Research online Publishing
www.ronpub.com

Open Access

Open Journal of Semantic Web (OJSW)
Volume 2, Issue 1, 2015
http://www.ronpub.com/ojsw
ISSN 2199-336X

BEAUFORD: A Benchmark for Evaluation of Formalisation of Definitions in OWL

Cheikh Kacfah Emani [A B], Catarina Ferreira Da Silva [B], Bruno Fiès [A], Parisa Ghodous [B]

[A] Centre Scientifique et Technique du Bâtiment, 290 route des Lucioles,
BP 209, 06904 Sophia Antipolis, France, {cheikh.kacfah, bruno.fies}@cstb.fr

[B] Université Lyon 1, LIRIS, CNRS, UMR5205, F-69622, France,
{cheikh.kacfah-emani, catarina.ferreira, parisa.ghodous}@univ-lyon1.fr

Abstract

In this paper we present BEAUFORD, a benchmark for methods which aim to provide formal expressions of concepts using the natural language (NL) definition of these concepts. Adding formal expressions of concepts to a given ontology allows reasoners to infer more useful pieces of information or to detect inconsistencies in this given ontology. To the best of our knowledge, BEAUFORD is the first benchmark to tackle this ontology enrichment problem. BEAUFORD allows the breaking down of a given formalisation approach by identifying its key features. In addition, BEAUFORD provides strong mechanisms to evaluate efficiently an approach even in case of ambiguity which is a major challenge in formalisation of NL resources. Indeed, BEAUFORD takes into account the fact that a given NL phrase can be formalised in many ways. Hence, it proposes a suitable specification to represent these multiple formalisations. Taking advantage of this specification, BEAUFORD redefines classical precision and recall and introduces other metrics to take into account the fact that there is not only one unique way to formalise a definition. Finally, BEAUFORD comprises a well-suited dataset to concretely judge of the efficiency of methods of formalisation. Using BEAUFORD, current approaches of formalisation of definitions can be compared accurately using a suitable gold standard.

Type of Paper and Keywords

Short communication: *Benchmark, automatic formalisation, natural language definitions*

1 Introduction

One of the most important ideas of Semantic Web technologies is the use of ontologies as background knowledge for information processing. The richness of the Web Ontology Language (OWL) enables to capture efficiently this knowledge. Ever since the invention of OWL, the degree of formalisation of web ontologies arouses discussions. This situation is due to the fact that complex formal ontologies are suitable for powerful reasoning (consistency checking and subsumption detection). But building an ontology having high logical specifications is a costly task which is, in addition, error-prone. To avoid modelling errors requires to have a deep understanding of the domain to model and also of the logic used to formalise it [10]. This problem is often mentioned as a main argument against the adoption of rich formal ontologies. To escape this problem, methods for (semi-)automatic knowledge acquisition from texts have been proposed. So far however, methods for building ontologies from texts, produce lightweight ontologies. Such ontologies mainly consist of a taxonomy and concepts and some relations between them [6]. But, over time, techniques which provide more expressive axioms

for ontology enrichment have emerged [9] [5] [14] [8]. Some of these techniques use common characteristics of instances of a given concept to figure out its expression [9] [5]. For example, by observing that all the instances of the concept `Pizza` are instances of the concept `Food` and also have a relation `hasTopping` with an individual of the knowledge base, we may conclude that (using Description Logics Notation - section 2)

$$\texttt{Pizza} \sqsubseteq \texttt{Food} \sqcap \exists \texttt{hasTopping}.\top \quad (1)$$

The other group of techniques provide expression of concepts using their natural language (NL) definition [14] [8]. For example, the expression presented by equation 1 above, can be obtained through the processing of the NL definition *"A pizza is a food always having a topping"*. Techniques of this group do not make any assumption (for example, having a minimal set of instances in the ontology) on the ontology to be enriched. Moreover, they only need a set of NL definitions to perform an automatic enrichment. In practice, such definitions already exist. Indeed, we can obtain definitions from Wikipedia pages, from dictionaries or sometimes from lightweight ontologies (for instance, the Open Biological and Biomedical Ontologies[1]). Unfortunately, to the best of our knowledge, there is not any standard experimental set-up to evaluate such approaches. This is why we propose BEAUFORD, the first benchmark able to evaluate efficiently approaches which formalise NL definitions.

BEAUFORD aims to break down approaches of formalisation of definitions. To achieve this goal, BEAUFORD first proposes a set of features which allow to clearly see what an approach is able to do and how the approach concretely works. In addition, BEAUFORD provides strong mechanisms to handle ambiguity, i.e. multiple formalisations of the same NL phrase. For instance, BEAUFORD amends the traditional definition of precision and recall to take into account ambiguity inherent to formalisation. Finally, BEAUFORD provides a data set made of real NL definitions which may be used to enrich existing domain ontologies.

The remainder of this paper is organised as follows: first, a brief description of Description Logics (OWL) and OWL (Section 2). Next, we present in details the task of formalisation of definitions in OWL (Section 3). Next, a brief state of the art (Section 4) introduces the presentation of the benchmark itself (Section 5). Finally, we see how in practice, BEAUFORD can be used to evaluate existing approaches of formalisation (Section 6).

[1] http://www.obofoundry.org/

2 DESCRIPTION LOGICS, OWL

Description Logics (DL) are a family of knowledge representation formalisms. They emerged from earlier representation formalisms like semantic networks and frames. Their origin lies in the work of Brachmann on structured inheritance networks [4]. Since then, DL have increased in popularity. DL are more expressive than propositional logic and can be understood as fragments of first-order logic. In DL the focus is on reasoning. The main goal is to have an interesting trade-off between expressiveness of a given DL and its complexity in time and space. More over their syntax is easily human-readable.

