1$) because, usually, definitions are found as isolated sentences (in dictionaries, or in ontologies as value of `skos:definition`, `rdfs:comment` properties [4], etc.). Similarly, since the defined concepts in definitions are generally the starting phrases of definitions, we do not focus on ($\mathcal{F}_2$). ($\mathcal{F}_4$) is out of scope of this paper. However, we deem important to state all the issues ($\mathcal{F}_1$) - ($\mathcal{F}_5$) the community faces when dealing with the automatic formalization of natural language definitions.

The remainder of this paper is organized as follows. First, we present a section devoted to related works (Sect. 2). Next we present our seven-step approach (Sect. 3.1 - 3.3). Then, we bring in our test material and results we obtain when evaluating our approach (Sect. 4). In the light of these results we discuss the strengths of our algorithm and envision possible improvements (Sect. 4.4). Finally, we make a comparison with similar approaches found within current literature (Sect. 5). We end the paper with conclusion and perspectives (Sect. 6).

2. Related Work

Ontologies support many tasks in the field of Semantic Web. Consequently, there is a real interest in developing methods to enrich existing ones or to build new ones from available unstructured data.

Contributing to the achievement of this goal, Wächter and colleagues [5] propose a method to build automatically hierarchies from texts. They have a

[4] Prefixes `skos` stands for http://www.w3.org/2004/02/skos/core# and `rdfs` for and http://www.w3.org/2000/01/rdf-schema#

three-step approach: first, terms are identified within text sources, next, definition of these terms are generated (through a web search) and finally, exploiting the *"is a"* pattern they suggest a hierarchical relation between terms. Still aiming to identify relations between concepts, we have the interesting research work of Tastsaronis et al. [6]. They propose to identify the predicate of a given ontology, which links two terms - labelled with the URIs of the concepts they refer to. The recent research work of Louge et al. [11] fulfills a similar goal by building a taxonomy from services' short descriptions, but a taxonomy is a lightweight ontology and is not expressive enough.

Another way to achieve the goal of adding more information to existing ontologies or knowledge bases (KBs) is to assign formal expressions to concepts. Hence, through reasoning, new pieces of information could be inferred. We find in the literature several research works aiming to provide expressions of concepts. In [7, 12] and [8], Bühmann, Lehman and colleagues try to get such expressions by *studying* the characteristics of concepts' instances in KBs automatically.

However all the works above-mentioned are far from our goal: NALDO addresses the problem of formalizations of given NL definition towards an existing domain ontology.

An excellent system which helps users to obtain formal expression from definitions is the Atempo Controlled English (ACE) [13]. However the input definitions of the ACE are not actually in NL. Indeed, ACE needs the user to write the definitions in a given format, so that the system can translate them automatically and without errors in OWL. For example, the definition we took here to illustrate our approach, "*A cell space is an anatomical space that is part of a healthy cell*", must be rewritten as *A cell-space is an anatomical-space that is part-of a cell.* We see that users have to use *hyphen* to indicate multi-word entities and that they should manually prune "meaningless" terms like *"healthy"* that appear in the NL definition.

Lexo [9] is another interesting approach which is close to NALDO when considering goals and inputs. Lexo is a heuristic-based tool that aims to obtain concepts' expressions from their definitions. Lexo provides the expression of concept from their definition but that expression is not aligned with any ontology. Hence, all its entities are new and still ambiguous. In addition, Lexo proposes a non-contextual formalization of meaningless NL predicates. For instance, when an auxiliary stands by itself as a predicate in an NL phrase, Lexo always assumes that this auxiliary denotes the same formal predicate, independently of the NL phrase or the given definition. Such assumption is too restrictive in practice. Finally, Lexo provides an *unduly* formalization of phrases. Indeed, Lexo formalizes a phrase considering each token of this phrase individually. For example, when following the transformation rules of Lexo, the phrase "red onion topping" in a definition will be turned into '$\texttt{:Red} \sqcap \texttt{:Onion} \sqcap \texttt{:Topping}$', which does not consider that this phrase may refer to a single concept. When refering to the PIZZA ontology in this case, this phrase could refer to `:RedOnionTopping`.

The approach of De Azevedo et al. [10] is similar to Lexo. On the contrary of Lexo, [10] uses reasoning to ensure the consistency of their expressions and to provide other expressions which can be semantically derived from the main expression, i.e. directly obtained from the definition. Such as Lexo, [10] builds DL ALC expressions from scratch, without trying to reuse entities from a domain ontology.

All these differences, regarding inputs (i.e. definitions and corresponding domain ontology), and goal (i.e. formalize the definitions w.r.t a corresponding domain ontology) do not allow us to make a quantitative comparison with all those existing works. However, we provide a detail comparison of NALDO with Lexo and the work of De Azevedo et al. [10] in section 5. We recall that NALDO is an approach whose goal is to support ontology building by suggesting automatically a formal expression of ontology concepts based on their NL definition. The main advantage of NALDO is that it foremost reuses the ontology entities in the expression that it provides.

Now that we have described state of the art research works, we present the steps of NALDO.

3. The NALDO approach

Fig. 1 gives an overview of the main steps of NALDO which are then detailed in sections 3.1 to 3.3. A demo is available at `http://tinyurl.com/j6vfuj5`. The NALDO approach is composed of the following steps: 1) Splitting, 2) Lemmatisation + Syntactic Pruning, 3) Semantic Pruning, 4) Concepts Identification, 5) Template Identification, 6) Property Identification and finally 7) Recombination. Steps 1 to 3 reuse existing tools, whereas steps 4 to 7 comprise our contributions.