DL allow representing domain knowledge using:

- *concepts* (a.k.a. classes) of the domain (e.g. `Pizza`, `Country`, etc.)
- *relations* (a.k.a. roles) which can exist between concepts or between *instances* of these concepts (e.g. `hasTopping`, `hasCountryOfOrigin`, etc.)
- *instances* (a.k.a. individuals or objects) that are members of concepts in the application domain (e.g. `myPizza`, `England`, etc.)

To present semantics of DL, we will use the following notation:

- A and B are atomic concepts
- $\top$ is a predefined concept with every individual as an instance
- C and D are complex concepts
- R and S are roles/relations
- a, $a_{i,1\leq i\leq n}$ are individuals

2.1 $\mathcal{ALC}$

$\mathcal{ALC}$ (*Attributive Language Complement*) is a centrally important DL from which comparisons with other varieties can be made. $\mathcal{ALC}$ enables to construct complex concepts from simpler ones using the following language constructs:

- Concept negation: $\neg C$
- Concept intersection: $C \sqcap D$
- Universal restriction: $\forall R.C$
- Concept union: $C \sqcup D$
- Existential quantification: $\exists R.C$

The Web Ontology Language (OWL) is being recommended as a web standard by the World Wide Web Consortium for modelling ontological knowledge. The most important variant of OWL, called OWL DL, is a so-called description logic, or DL for short [3], is based on the DL known as $\mathcal{SHOIN}(D)$. We will introduce it below.

2.2 $\mathcal{SHOIN}(D)$

The constructor $\mathcal{S}$ in $\mathcal{SHOIN}$ stands for $\mathcal{ALC}$ with the addition of *role transitivity* ($Tr(R)$). The three other constructs are:

- $\mathcal{H}$: role hierarchy ($R \sqsubseteq S$)
- $\mathcal{O}$: nominals (enumerated classes of object value restrictions $\{a\}$ or $\{a_1, ..., a_n\}$)
- $\mathcal{I}$: role inverse (R^-)
- $\mathcal{N}$: (unqualified) number restrictions ($\leq nR.\top$, $\geq nR.\top$)

The symbol (D) which follows the "name" of the DL family $\mathcal{SHOIN}$ indicates that the use of data type properties (e.g. `hasCalorificContentValue`), data values or data types (e.g. integer, real, string,etc.) is allowed.

We notice that $\mathcal{SHOIN}(D)$ does not allow qualified number restrictions. For example, lets us consider the definition $\mathcal{D}_2$ = *"A siciliana pizza has at least 3 vegetable toppings"*. We expect $\mathcal{D}_2$ to be formalised as

$$\texttt{SicilianaPizza} \sqsubseteq \geq 3\, \texttt{hasTopping.VegetableTopping}.$$

But in $\mathcal{SHOIN}(D)$, $\mathcal{D}_2$ is formalised as

$$\texttt{SicilianaPizza} \sqsubseteq \geq 3\, \texttt{hasTopping}.\top$$

We see that in this formalisation we cannot state the range of property `hasTopping`, i.e. force all the toppings to be a kind of `VegetableTopping` as stated by de definition. Since we find qualified number restrictions in many real definitions we use here the DL fragment known as $\mathcal{SHOIQ}(D)$ to express formally the NL definitions in this benchmark. $\mathcal{Q}$ denotes the qualified number restrictions ($\leq nR.C$, $\geq nR.C$).

To conclude with this brief survey on DL, let us mention that there exists a fragment more expressive than $\mathcal{SHOIQ}(D)$. This fragment is denoted $\mathcal{SROIQ}(D)$ where $\mathcal{R}$ denotes complex roles inclusion i.e axioms of the form $R \circ S \sqsubseteq R$ and $S \circ R \sqsubseteq R$ [7]. $\mathcal{SROIQ}(D)$ supports OWL 2 DL [2]. Although the emphasis is on the $\mathcal{R}$, $\mathcal{SROIQ}(D)$ includes also *disjunction* and *negation* of roles, *reflexive* and *irreflexive* roles [7]. Enven if $\mathcal{SROIQ}(D)$ is decidable, we have limited the scope of this benchmark to $\mathcal{SHOIQ}(D)$, since we focus on definition of concepts.

3 Formalisation of Definitions

In this section we will define concretely the task of "formalisation of definitions".

Definition 3.1. A *definition* $\mathcal{D}$ is a NL sentence which allows the enunciation of the characteristics of a concept, of a word, of an object.

Definition 3.2. Formalize a natural language definition $\mathcal{D}$ w.r.t an ontology $\mathcal{O}$ is to propose *a formal expression*:

1. based on terms of $\mathcal{D}$
2. which subsumes or which is equivalent to *the concept* defined in $\mathcal{D}$
3. which uses foremost entities of $\mathcal{O}$.

From these two definitions, we note that:

- A definition $\mathcal{D}$ defines one and only one concept at a time. Most of real definitions (i.e. found in dictionaries and encyclopaedia) respect this criteria. For instance $\mathcal{D}$ cannot be a sentence like *"A **siciliana pizza** has at least 3 vegetable toppings and a **lujuhman pizza** is made with lamb, vegetables and feta cheese"*, which defines at the same time two NL terms (in bold).
- $\mathcal{D}$ must not contain facts that do not contribute to the characterisation of the defined concept. For instance the sentence *"I remember that lujuhman pizza contains lamb"* is not a definition. Indeed, the assertion *"I remember that"* does not give any piece of information on "Lujuhman pizza" itself.
- $\mathcal{D}$ is not necessarily of the form $< A,\ is\ a,\ B >$. The definition $\mathcal{D}_2$ is an actual example.
- The formal expression must avoid creating new entities when entities of $\mathcal{O}$ can be used.

Some others key characteristics of the formalisation tasks are worthy to be emphasized here. They concern *ambiguity* and *incomplete formalisation*.

Ambiguity: express formally a NL phrase implies to (i) understand what this phrase says exactly and (ii) to use the appropriate entities and constructors to formalise this sentence. In both of these steps, two domain experts and/or ontology designers can provide different results for the same inputs.