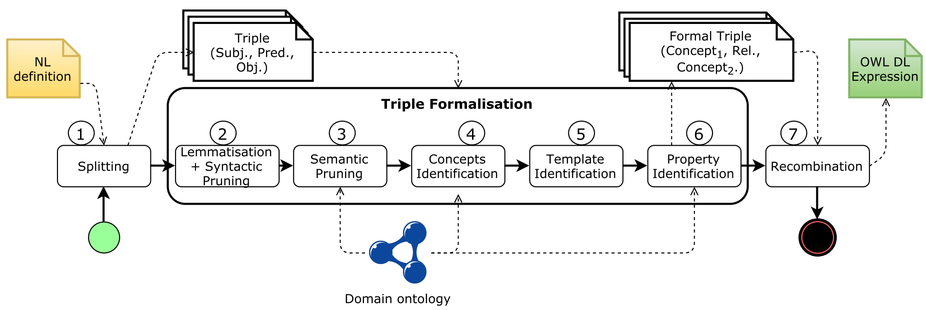

Figure 1: Steps of NALDO

3.1. Splitting

Splitting is the first step of the processing of a NL definition (see step 1 Fig. 1). The aim of this task is to identify all the pieces of information within the definition $\mathcal{S}_j$. Each chunk is intended to be *informative* and *atomic*. The informativeness means that a chunk delivers a piece of information on a given subject. We take advantage of the suitable task called Open Information Extraction (OIE) for this splitting. An OIE-tool generally provides each piece of information in form of a triple $\langle$ subject, predicate, object $\rangle$ [14], [15]. In addition, atomicity guarantees that each triple is small enough so that we cannot extract any other piece of information from it.

In this work, we have chosen the system called `CSD-IE`[5] of Bast and Haussmann [15] to perform OIE. First, let us mention that to the best of our knowl-

[5] `http://ad-wiki.informatik.uni-freiburg.de/research/Projects/CSDIE`

edge, `ClausIE` [14] and `CSD-IE` are the most efficient OIE-tools. Secondly, characteristics of `CSD-IE` in comparison with `ClausIE` make the former more suitable for us than the latter, because it presents the following aspects:

- **Minimality**, i.e. *atomicity* mentioned above, and **coverage**, which means that there is a concern to use each word from the original sentence at least once in any resulting triple.

- **N-uples.** Instead of presenting an assertion in the form of a triple, `CSD-IE` can provide *n-uples* also called *tuples* i.e. $\langle$`subject, predicate, phrase`$_1$`, phrase`$_2$,...$\rangle$. For instance, in addition to the triple $\tau_0 = \langle$`A cell space, is, an anatomical space`$\rangle$, `CSD-IE` returns this other tuple $\langle$`an anatomical space, is, part, of a healthy cell`$\rangle$ from the sentence $\mathcal{S}_1$. From this tuple, we can obtain:

  1. A single triple, for instance
     - $\tau_1 = \langle$an anatomical space, is, part of a healthy cell$\rangle$ or
     - $\tau_a = \langle$an anatomical space, is part, of a healthy cell $\rangle$
  2. Or a set of triples, for example
     - $\tau_\alpha =\langle$ an anatomical space, is, part $\rangle$ and $\tau_\beta =\langle$ part, of, a healthy cell $\rangle$

  Tuples allow us to consider different levels of granularity for triples. Hence, the possible cuttings of $\mathcal{S}_1$ are 'τ_0 `and` τ_1', 'τ_0 `and` τ_a', and 'τ_0 `and` τ_α `and` τ_β'.

3.2. Rewriting a Triple as an OWL DL Statement

The goal of this task is to write each triple τ_i, made of chunks in NL, as an `OWL DL` statement. The formal expression we obtain denotes a relation between two concepts w.r.t to a given domain ontology $\mathcal{O}$, *if possible*. We denote respectively $?C_s$ and $?C_o$ the concepts or individuals that stand in the position of the *subject* and the *object* within the formal expression. In practice, this task consists in instantiating one of the templates presented in Table 1.

To achieve this goal, we proceed in four steps: *(i)* The lemmatisation and the syntactic pruning of the triple, *(ii)* the semantic pruning, *(iii)* instantiation

Names	Expressions
Subsumption	$?C_s \sqsubseteq ?C_o$
Equivalence	$?C_s \equiv ?C_o$
Existential restriction	$?C_s \sqsubseteq \exists ?property.?C_o$
Universal restriction	$?C_s \sqsubseteq \forall ?property.?C_o$
HasValue restriction	$?C_s \sqsubseteq \exists ?property.?C_o$, where $?C_o$ is an individual
MinCardinality Restriction	$?C_s \sqsubseteq \geq ?n\, ?property.?C_o$
MaxCardinality Restriction	$?C_s \sqsubseteq \leq ?n\, ?property.?C_o$
Cardinality Restriction	$?C_s \sqsubseteq = ?n\, ?property.?C_o$

Table 1: Templates of atomic `OWL DL` restrictions (variables are preceded by '?')

of $?C_s$ and $?C_o$, *(iv)* identification of the corresponding template and *(v)* instantiation of the variable $?property$. These steps are numbered 2 to 6 in Fig. 1.

3.2.1. Lemmatisation and Syntactic Pruning

The goal of this step is to have canonical forms of the tokens (i.e. lexical units) within the triples, and to remove from the triples all the stop words. At this stage, lemmatisation and syntactic pruning are carried out for each component (i.e subject, predicate and object) of τ_i.

3.2.2. Semantic Pruning

Between the tokens remaining in τ_i after the syntactic pruning, we can still have some noise. Indeed, we consider that tokens (at this stage) in τ_i which stems are not shared with those of the labels of domain entities, even when considering their synonyms[6], are useless for the linking of $\mathcal{O}$ and τ_i. Using this pruning prevents the overloading of the triple, with a token that will not match any token of the labels of entities in $\mathcal{O}$. Table 2 gives the results for lemmatisation, syntactic and semantic prunings of our running example. We notice that, the semantic pruning removes the token *"healthy"* from all the triples where this token is found. So, the word *"healthy"* will not cause any

[6]We use the paraphrases database PPDB of Pavlick et al. [16] to identify synonyms.