Incomplete (or partial) formalisation: in most cases, $\mathcal{D}$ mentions many pieces of information and using only

some of these pieces can nevertheless lead to a right and useful output. For instance from the definition $\mathcal{D}_3$ = *"A vegetarian pizza is any pizza which has vegetable toppings and no other toppings"*, one can provide the formal expression:

$$\texttt{VegetarianPizza} \sqsupseteq \texttt{Pizza} \sqcap \exists \texttt{hasTopping.VegetableToppings} \quad (*)$$

We see that this result uses only a part of the definition. Indeed, the phrase *"is any"* in the definition suggests an equivalence instead of a simple subsumption. Moreover, the expression *"no other toppings"* hints a universal restriction instead of an existential quantification. Formal expressions like $(*)$, which express only a part of the whole idea convey by the definition are so called *incomplete* or *partial* expression. Even if they are not "complete", they are not *semantically wrong* and are useful in practice.

Features of Formalisation Approaches

This subsection presents the features we deem necessary in a benchmark to break down an approach of formalisation.

Inputs. First of all, it is pivotal to know the input data required by the approach. Moreover, all the hypothesis and the requirements of the approach must be clearly identified.

Automation. Another important point is to know if the approach is automatic or not. If not, it is important to know which are the tasks devoted to user.

Schema Independence. The goal here is to know if the approach can work across schemas. In other words, we must figure out if the approach of formalisation is unsupervised or if the supervision can be performed only once. In the latter case, the classifier provided by this supervision can be re-used on different schemas.

Expressiveness. It is important to know how expressive are the expressions that the method provides. High logical specifications are the most useful for powerful inferences but it is very challenging to obtain them automatically from NL resources.

Handling of Ambiguity. When it is done manually, formalisation of a NL phrase can give many results depending on the domain expert. Hence, the goal here is to know if the method of formalisation provides mechanisms which deal with the multiplicity of interpretations.

Alignment with the given Ontology. In definition 3.2 we highlight the fact that a formalisation approach must re-use as possible entities already present in the ontology to be enriched. This feature expresses the ability of a given formalisation approach to find the most suitable entity in the domain's ontology that can represent a given phrase of a NL definition.

4 Related Work

As we mentioned in the introduction of this paper, to the best of our knowledge, BEAUFORD is the first benchmark for the evaluation of approaches of formalisation of NL definitions. It is thus difficult to compare it to benchmarks targeting the current formalisation task. Nevertheless, when we go through literature, we find experimental set-ups for tasks in both fields of NL processing and Semantic Web like Question Answering (QA) [1], Named Entity Recognition and Disambiguation (NERD) [11] and Acquisition of Class Disjointness [13].

Given a NL question, a QA system must be able to retrieve all the correct answers (*recall*), and only them (*precision*), from a knowledge base. Usually, when evaluating this QA systems, the set of correct answers is fixed. In other words, there is not any debate to say whether a given object is the answer (or belongs to the set of answers) to a question. We see that in this case the "classical" f1-measure can be used to efficiently compare performances of QA systems. The same conclusion holds for NERD systems. In NERD, the set of entities to be identified is well known in advance. Consequently, performances of a given NERD system can be evaluated in the light of its precision and recall. As we mentioned at the end of section 3, ambiguity is intimately linked to the task of formalisation of definition. Classical f1-measure is hence not suitable to evaluate formalisation approaches we are interested in.

In acquisition of class disjointness, ontology designers do not always agree to assert two classes as disjoint or not. Völker and colleagues [13] thus face cases of ambiguity in the set of correct answers. To handle ambiguities, they calculate f1-measure in two main cases: for the subset of the gold standard (i) where all the ontology designers agree and (ii) where at least 50% of the designers agree. When dealing with formalisation of definitions, we cannot always exclude some possible results. Indeed, two formal expressions written with two different set of entities can be, for example, equivalent. For the sake of consistency, we must provide mechanisms to judge these two formalisations as correct, without ex-

cluding any of them. Hence, it is pivotal to redefine the classical formula of precision and recall, to be able to handle multiple formalisations of the same definition.

We have presented the task of formalisation of definitions, its concerns and the challenges to solve w.r.t to the state of the art, we now present BEAUFORD.

5 Design of the BEAUFORD Benchmark

5.1 Scope of BEAUFORD

In this subsection, based on the features described in section 3, we delimit the scope of approaches that BEAUFORD is able to evaluate.
BEAUFORD deals with approaches which aim to provide formal expression of concepts from *NL definitions* (as stated in definition 3.1). In literature, there also exist approaches which learn concept expression by identifying common characteristics of the instances of this concept in a given knowledge base like [5, 9]. Such approaches are out of the scope of BEAUFORD. Moreover, approaches we are interested in are expected to be as *automatic* as possible. They are expected to run fluently for a pair $\langle$definition, domain ontology$\rangle$. The expressiveness expected of formal expression is $\mathcal{SHOIQ}(D)$.

In addition, BEAUFORD takes into account the fact that the formal translation of a NL definition does not always provide a single result. Mechanisms to handle this key aspect of formalisation are detailed in subsections 5.2 and 5.4.

5.2 Ambiguity and Alignments

The definition 3.2 requires that an approach of formalisation uses foremost entities of the existing ontology $\mathcal{O}$. Nonetheless, when trying to align a NL phrase which does not match exactly an entity in $\mathcal{O}$, we have to choose between two main possibilities. We name them *sharp* and *large* formalisations. We define them right below.

Definition 5.1. A sharp formalisation is a formalisation which intends to always find, for a given NL phrase, the most likely entity in the domain ontology $\mathcal{O}$ to formalise this NL phrase. Hence, proposal of new entities is done when sure that there is not any entity in $\mathcal{O}$ that can directly represent that new entity.

For instance, a formal expression of the definition $\mathcal{D}_3$ = *"Luhjuhman pizza is a pizza made with lamb, vegetables, and feta cheese"* in the sharp formalisation mind, *should* use the existing concepts (in the pizza ontology) `LujhmanPizza`, `Pizza`, `VegetableTopping` and `CheeseTopping` (or `GoatCheeseTopping` instead). Moreover, as there is not any entity in the pizza ontology able to represent directly the phrase "lamb", a new entity, for example `Lamb`*, can be propose. In this example, using Web redundancy, the new concept `Lamb`* can be replaced by `MeatTopping` because *lamb is a (kind of) meat*. But to map the NL phrase "lamb" directly to the entity `MeatTopping` is very challenging and goes beyond classic string matching algorithms. We do not actually consider it as a valid example of sharp alignment.

Note In this paper, we use * to tag every new entity, i.e. which is not (already) part of the domain's ontology $\mathcal{O}$.

Definition 5.2. Large formalisation is a formalisation where trend is the creation of new entities. Nevertheless, priority remains on the re-use of entities of the domain's ontology $\mathcal{O}$.

For example, for the definition $\mathcal{D}_3$ given above, a possible result *should* use the existing entity `LujuhmanPizza` (exact string matching). Because, trend is the creation of new entities, the formal expression *could* use the new entities `Vegetable`*, `FetaCheese`* (or the intersection of $\{$`Feta`* and `Cheese`$^*\}$) and `Lamb`*.

Having these two most distant types of formalisation for each phrase of the definition allows to cover a wide space of possible formalisations of the whole definition. Indeed, depending on how a formalisation method proceeds, it may provide a formal expression with a mix of large and sharp alignments. For instance $\mathcal{D}_3$ can be formalised by

`LujuhmanPizza` $\sqsubseteq$ `Pizza` $\sqcap$ ($\exists$ `isMadeWith`*.`Lamb`*
$\sqcap \exists$ `hasTopping`.`VegetableTopping`
$\sqcap \exists$ `isMadeWith`*.`FetaCheese`*)

where the concept `VegetableTopping` results from a sharp linking and `FetaCheese`* from a large one.

Although entities resulting of sharp and large alignments can coexist in the same formal expression, there is an important need of *consistency*. Indeed, we consider that if a method is able to perform a challenging alignment then it must be able to perform a common sense alignment. For instance, using $\mathcal{D}_3$, we consider that obtaining that "feta cheese" can be formalised by the entity `GoatCheeseTopping` is far more challenging than obtaining that "vegetables" can be formalised using `VegetableTopping`. Hence, achieving the former alignment must lead to an identification of the latter. Likewise, a formalisation approach not able to map "vegetables" to `VegetableTopping` must not be able to map "feta cheese" to `GoatCheeseTopping`.

Another issue introduced by the use of new entities affects the range of the properties. Indeed, we cannot always assert that an existing relation can be filled by a new concept. For instance, w.r.t. the pizza ontology, it is unwanted to write something like `hasTopping.Lamb*`. That's because the range of the relation `hasTopping` is `PizzaTopping` and it is unproven yet that `Lamb*` is a sub-concept of `PizzaTopping`.
All the issues about the coexistence of new and existing entities in the same formal expression are presented below. We introduce them with what we call *incompatible entities*.

Definition 5.3. Let $\mathcal{D}$ be a NL definition, A and B two concepts (or individuals) and R a role. If A, B and R can be used in a formal expression of $\mathcal{D}$, then the following cases of incompatibility holds:

- *A is incompatible with B* if when an approach identifies A, this approach must not be able to identify B. This incompatibility happens when it is more challenging to identify A than B or vice versa. For instance, for the definition $\mathcal{D}_3$, `GoatCheeseTopping` is incompatible with `Vegetable`.

- *A is incompatible with R* if A (or its type - when A is an individual) cannot stand as a filler of the role R. For instance, still using $\mathcal{D}_3$, `hasTopping` is incompatible with `Lamb*`.

The relation of incompatibility thus defined is *symmetric*.

Let us mention that results of sharp and large formalisations sometimes meet. The first meeting point is in the case of an exact string matching (for example "Lujuhman pizza" cannot be mapped to anything else than `LujuhmanPizza`). The second meeting point is in case of the non existence of a concept (at the current state of $\mathcal{O}$). Indeed, when $\mathcal{O}$ lacks a concept, described by a NL phrase mentioned in $\mathcal{D}$, there is no choice than the creation of a new concept. An illustration of this case is given by the phrase "lamb" within $\mathcal{D}_3$.

5.3 Specification of a Formalisation Result

BEAUFORD encapsulates all the pieces required for the formalisation of a definition within a given schema. This schema is available through a XML Schema Definition (XSD) file accessible at `http://tinyurl.com/opllwpt`. To represent the result of the formalisation of a given definition, we need three elements:

1. The list of all possible formal entities required to formalise this definition

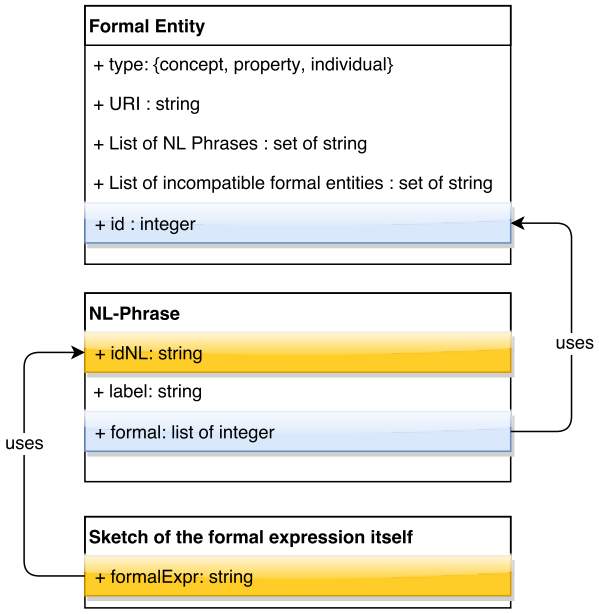


Figure 1: A schematic view of the specifications of BEAUFORD.