NL Phrases	**Triples**	**Lemma + Syn. Pruning**	**Semantic Pruning**
A cell space	τ_0	cell space	cell space
is	τ_0, τ_1, τ_α	-	-
an anatomical space	τ_0, τ_1, τ_a, τ_α	anatomical space	anatomical space
part of a healthy cell	τ_1	part healthy cell	part cell
is part	τ_a	part	part
of a healthy cell	τ_a	healthy cell	cell
part	τ_α, τ_β	part	part
of	τ_β	-	-
a healthy cell	τ_β	healthy cell	cell

Table 2: Some examples of lemmatisation, syntactic and semantic pruning.

distortion when we will look for the entities, of the domain ontology, that match the NL phrases of the triples.

3.2.3. Concepts Identification

Concepts identification is the 4th step of Fig. 1. In our approach, we uppermost identify concepts instead of properties. Indeed, there are too many ways to assert the same thing about a given resource (e.g: *"an anatomical space* ***is part of/is a region of/composes*** *a healthy cell"*). Moreover, some predicates/verbs are useless for property identification (e.g: be, have, of, from, etc.). Also, at this stage we deal with an atomic triple and its structure already suggests a group of words to bind $?C_s$ and another one for $?C_o$. NALDO relies on the algorithm 1 to bind $?C_o$ and $?C_s$.

Algorithm 1 works as follows:

- To instantiate $?C_s$ (resp. $?C_o$), we use the subject (resp. the object) part of τ_i (Line 1).
- For the subject and object of τ_i (*after their syntactic and semantic prunings* - lines 1 and 2), NALDO identifies the entity within $\mathcal{O}$ with the highest matching score (loop on lines 3-8).

Algorithm 1: Binding of $?C_s$ for the triple τ_i and score computation

Input : τ_i, $\mathcal{O}$
Output: $?C_s$, scoreCs

1 subjectSynP $\leftarrow$`lemmatisationAndSynPruning` (τ_i.subject);
2 subjectSemP $\leftarrow$`SemanticPruning` (subjectSynP); scoreCs $\leftarrow 0.0$;
3 **for** *each* uri *in* $\mathcal{O}$.classesAndIndividuals **do**
4   **for** *each* label *in* uri.altLabels **do**
5   temp $\leftarrow$ `Similarity` (label, subjectSemP);
6   **if** temp $>$ scoreCs **then** $?C_s \leftarrow$ uri; scoreCs $\leftarrow$ temp;
7   **end**
8 **end**
9 **if** scoreCs $< \sigma_c$ **then**
10     $?C_s \leftarrow$ `newUri` (subjectSynP); nbOfTokens $\leftarrow$ subjectSynP.nbOfTokens;
11   scoreCs $\leftarrow \mu_c/($nbOfTokens $+$ `max`$(0,$ nbOfTokens $-$ $\mathcal{O}$.nbOfTokensClass$))$;
12 **end**

- The string similarity function used here (Line 5) is the average of the *jaccard_2 similarity*[7] [17] and the *cosine similarity* [19]. This matching score allows taking into account a similarity at word-level, with the cosine, and at the character level, with the jaccard_2 similarity.

- When the similarity score is under a certain threshold σ_c(line 9), a new entity (line 10) having a new score (line 11) is created. This new score, capped to a value μ_c, decreases with the number of tokens of the label of the new entity and the average number of tokens of labels in $\mathcal{O}$ (variable name nbOfTokensClass). Tab 4 shows result for our running example.

3.2.4. Template Identification

The templates we have to deal with are built around a given set of operators ($\sqsubseteq$, $\equiv$, $\forall$, $\exists$, etc.). To identify the correct template of τ_i means to find out in

[7] jaccard_2 similarity [17] is an adaptation of the jaccard similarity [18]: jaccard computes the similarity between two bags of words, where jaccard_2 considers each word in the bags as a set of two-characters words.

List of keywords	**Corresponding template**	**Symbol**
is a, is an, are a	Subsumption	$\sqsubseteq$
is any, are any, are all	Equivalence	$\equiv$
at least n, greater than n	MaxCardinality Restriction	$\leq n$
at most n, less than n	MinCardinality Restriction	$\geq n$
with n, exactly n, uniquely n	Cardinality Restriction	$= n$
only, uniquely, exclusively	Universal Restriction	$\forall$

Table 3: Some examples of phrases for template identification (n = digits)

	NL Subject $\rightarrow$ Formal Entity/Score	**NL Object $\rightarrow$ Formal Entiy/Score**	**Sym.**
τ_0	cell space $\rightarrow$ `CellSpace`/1.0	anatomical space$\rightarrow$`AnatomicalSpace`/1.0	$\sqsubseteq$
τ_1	anatomical space $\rightarrow$ `AnatomicalSpace`/1.0	part cell$\rightarrow$`new:PartHealthyCell`/0.3	$\exists$
τ_a	anatomical space $\rightarrow$ `AnatomicalSpace`/1.0	cell $\rightarrow$ `Cell`/1.0	$\exists$
τ_α	anatomical space $\rightarrow$ `AnatomicalSpace`/1.0	part $\rightarrow$ `new:Part`/0.5	$\exists$
τ_β	part $\rightarrow$ `new:Part`/0.5	cell $\rightarrow$ `Cell`/1.0	$\exists$

Table 4: Concepts, matching scores and templates' symbols for our triples. The prefix `new:` indicates new entities (not found in $\mathcal{O}$)

the text of τ_i any word or group of words that could lead to one of the operators we just listed.

This identification process takes advantage of a list of predefined phrases as illustrate in Table 3. Practically, we first verify if $?C_o$ is an *individual*. If yes, we are facing a HasValue restriction. If not, we find, in the *order* exposed in Table 3, which phrases the triple τ_i (without any processing) matches. The existential restriction $\exists$ is the default template because it states that there exists a link between two entities. Table 4 presents in its last column the template for our example.