2. The list of all NL phrases of the definition that we need to formalise in other to have the full formal expression. Among other pieces of information, we have to tell which formal entities can be used to formalise a given NL phrase.

3. Finally, we have to provide the *sketch* of the formalisation result. This sketch reuses the identifiers of NL-phrases with the required DL constructors to show how the final formal expression should look like.

A simplified view of the interaction of formal entities, NL-phrases and the sketch of formalisation is illustrated by Figure 1.
We now present the semantics of each node of this XSD specification through a running example.

5.3.1 Entities

For each definition, BEAUFORD requires the list of *all possible entities* (existing entities and possible new ones) that could be used to formalise this definition. In practice, for a precise definition, this list comprises the union of all the entities used by ontology designers to formalise this definition.
Each of these entities respect the following description:

- the *type*, i.e. concept, predicate or individual, of the entity (<*type*>)

- a *shortened URI* (<*uri*>)

- the list of *NL-phrases* of $\mathcal{D}$ that this entity formalises (<*nlEntity*>)
- an identifier (<*id*>) which is a number used to represent this entity where needed
- the list of identifiers of entities incompatible with this entity (<*incompEnt*>)

Some examples of entities are provided by listing 1.

Listing 1: An example of entities that can be used to formalise $\mathcal{D}_3$

```
<entity>
 <type>concept</type>
 <uri>Vegetable*</uri>
 <nlEntity>vegetables</nlEntity>
 <id>1</id>
</entity>
<entity>
 <type>concept</type>
 <uri>VegetableTopping</uri>
 <nlEntity>vegetabkes</nlEntity>
 <id>2</id>
</entity>
<entity>
 <type>concept</type>
 <uri>Feta*</uri>
 <nlEntity>feta</nlEntity>
 <id>3</id>
</entity>
<entity>
 <type>concept</type>
 <uri>Cheese*</uri>
 <nlEntity>cheese</nlEntity>
 <id>4</id>
</entity>
<entity>
 <type>concept</type>
 <uri>Lamb*</uri>
 <nlEntity>lamb</nlEntity>
 <id>5</id>
</entity>
...
<entity>
 <type>concept</type>
 <uri>:GoatCheeseTopping</uri>
 <nlEntity>feta</nlEntity>
 <nlEntity>cheese</nlEntity>
 <id>7</id>
 <incompEnt>1</incompEnt> <!-- 1=Vegetable* -->
</entity>
...
<entity>
 <type>property</type>
 <uri>hasTopping</uri>
 <nlEntity>made with</nlEntity>
 <id>10</id>
 <incompEnt>1</incompEnt>
 <incompEnt>5</incompEnt> <!-- 5=Lamb* -->
</entity>
```

5.3.2 Natural Language Phrases to Formalise

Formalising a NL definition requires formalising all the valuable NL phrases of this definition (for instance, stop words are usually useless in formalisation approaches). In this section of the specification, BEAUFORD structures this list of valuable NL-phrases. Formalisations of some phrases of $\mathcal{D}_3$ are shown by listing 2. This example re-uses entities of listing 1.

Listing 2: An example of formalisations of some NL phrases taken from $\mathcal{D}_3$

```
<nlPhrase>
 <idNL>VegEnt</idNL><label>vegetables</label>
 <formal>1</formal> <!-- 1=Vegetable*-->
 <formal>2</formal> <!-- 2=VegetableTopping-->
 <!-- "vegetables" can be formalised by the
  entity with the id 1 or entity with the id 2-->
</nlPhrase>
<nlPhrase>
 <idNL>FetaCheNL</idNL>
 <label>feta cheese</label>
 <formal>7</formal><formal>3 ⊓ 4</formal>
 <!-- We can represent formally
   "feta cheese" by the entity
  7 (GoatCheeseTopping) or the intersection
  of entities 3 (Feta*) and 4 (Cheese*) -->
 ...
</nlPhrase>
<nlPhrase>
 <idNL>LujEnt</idNL>
 <label>lujuhman pizza</label>
 ...
</nlPhrase>
<nlPhrase>
 <idNL>PizzaEnt</idNL><label>pizza</label>
 ...
</nlPhrase>
<nlPhrase>
 <idNL>LambEnt</idNL><label>lamb</label>
 ...
</nlPhrase>
<nlPhrase>
 <idNL>madeEnt</idNL><label>made with</label>
 <formal>10</formal>
 ...
</nlPhrase>
```

From the listing 2, we see that each NL phrase (<*nlPhrase*>) is made of:

- an identifier (<*idNL*>) which will be used in the formal expression of the $\mathcal{D}$ to denote the formal expression of this phrase
- a label (<*label*>) which is constituted of NL-phrases with a known formalisation (given in <*entity*> - section 5.3.1)
- an exhaustive list of its possible formalisation (<*formal*>). With this list, BEAUFORD is aware that multiple formalisations of the same phrase (and thus of the same definition) are possible.

5.3.3 Sketch of a Formal Expression

Finally for each definition BEAUFORD provides the exact *sketch* of the formal expression made of DL con-

structors and identifiers of NL phrases (signalled by the tag <*idNL*> - section 5.3.2) of this definition. Listing 3 gives the sketch of the formal expression of $\mathcal{D}_3$.