3.2.5. Properties Identification

Now that we have possible concepts $?C_o$ and $?C_s$, we need to provide the *property* that supports the relation between them. Let us note that this step is useless when we already know that we are facing a classic subsumption or an equivalence. We obtain the list of properties that can link $?C_s$ and $?C_o$ by executing the following query displayed in listing 1:

```
1 PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
2 PREFIX owl: <http://www.w3.org/2002/07/owl#>
3 PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
4 SELECT DISTINCT ?p  WHERE {
5 {{?hierarchySubj owl:onProperty ?p.
6 :Cs rdfs:subClassOf* ?hierarchySubj.}
7 UNION
8 {:Cs rdfs:subClassOf* ?DomainP. ?p rdfs:domain ?DomainP}}.
9 {{?r2 owl:onProperty ?p. ?r2 ?prop ?hierarchyObj.
10 :Co rdfs:subClassOf* ?hierarchyObj.
11 FILTER(!(?prop = rdf:type) && !(?prop = owl:onProperty))}
12 UNION
13 {:Co rdfs:subClassOf* ?RangeP. ?p rdfs:range ?RangeP}}}
```

Listing 1: Query for property identification

This query of listing 1 allows NALDO to select a property `?p` if:

- $?C_s$ is included in the domain of `?p` (line 8) *or* $?C_s$ is included in an `owl:Restriction` built on top of `?p` (lines 5-6) *and*
- $?C_o$ is included in a class targeted by a `owl:Restriction` built on top of `?p` (lines 10-12) *or* $?C_o$ is included in the range of `?p` (line 13).

Then we have to rank this list of properties according to the tokens of τ_i. This ranking is similar to the one performed to bind concepts. Algorithm 2 details the binding of the *?property* for τ_i. In this algorithm, when the property of the triple has only meaningless tokens (usually auxiliaries or prepositions), its matching score depends on the tokens of the τ_i and the (potential) additional tokens from labels of $?C_s$ and $?C_o$ (line 8 in the algorithm).

Algorithm 2: Binding of $?property$ for the triple τ_i and score computation

Input : τ_i, $\mathcal{O}$, $?C_s$, $?C_o$

Output: $?property$, scoreP

tripleUsefulTokens $\leftarrow$ `lemmatisationAndSynPruning` (τ_i);
tripleUsefulTokens.add ($?C_s$.label); tripleUsefulTokens.add ($?C_o$.label);
predicate $\leftarrow$`lemmatisationAndSynPruning` (τ_i.predicate);
properties $\leftarrow$ `compatibleProps` ($?C_s$,$?C_o$); scoreP $\leftarrow$ 0.0;
if predicate = "" **then**
    for *each* property *in* properties **do**
        for *each* label *in* property.altLabels **do**
            temp $\leftarrow$ `cosineSimilarity` (label, tripleUsefulTokens);
            if temp $>$ scoreP **then** $?property$ $\leftarrow$ property; scoreP $\leftarrow$ temp;
        end
    end
else
    for *each* property *in* properties **do**
        for *each* label *in* property.altLabels **do**
            temp $\leftarrow$ `NgramSimilarity` (label, predicate, 2);
            if temp $>$ scoreP **then** $?property$ $\leftarrow$ property; scoreP $\leftarrow$ temp;
        end
    end
end
if scoreP $<$ σ_p **then**
    $?property$ $\leftarrow$ `newUri` (predicate); nbOfTokens $\leftarrow$ predicate.nbOfTokens;
    scoreP $\leftarrow$ $\mu_p/(\text{nbOfTokens} + \max(0, \text{nbOfTokens} - \mathcal{O}.\text{nbOfTokensProps}))$;
end

The splitting of $\mathcal{S}$ provides a collection of sets of triples. Each set of triple is a candidate to the formalization of the definition. To choose the most suitable set of triples, NALDO ranks this collection. To perform this ranking, we assign a score to each formal triple. This score depends on `ScoreCs`, `ScoreCo` obtained from algorithm 1 and `ScoreP` from algorithm 2 (we consider that triples denoting subsumption or equivalence, have property score of 1.0). The score of a cutting is the *average* of the score of its triples.

Having a single set of formal triples, we need to put them together to obtain a formal expression for the input definition. For our running example, the properties and the final scores of triples are presented in Table 5. We recall that we have three possible cuttings: 'τ_0 **and** τ_1' (score = **2.325**), 'τ_0 **and** τ_a' (score = **2.825**), and 'τ_0 **and** τ_α **and** τ_β' (score = **2.33**). In the light of these scores, we will use τ_0 and τ_a for the final expression of $\mathcal{S}_1$.

Triples	**Formal expression**	**Scores**
τ_0	`CellSpace`$\sqsubseteq$ `AnatomicalSpace`	$1.0 + 1.0 + 1.0 = \mathbf{3.0}$
τ_1	`AnatomicalSpace` $\sqsubseteq$ $\exists$`hasPart.new:PartHealthyCell`	$1.0+0.35+0.3 = \mathbf{1.65}$
τ_a	`AnatomicalSpace` $\sqsubseteq$ $\exists$`partOf.Cell`	$1.0+0.65+1.0 = \mathbf{2.65}$
τ_α	`AnatomicalSpace` $\sqsubseteq$ $\exists$`partOf.new:Part`	$1.0+0.35+0.5 = \mathbf{1.85}$
τ_β	`new:Part` $\sqsubseteq$ $\exists$`partOf.Cell`	$0.5+0.65+1.0 = \mathbf{2.15}$

Table 5: Properties and final scores, after running algorithms 1 and 2, for τ_0, τ_1, τ_a, τ_α and τ_β (with (σ_c, μ_c, σ_p, μ_p) = (0.40, 0.35, 0.35, 0.30))

3.3. Recombination

Recombination is the 7th (Fig. 1) and final step of the formalization process. Its aim is to combine a set of formal triples to obtain a single expression. Besides the triples to be combined, we compute the structure Σ_j that gives information on the manner these triples are linked. To perform this merging, we rely on two main functions that we call *factorization* and *refinement*. They both take as input two triples ($\tau_i = (C_s^i\, \texttt{rel}_i\, C_o^i)$ and $\tau_j = (C_s^j\, \texttt{rel}_j\, C_o^j)$), an operator (`and` or `or`) and return a new triple that is a merging of the input triples. To be coherent with the possible formal expression of a triple, $\texttt{rel}_i$ and $\texttt{rel}_j$ can only be subsumption, equivalence or the body of a restriction. We detail factorization and refinement in the following subsections.