Listing 3: Sketch of the formal expression of $\mathcal{D}_3$

```
<formalExpr>
 LujEnt ⊒ PizzaEnt ⊓ (∃ madeEnt.LambEnt
  ⊓ ∃ madeEnt.VegEnt ⊓ ∃ madeEnt.FetaCheEnt)
</formalExpr>
```

The formal expression of a given definition has *a unique sketch*. This sketch allows BEAUFORD to cover all the possible valid formal expressions (see definition 5.4) of a definition and thus to efficiently deal with ambiguity. Indeed, the sketch uses identifiers of NL-phrases to be formalised. An identifier of a NL-phrase is a slot which can be filled by any possible formalisation of this NL-phrase. For instance, we can instantiate the slot `vegEnt` in listing 3, by the formal entities 1 (i.e. `Vegetable`$\star$) or 2 (i.e. `VegetableTopping`).

The full example showing the specification of the formalisation of definition $\mathcal{D}_3$ is available at `http://tinyurl.com/nmfluek`.

Remark. It is important to underline that the identifier of a NL-phrase may appear more than once in a given sketch. In such cases, this identifier can be filled with different formal entities. For example, the NL-phrase `madeEnt` which represents the phrase "made with" can be formalised in the *same expression* by the role `madeWith`* (to be filled with `Lamb`*) and the role `hasTopping` (to be filled with `VegetableTopping`).

Definition 5.4. A valid formal expression of a given definition must be a consistent instantiation of the sketch. Consistent here means that it is not permitted to have two incompatible entities in the formal expression.

For instance:

1. The following instantiation of the sketch in listing 3 is valid because it does not contain any inconsistency
   `LujuhmanPizza` $\sqsubseteq$ `Pizza` $\sqcap$ ($\exists$`madeWith`*.`Lamb`* $\sqcap \exists$ `hasTopping`.`VegetableTopping` $\sqcap \exists$ `madeWith`*.`FetaCheese`*)

2. But in this second case the formalisation is incorrect because it contains at the same time `Vegetable`* and `GoatCheeseTopping` which are considered incompatible
   `LujuhmanPizza` $\sqsubseteq$ `Pizza` $\sqcap$ ($\exists$`madeWith`*.`Lamb`* $\sqcap \exists$ `madeWith`*.`Vegetable`* $\sqcap \exists$ `madeWith`*.`GoatCheeseTopping`*)

5.4 Metrics: Precision, Recall and Confidence

At section 5.3, we presented mechanisms that BEAUFORD proposes to handle all the possible results of the formalisation of a definition. Now we present the metrics to evaluate formalisation approaches by taking into account the multiplicity of formal expressions for a given definition.

To calculate the various metrics of a given approach, we assume that for each definition, the expected results of formalisation (gold standard) are available and follow the specification of section 5.3. Hence, for each NL definition we have:

- E_a the set of entities (without duplicates) provided by the given approach
- E_b the set of entities, using the tag <*uri*> of each <*entity*>, (without duplicates) suggested in the gold standard
- I_a the set of pairs of incompatible entities (as mentioned in the gold standard) found in E_a
- f_a, the formal expression actually provided by the approach

In addition, we define:

- the boolean function $instance(f, \mathcal{D})$ which returns 1 is f is a valid formalisation (as mentioned in definition 5.4) of the sketch of the formal expression of $\mathcal{D}$ and 0 otherwise.
- the function $nlPhrases(X)$ which returns the set (without duplicates) of NL-phrases actually formalised by formal entities which constitute the set X

With these elements, we use the equations 2-4 below to compute *precision*, *recall* and *confidence* for a set of definitions $\{\mathcal{D}\}$.

$$precision = \frac{\Sigma_{\{\mathcal{D}\}}\left(|E_a \cap E_b| - |I_a|\right)}{\Sigma_{\{\mathcal{D}\}}|E_a|} \tag{2}$$

$$recall = \frac{\Sigma_{\{\mathcal{D}\}}|nlPhrases\left(E_a \cap E_b\right)|}{\Sigma_{\{\mathcal{D}\}}|nlPhrases\left(E_b\right)|} \tag{3}$$

$$confidence = \frac{\Sigma_{\{\mathcal{D}\}}|instance(f_a, \mathcal{D})|}{|\{\mathcal{D}\}|} \tag{4}$$

Let us note that:

- Precision is the ratio between (i) the number of formal entities correctly identified ($|E_a \cap E_b|$) by the approach and penalizes in presence of incompatible entities (by means of the quantity $|I_a|$) and (ii) the total number of entities identified by the approach.

- The recall is the ratio of NL-phrases which are correctly formalised by the approach. BEAUFORD does not take the percentage of correct formal entities (i.e $|E_a \cap E_b| \div |E_b|$) because an entity can formalise more than one NL-phrase. For example `:GoatCheeseTopping` formalises both "feta" and "cheese". BEAUFORD considers that recall in a formalisation approach, measures the ability of this approach to formalise accurately all the meaningful NL-phrases of definitions.
- Confidence as defined by equation 4 denotes the percentage of formal expressions which are in all points accurate as stated in definition 5.4.

Some Examples

We take the NL definition $\mathcal{D}_3$ = *"Luhjuhman pizza is a pizza made with lamb, vegetables, and feta cheese"* for this illustration. The expected result of the formalisation of $\mathcal{D}_3$ is described in the file `http://tinyurl.com/nmfluek`.