3.3.1. Factorization

We perform this operation when $C_s^i = C_s^j = C_s$. The two triples thus make assertions about the same subject. The resulting triple regroups these two assertions into a single one about C_s. Equations (1) - (8) below ($*$ denotes `and` or `or`) provide all the possible results of factorisation. We make use of lower case for presentation purposes.

$$(c_s \sqsubseteq c_o^i) * (c_s \equiv | \sqsubseteq c_o^j) \rightarrow c_s \sqsubseteq (c_o^i * c_o^j) \tag{1}$$
$$(c_s \equiv c_o^i) * (c_s \equiv c_o^j) \rightarrow c_s \equiv (c_o^i * c_o^j) \tag{2}$$
$$(c_s \equiv | \sqsubseteq c_o^i) * (c_s \sqsubseteq c_o^j) \rightarrow c_s \equiv (c_o^i * c_o^j) \tag{3}$$
$$(c_s \sqsubseteq c_o^i) * (c_s\, r_j\, c_o^j) \rightarrow c_s \sqsubseteq (c_o^i * r_j\, c_o^j) \tag{4}$$
$$(c_s \equiv c_o^i) * (c_s \sqsubseteq r_j\, c_o^j) \rightarrow c_s \sqsubseteq (c_o^i * r_j\, c_o^j) \tag{5}$$
$$(c_s \sqsubseteq r_i\, c_o^i) * (c_s \sqsubseteq c_o^j) \rightarrow c_s \sqsubseteq (r_i c_o^i * c_o^j) \tag{6}$$
$$(c_s \sqsubseteq r_i\, c_o^i) * (c_s \equiv c_o^j) \rightarrow c_s \sqsubseteq (r_i c_o^i * c_o^j) \tag{7}$$
$$(c_s \sqsubseteq r_i\, c_o^i) * (c_s \sqsubseteq r_j\, c_o^j) \rightarrow c_s \sqsubseteq (r_i c_o^i * r_j\, c_o^j) \tag{8}$$

$(C_s \sqsubseteq r_i\, C_o^i)$ (in equation (8)) means members of the class C_s are also members (subsumption) of the anonymous class $r_i\, C_o^i$. Likewise, $(C_s \sqsubseteq r_j\, C_o^j)$ means $C_s \sqsubseteq (r_j\, C_o^j)$. Thus, the first member of (8) means $(C_s \sqsubseteq r_i\, C_o^i)$ `and|or` $(C_s \sqsubseteq r_j\, C_o^j)$. We can conclude that C_s subsumes the combination (`and` or `or`) of $r_i\, C_o^i$ and $r_j\, C_o^j$. Moreover, when we have to put together subsumption and equivalence, we decide to choose subsumption, since equivalence includes subsumption.

3.3.2. Refinement

In this work, to refine a concept is to add a precision to make it a more specific concept. We perform refinement when $C_o^i = C_s^j$. Hence, it means that a precision is made about C_o^i in the second triple τ_j. All the possible scenarios of refinement are solved through equations (9) - (16) below. Let us note that using the connector `and` is the only way to handle the piece of information brought by τ_j (one cannot provide further information on an entity using `or`).

$$(c_s \sqsubseteq c_o^i) \sqcap (c_o^i \sqsubseteq | \equiv c_o^j) \rightarrow c_s \sqsubseteq (c_o^i \sqcap c_o^j) \tag{9}$$
$$(c_s \equiv c_o^i) \sqcap (c_o^i \equiv c_o^j) \rightarrow c_s \equiv (c_o^i \sqcap c_o^j) \tag{10}$$
$$(c_s \equiv | \sqsubseteq c_o^i) \sqcap (c_o^i \sqsubseteq c_o^j) \rightarrow c_s \sqsubseteq (c_o^i \sqcap c_o^j) \tag{11}$$
$$(c_s \sqsubseteq c_o^i) \sqcap (c_o^i \sqsubseteq r_j\, c_o^j) \rightarrow c_s \sqsubseteq (c_o^i \sqcap r_j\, c_o^j) \tag{12}$$
$$(c_s \equiv c_o^i) \sqcap (c_o^i \sqsubseteq r_j\, c_o^j) \rightarrow c_s \equiv (c_o^i \sqcap r_j\, c_o^j) \tag{13}$$
$$(c_s \sqsubseteq r_i\, c_o^i) \sqcap (c_o^i \sqsubseteq c_o^j) \rightarrow c_s \sqsubseteq (r_i(c_o^i \sqcap c_o^j)) \tag{14}$$
$$(c_s \sqsubseteq r_i\, c_o^i) \sqcap (c_o^i \equiv c_o^j) \rightarrow c_s \sqsubseteq (r_i(c_o^i \sqcap c_o^j)) \tag{15}$$
$$(c_s \sqsubseteq r_i\, c_o^i) \sqcap (c_o^i \sqsubseteq r_j\, c_o^j) \rightarrow c_s \sqsubseteq (r_i(c_o^i \sqcap r_j\, c_o^j)) \tag{16}$$

To evaluate the simplification of a complex expression, we use the *recursive factorisation algorithm* proposed in [20, p. 13].

For our running example, the expression to recombine is 'τ_0 `and` τ_a' i.e.:
$(\texttt{CellSpace} \sqsubseteq \texttt{AnatomicalSpace}) \sqcap (\texttt{AnatomicalSpace} \sqsubseteq \exists\texttt{partOf.Cell})$
This expression strictly follows the refinement equation 12. We thus obtain $\texttt{CellSpace} \sqsubseteq (\texttt{AnatomicalSpace} \sqcap \exists\texttt{partOf.Cell})$.