Based on this expected result, we have:

- Eb = Set of entities suggested by the gold standard = {`LujuhmanPizza`, `Pizza`, `madeWith`*, `hasTopping`, `Lamb`*, `Vegetable`*, `VegetableTopping`, `Feta`*, `Cheese`*, `CheeseTopping`, `GoatCheeseTopping`, `FetaCheese`*}
- Set of pairs of incompatible properties suggested by the gold standard = {{`Vegetable`*, `GoatCheeseTopping`}, {`Vegetable`*, `CheeseTopping`}, {`Lamb`*, `hasTopping`}, {`Vegetable`*, `hasTopping`}}
- $nlPhrases(Eb)$ = Set of NL-phrases to formalise = {Lujuhman pizza, pizza, made with, vegetables, lamb, feta, cheese}

Result 1

- $Formalisation_1$ = `LujuhmanPizza` $\sqsubseteq$ `Pizza` $\sqcap$ ($\exists$ `madeWith`*.`Lamb`* $\sqcap \exists$`madeWith`*.`Vegetable`* $\sqcap \exists$`madeWith`*.`GoatCheeseTopping`)
- Ea = Set of entities suggested by the approach = {`LujuhmanPizza`, `Pizza`, `madeWith`*, `Lamb`*, `Vegetable`*, `GoatCheeseTopping`}
- Ia = pairs of incompatible entities in Ea = `Vegetable`*, `GoatCheeseTopping`
- Precision $= \dfrac{|Ea \cap Eb| - |Ia|}{|Ea|} = \dfrac{6-1}{6} = 0.83$
- Recall $= \dfrac{|nlPhrases(Ea \cap Eb)|}{|nlPhrases(Eb)|} = \dfrac{7}{7} = 1.0$
- Confidence $= 0$. Since there are incompatible entities the formal expression is incorrect

Result 2

- $Formalisation_2$ = `LujuhmanPizza` $\sqsubseteq$ `Pizza` $\sqcap$ ($\exists$ `madeWith`*.`Lamb`* $\sqcap \exists$`hasTopping`.`VegetableTopping` $\sqcap \exists$`madeWith`*.`FetaCheese`*)
- Ea = {`LujuhmanPizza`, `Pizza`, `madeWith`*, `hasTopping`, `Lamb`*, `Vegetable`*, `FetaCheese`*}
- Ia = pairs of incompatible entities in $Ea = \{\}$
- Precision $= \dfrac{|Ea \cap Eb| - |Ia|}{|Ea|} = \dfrac{7-0}{7} = 1.0$
- Recall $= \dfrac{|nlPhrases(Ea \cap Eb)|}{|nlPhrases(Eb)|} = \dfrac{7}{7} = 1.0$
- Confidence $= 1$. No incompatible entities and all the DL constructors are correctly used

BEAUFORD also provides a seed data set to evaluate the efficiency of formalisation approaches. This data set is made of NL definitions (as defined earlier in this paper) and domain ontologies to be enriched by formalising these definitions. Moreover, BEAUFORD provides formal expressions resulting of large and sharp formalisations. The data set is presented in the next section.

5.5 Data Set

To evaluate concretely an approach based on the characteristics listed above, BEAUFORD provides a data set. This data set is made of 3 subsets each composed of a domain ontology and 25 real definitions related to each ontology. All these data and their description are available at http://tinyurl.com/pkx9hz5.

The domain ontologies used in BEAUFORD are:

1. The well-known Pizza Ontology (`PIZZA`)
2. The Vertebrate Skeletal Anatomy (`VSAO`) Ontology[2]
3. The Semantic Sensor Network (`SSN`) Ontology[3]

These three ontologies put together have the following advantages:

[2] http://purl.obolibrary.org/obo/vsao.owl
[3] http://www.w3.org/2005/Incubator/ssn/ssnx/ssn

- *Existing ontologies.* These ontologies already exist and are commonly used. This avoids us to build (unreal) ontologies from scratch to evaluate the current task we are interested in.

- *Existing set of official definitions.* To evaluate a formalisation method towards each of these ontologies, we do not absolutely need to *create* definitions. Indeed, official versions of these ontologies already provide definitions. In VSAO, definitions are asserted by means of the `obo:IAO_000115`[4] property. In SSN, definitions can be found in the `rdfs:comment` annotation property. Concerning `PIZZA`, some definitions are also already provided (mainly through `rdfs:comment` annotation). But to have a substantial set of definitions, we add more definitions from various resources (Wikipedia and some restaurants sites[5]).

- *Variety of domains.* Each of these ontologies covers a specific domain. It allows us to see how domain-independent is the formalisation approach. Moreover, because they have been designed by different group of people, design choices (conceptual schema, level of details on labels of entities, etc.) are different. For instance, concepts in `VSAO` have an average number of 2.08 tokens meanwhile the same quantity in `PIZZA` and `SSN` is respectively 1.66 and 1.49. The fact that entities have labels made of many tokens is very important in practice. Indeed, if tokenizers in natural language processing are able to identify tokens (usually words), it is a challenging task to know which tokens (must) go together to form an entity w.r.t to a given domain. For instance the phrase "X is a tomato topping" can lead to the expression $\mathbf{X} \sqsubseteq$ **:TomatoTopping** (the two tokens "tomato" and "topping" are taken together) or to $\mathbf{X} \sqsubseteq$ (**new:Tomato** $\sqcap$ **:PizzaTopping**) (if "tomato" and "topping" are considered separately).

6 BEAUFORD IN PRACTICE

We claim that we can use BEAUFORD to evaluate efficiently approaches of formalisation of definitions. Strengths of BEAUFORD rely on:

- mechanisms to handle ambiguity, i.e. all the multiple expressions which can formalise a given definition. These mechanisms are the encapsulation of all the constituents of possible formalisations of a definition in the same place (section 5.3) and the redefinition of the classical formulas of precision and recall and the introduction of a new metric, the confidence, to take into account ambiguity.

- A suitable corpus covering various domains and with formalised definitions.

6.1 Using BEAUFORD Corpus in the state

BEAUFORD provides a corpus which can be used to evaluate, without any other need, formalisation methods or tools. In this case, these tools have to formalise for a domain ontology (between `PIZZA`, `VSAO` and `SSN`), the corresponding NL definitions. Finally, likewise the above examples of formalisations and evaluations (section 5.4), the formal expressions of each definition provided by the tool can then be compared with the gold standard. Using the definitions proposed in this BEAUFORD dataset, authors of [8] have evaluated `NALDO` against the three domain ontologies `PIZZA`, `VSAO` and `SSN`. We do not actually report here the result of the evaluation of NALDO [8] using the BEAUFORD data set because the metrics used in [8] are different of those we define here.