We provide this running example with details at `http://tinyurl.com/h7vgo3r`.

4. Evaluation

4.1. Benchmark and Metrics

We evaluate NALDO against the BEAUFORD benchmark [21]. BEAUFORD is a benchmark dedicated to the formalization of definitions. It provides a list of features for a formalization approach, a dataset and a set of metrics for the current task.

- BEAUFORD provides a corpus made of three domain ontologies (`VSAO`, the Semantic Sensor Network Ontology (`SSN`) and the pizza ontology (`PIZZA`)) and 25 definitions for each of these domains. Knowing that a definition can be formalized in many ways, BEAUFORD provides a set of possible formalizations for each definition. For the current evaluation, considering the goal of NALDO, we consider only the formalizations that give priority to entities found in the ontologies[8].

- Finally, BEAUFORD proposes three metrics to evaluate formalization of definitions: *precision*, *recall* and *confidence*. Basically, within BEAUFORD, precision denotes the ratio of formal entities correctly identified, recall quantifies the number of NL phrases within the definition that are correctly formalized and the confidence is the ratio of definitions of the corpus that are correctly formalized in all points. [21] provides additional information and illustrations about these metrics.

4.2. Evaluation Results

A demo of NALDO is accessible at http://tinyurl.com/j6vfuj5. NALDO is written in Java and uses the Stanford suite[9] for natural language processing tasks.

For each of the three domain ontologies of the BEAUFORD corpus, we have computed precision, recall and F1-measure. These metrics are function of the values of the parameters σ_c, μ_c, σ_p and μ_p (Fig. 2-4). These parameters serve to control the quality of the matching between terms of the NL definitions and labels of ontologies' entities (algorithms 1 and 2). Accordingly, an interpretation

[8] Indeed some formalizations encourage the use of new entities to express a definition. See [21, Sect. 5.2 and 5.3] for details.

[9] https://nlp.stanford.edu/software/

of Fig. 2-4 gives us the highest values for parameters σ_c, μ_c, σ_p and μ_p to use in concepts and properties identification. We consider:

- $\sigma_c > \mu_c$ and $\sigma_p > \mu_p$. For the creation of a new class, the string matching score must be under σ_c. Since we want to discourage the creation of new entities, we find suitable to top the score of a new class to a value μ_c that is less than σ_c. The same explanation holds for σ_p and μ_p.
- $\mu_c = \sigma_c - 0.05$ and $\mu_p = \sigma_p - 0.05$. Since we cannot evaluate NALDO for all the possible values of our four parameters, these equations help us to reduce the number of tests to carry on. Hence, precision, recall and F1 are computed only as function of σ_c and σ_p and their values represented using a 3D diagram (Fig. 2 - 4). In these diagrams, the color varies from the lowest to the highest values on the z-axis, respectively from *blue*, then *yellow*, then *orange* and finally *red*.

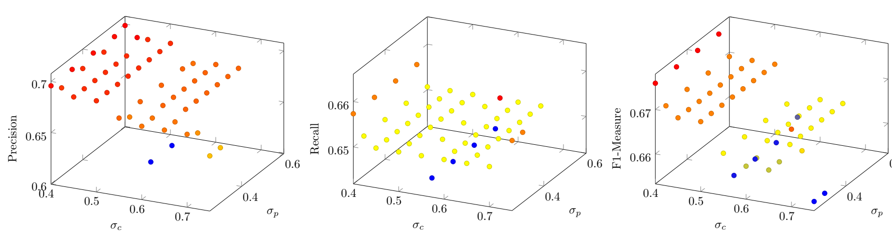

Figure 2: Precision, Recall and F1-Measure for the `VSAO` domain.

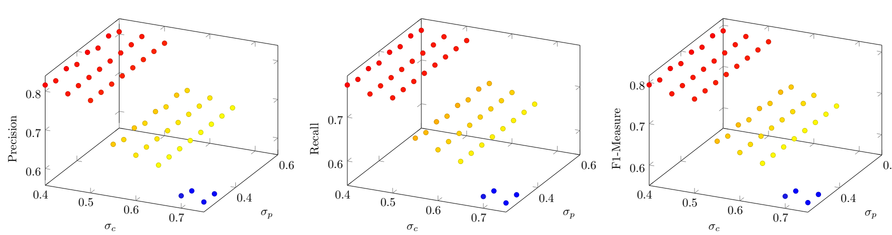

Figure 3: Precision, Recall and F1-Measure for the `PIZZA` domain.

Finally, for each domain ontology, for the 4-uple (σ_c, μ_c, σ_p, μ_p) for which the F1-measure is the highest, we have computed confidence, i.e., the number of definitions that have the formal expression correct [21]. Table 6 presents these

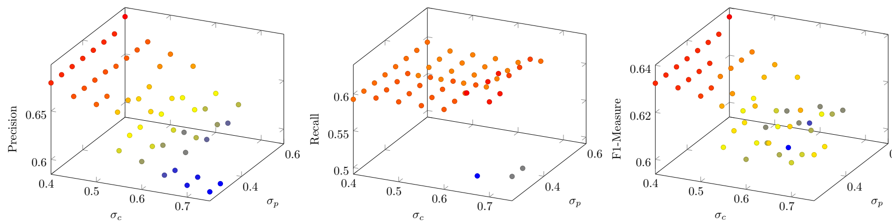

Figure 4: Precision, Recall and F1-Measure for the `SSN` domain.

results. It is useless to find confidence for every 4-uples (σ_c, μ_c, σ_p, μ_p), because a low F1 means a misidentification of several formal entities and thus incorrect final expressions.

For each of the 75 definitions of the BEAUFORD dataset, the detailed results are available at `http://tinyurl.com/hq8vlou`.

Domains	**Best F1-measure**	(σ_c, μ_c, σ_p, μ_p)	**Confidence**
`VSAO`	0.68	(0.40,0.35,0.25,0.20)	07/25=28%
`PIZZA`	0.79	(0.40,0.35,0.35,0.30)	08/25 = 32%
`SSN`	0.63	(0.40,0.35,0.55,0.50)	08/25=32%

Table 6: Confidence for `VSAO`, `PIZZA` and `SSN` calculated at the best F1 score.

4.3. Results Analysis

The analysis of our results shows that:

- For the three domain ontologies, the best F1-measure is obtained with low thresholds. Indeed, σ_c is under 0.40 and σ_p is less than 0.60. It means that, usually, the NL phrases in definitions are built upon a terminology wider than the one covered by domain ontologies. Additionally, NL definitions are not concise and surround the key pieces of information with attractive phrasing and wordy details. Consequently, to gain in logical expressiveness, lightweight ontologies need to represent a more important number of real world entities.

- The best 4-uple of parameters differ from a domain to another. This discrepancy underlines well the differences between the three ontologies. For instance, a domain and a range of almost all the properties within `PIZZA` ontology are provided, when it is the case for very few of them in `VSAO`. Hence, the list of compatible properties between two concepts in `PIZZA` is less noisy properties (hence a better matching) than a similar list in `VSAO`. Another key point here is the number of tokens used for labels of entities. The average number of tokens for `SSN` is 1.49, 1.66 for `PIZZA` when it reaches 2.08 for `VSAO`. These values impact the best parameters of each ontology. Indeed, thresholds decrease from `SSN` to `PIZZA` and to `VSAO` (third column of Table 6): higher threshold to match a 1.5-token expression than for the matching of a 2 or more tokens expression.

- The values of the confidence, as defined in BEAUFORD benchmark [21], are very low - under 30 %, especially when compared to F1 measures. It means that, in about 70 % of the formal expression, NALDO misidentifies at least one entity. We present the main sources of errors in the next section.

4.4. Errors Analysis

The main sources of errors in NALDO and some possible improvements follow.

- **OIE.** The first step of the formalization process is the splitting of the definition. Although OIE helps to handle long and complex sentences, it is not free of error. Hence, when OIE fails, even if we can identify some entities correctly, the final expression may be wrong. The result of OIE itself depends directly of the quality of the parser. CSD-IE, that we use for OIE, uses the Stanford parser. For example the sentence *"A spicy topping is a kind of topping"* leads to incorrect tuple (A spicy, is topping) because topping is tagged as a verb (VBG), instead of a noun (NN), by the parser. A training of parsers on annotated corpus built from resources of the domain of interest may lead to better result of parsing, and thus improve information extraction from definitions of this domain.

- **Thresholds for entities identification.** The identification of entities relies strongly on the values of σ_c and σ_p. These thresholds can lead to the refusal of a good matching or the approval of a wrong one.

- **Paraphrases resolution (PR).** Although the advantages of such task are indisputable, PR causes some problems. PPDB [16] is a general and incomplete database. Consequently, some paraphrases that are true in general are false in a given domain and some paraphrases may be missing. These phenomena may cause unwanted alignments. The use of a paraphrase resource, built specially for the domain of the ontology one aims to enrich, should improve results of this task.

- **"One to one alignment".** Except the syntactic and semantic prunings, which allow us to avoid useless tokens in string matching, we consider that each NL phrase and each triple is worth for the formalization process. However, in practice, a whole piece of a definition may be useless. For instance, in the definition 11 "Deployment is the ongoing process of entities **deployed for a particular purpose**." of the `SSN`-BEAUFORD corpus, the phrase in bold is useless for the formalization suggested by the gold standard in BEAUFORD. We must enhance our pruning method to achieve a higher precision.

4.5. NALDO in practice

Here we discuss briefly how one can use NALDO in practice. For example for a new ontology, the first step may consist in finding suitable parameters (σ_c, μ_c, σ_p, μ_p) for concepts and properties matching. This can be done by evaluating the approach on a small set of definitions (training set) for many values and choose the parameters which provide the best F1-measure on that training set. Then using these parameters (or default ones), an ontology designer can use NALDO to get a proposal of formal expressions of concepts given their NL definitions. Outputs of NALDO can be then added, with or without update, to the ontology.

5. Comparison of Naldo, Lexo [9] and the work of De Azevedo et al. [10]

In this section, we detail the comparison between NALDO and some state-of-the art approaches. Table 7 presents the characteristics of each of these works. This table uses the issues ($\mathcal{F}_1$) - ($\mathcal{F}_5$), provided in section 1.1, to ease the comparison between approaches and to see clearly what is done and what is still to address by future works. In addition, this table shows that the main difference between NALDO and the existing works is the ability of NALDO to link the terms of the NL definition with the entity of the domain ontology. Hence, the expression that NALDO provides can be added directly to the ontology.

This table shows that:

- Neither Lexo, nor De Azevedo et al.'s work tackle issues ($\mathcal{F}_1$)-($\mathcal{F}_5$) (see section 1.1), while Naldo addresses issues ($\mathcal{F}_3$) and ($\mathcal{F}_5$). In other words, NALDO implements mechanisms for "basic" linking (($\mathcal{F}_3$), as explained in section 1.1.3) and for pruning of meaningless phrases found in definitions.
- The evaluation of the three approaches in three different domains highlights that NALDO outperforms both Lexo and De Azevedo et al. approach by at least 0.13 up to 0.38 of F1 measure.

6. Conclusion and perspectives

In this paper, we present NALDO, an automatic approach that provides formal expression of concepts from their textual definitions. After the presentation of the research problems related to the automatization of such approach, we show how NALDO focuses on the issues of basic linking and pruning. To address these issues and to provide expressions with a given semantics, NALDO uses entities of existing ontologies describing the domain of these concepts. NALDO

Characteristics	Lexo [9]	Work of De Azevedo et al. [10]	NALDO
Input	NL definition	NL definition	NL definition and Domain ontology
Expressiveness of output	SHROIQ	ALC	SHROIQ
Issues			
($\mathcal{F}_1$)	No	No	No
($\mathcal{F}_2$)	No	No	No
($\mathcal{F}_3$)	No	No	Yes
($\mathcal{F}_4$)	No	No	No
($\mathcal{F}_5$)	No	No	Yes
Results on VSAO			
Precision	0.52	0.61	0.70
Recall	0.44	0.44	0.66
F1	0.48	0.51	0.68
Results on PIZZA			
Precision	0.40	0.42	0.82
Recall	0.40	0.4	0.77
F1	0.40	0.41	0.79
Results on SSN			
Precision	0.47	0.53	0.68
Recall	0.45	0.48	0.59
F1	0.46	0.50	0.63

Table 7: Comparison of NALDO with Lexo and De Azevedo et al. [10]

can achieve up to 0.79 of F1-measure and a confidence of 31% i.e. 23 of the 75 tested definitions. The given score is obtained for a unique combination, for a given domain, of the four key parameters which support NALDO. The combination can be used to extrapolate NALDO on a larger corpus. This work also revealed several crucial issues, of which one is prominent: the pruning, from a text, of meaningless phrases w.r.t. to a knowledge base and thus useless for formal expressions of concepts. NALDO addresses this issue using semantic and syntactic prunings, however evaluation results suggest that we should strengthen this filtering. In addition, NALDO employs a general paraphrase database; we intend to make use of domain resources for future works. Another issue needs a closed look: ”Strong” entity linking where the formalization approach is able to identify suitable entities in the domain ontology, but which are not lexically found in the definition. Finally, NALDO is being adapted to deal with sentences out of the scope of definitions such as business rules and policies.

References

[1] V. Kashyap, A. Sheth, Semantic heterogeneity in global information systems: The role of metadata, context and ontologies, Cooperative information systems: Current trends and directions (1998) 139–178.

[2] I. Boukhari, L. Bellatreche, S. Jean, An ontological pivot model to interoperate heterogeneous user requirements, in: International Symposium On Leveraging Applications of Formal Methods, Verification and Validation, Springer, 2012, pp. 344–358.

[3] T. Berners-Lee, J. Hendler, O. Lassila, The semantic web, Scientific american 284 (5) (2001) 28–37.

[4] P. Mika, Ontologies are us: A unified model of social networks and semantics, Web semantics: science, services and agents on the World Wide Web 5 (1) (2007) 5–15.

[5] T. Wächter, M. Schroeder, Semi-automated ontology generation within obo-edit, Bioinformatics 26 (12) (2010) i88–i96.

[6] G. Tsatsaronis, A. Petrova, M. Kissa, Y. Ma, F. Distel, F. Baader, M. Schroeder, Learning formal definitions for biomedical concepts, in: OWLED, 2013.

[7] L. Bühmann, D. Fleischhacker, J. Lehmann, A. Melo, J. Völker, Inductive Lexical Learning of Class Expressions, in: 19th EKAW, Springer, 2014, pp. 42–53.

[8] J. Lehmann, S. Auer, L. Bühmann, S. Tramp, Class expression learning for ontology engineering, Web Semantics: Science, Services and Agents on the World Wide Web 9 (1) (2011) 71–81.

[9] J. Völker, P. Hitzler, P. Cimiano, Acquisition of OWL DL axioms from lexical resources, in: ESWC, Springer, 2007, pp. 670–685.

[10] R. R. de Azevedo, F. Freitas, R. Rocha, J. A. A. de Menezes, C. M. de Oliveira Rodrigues, M. C. Gomes, Representing knowledge in DL ALC from text, Procedia Computer Science 35 (2014) 176–185.

[11] T. Louge, M. H. Karray, B. Archimède, Investigating a method for automatic construction and population of ontologies for services: Performances and limitations, in: 2018 IEEE/ACS 15th International Conference on Computer Systems and Applications (AICCSA), IEEE, 2018, pp. 1–6.

[12] L. Bühmann, J. Lehmann, P. Westphal, S. Bin, Dl-learner structured machine learning on semantic web data, in: Companion Proceedings of the The Web Conference 2018, WWW '18, International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, Switzerland, 2018, pp. 467–471. doi:10.1145/3184558.3186235.

[13] T. Kuhn, Controlled English for knowledge representation, Ph.D. thesis, Faculty of Economics, Business Administration and Information Technology of the University of Zurich (2010).

[14] L. Del Corro, R. Gemulla, Clausie: clause-based open information extraction, in: Proceedings of the 22nd WWW, 2013, pp. 355–366.

[15] H. Bast, E. Haussmann, Open Information Extraction via Contextual Sentence Decomposition, in: 7th ICSC, IEEE Computer Society, 2013, pp. 154–159.

[16] E. Pavlick, J. Bos, M. Nissim, C. Beller, B. V. Durme, C. Callison-Burch, Adding semantics to data-driven paraphrasing, in: 53rd ACL, 2015, pp. 1512–1522.

[17] V. T. Nguyen, C. Sallaberry, M. Gaio, Mesure de la similarité entre termes et labels de concepts ontologiques, in: Conférence en recherche d'information et applications, 2013, pp. 415–430.

[18] M. Hadjieleftheriou, D. Srivastava, Weighted set-based string similarity., IEEE Data Eng. Bull. 33 (1) (2010) 25–36.

[19] A. Singhal, et al., Modern information retrieval: A brief overview, IEEE Data Eng. Bull. 24 (4) (2001) 35–43.

[20] C. K. Emani, C. Ferreira Da Silva, B. Fiès, P. Ghodous, Improving Open Information Extraction for Semantic Web Tasks, TCCI.
URL `https://hal.archives-ouvertes.fr/hal-01229542`

[21] C. K. Emani, C. Ferreira Da Silva, B. Fiès, P. Ghodous, BEAUFORD: A Benchmark for Evaluation of Formalisation of Definitions in OWL, OJSW 2 (1) (2015) 3–14.