6.2 Extending BEAUFORD Corpus

Since BEAUFORD provides the specification of results of a gold standard for this task, its current corpus can be freely and easily extended. Indeed, a corpus is made up of an existing domain ontology to be enriched, a set of NL definitions and the expected results of the formalisation of these definitions. The only constraint is that these results may respect the specifications detailed in section 5.3. For instance, Völker et al. [14] use the Proton Ontology [12], among other ontologies to evaluate their approach, `LEXO`. We can thus imagine that, the definitions used for a critical discussion of their work, can be formalised under the specification of BEAUFORD and thus can be re-used later to evaluate formalisation tools and then compare with performances of `LEXO`. It is important to notice that, actually there is not any metrics or values in [14] to describe the performances of `LEXO`.

7 CONCLUSION AND FUTURE WORK

In this paper, we present BEAUFORD, a benchmark suitable to evaluate methods of formalisation of NL definitions. The formalisation of a given definition to enrich an ontology can give different results based on the understanding and the choice of entities of this ontology to rewrite the definition. Handle all these possible cases is a major concern to efficiently evaluate formalisation. The main contribution of this benchmark is to

[4] The prefix `obo` stands for http://purl.obolibrary.org/obo/

[5] http://www.pizzaexpress.com/our-food/our-restaurant-menu/mains/, http://www.nutritionrank.com/

provide strong mechanisms to handle multiple formalisation results for a given definition. Indeed, BEAUFORD considers that a NL sentence can be split in many ways and that each chunk can be formalised in different ways. BEAUFORD defines formally all the parts which constitute a formalisation result using an XSD specification. This specification handles efficiently the multiple formalisation result. Moreover, based on the elements found within the specification of a formalisation result, BEAUFORD redefines precision and recall and introduces the notion of confidence to judge of the overall consistency of a formalisation approach. Despite BEAUFORD covers numerous key aspects required by formalisation, we can extend its abilities by handling *partial formalisation* and *chain of properties* which are also important concerns. Indeed, the whole formal expression obtained from a definition contains many subexpressions which are of great help for ontology enrichment. For instance from $\mathcal{D}_3$ we can extract expressions `LujuhmanPizza` $\sqsubseteq$ `Pizza` and `LujuhmanPizza` $\sqsubseteq \exists$ `hasTopping.VegetableTopping`. Moreover, if a formal entity can represent the sense of many NL-phrases (like `GoatCheeseTopping` with "feta" and "cheese" in $\mathcal{D}_3$), the reverse is also possible. For example, let us imagine that the pizza ontology is extended from Figure 2.A to 2.B. Hence, the NL-phrase "made with" in $\mathcal{D}_3$ could be formalised by the chain of properties `hasTopping` $\circ$ `hasIngredient`.

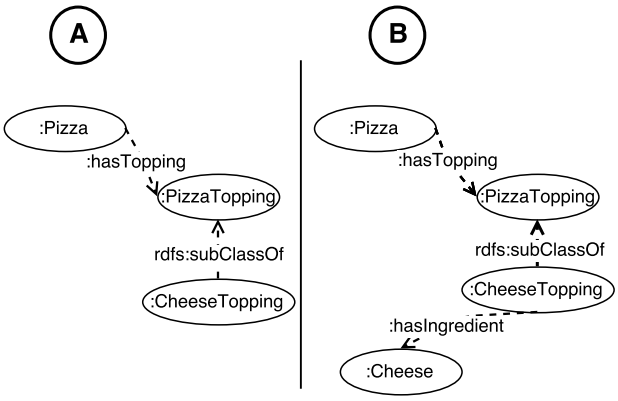


Figure 2: Two examples of RDF graphs of pizzas, toppings and ingredients. Ellipse shape nodes represent RDF resources and dashed arcs RDF properties.

REFERENCES

[1] CLEF Question Answering Track. [Online]. Available: http://nlp.uned.es/clef-qa/

[2] (2012, dec) OWL 2 Web Ontology Language New Features and Rationale (Second Edition). [Online]. Available: http://www.w3.org/TR/owl2-new-features/

[3] F. Baader, *The description logic handbook: theory, implementation, and applications*. Cambridge university press, 2003.

[4] R. J. Brachman, "A Structural Paradigm for Representing Knowledge." DTIC Document, Tech. Rep., 1978.

[5] L. Bühmann, D. Fleischhacker, J. Lehmann, A. Melo, and J. Völker, "Inductive lexical learning of class expressions," in *Knowledge Engineering and Knowledge Management*. Springer, 2014, pp. 42–53.

[6] P. Cimiano, A. Mdche, S. Staab, and J. Vlker, "Ontology learning," in *Handbook on Ontologies*, ser. International Handbooks on Information Systems, S. Staab and R. Studer, Eds. Springer Berlin Heidelberg, 2009, pp. 245–267.

[7] I. Horrocks, O. Kutz, and U. Sattler, "The Even More Irresistible $\mathcal{SROIQ}$," in *Proceedings of the 10th Int. Conf. on Principles of Knowledge Representation and Reasoning*, vol. 6. AAAI Press, 2006, pp. 57–67.

[8] C. Kacfah Emani, C. Ferreira Da Silva, B. Fiès, P. Ghodous, and A. Zarli, "NALDO: From natural language definitions to OWL DL expressions," 2015, Demo tool available at http://tinyurl.com/na4ecs3.

[9] J. Lehmann, S. Auer, L. Bühmann, and S. Tramp, "Class expression learning for ontology engineering," *Web Semantics: Science, Services and Agents on the World Wide Web*, vol. 9, no. 1, pp. 71–81, 2011.

[10] A. Rector, N. Drummond, M. Horridge, J. Rogers, H. Knublauch, R. Stevens, H. Wang, and C. Wroe, "OWL Pizzas: Practical Experience of Teaching OWL-DL: Common Errors and Common Patterns," in *Engineering Knowledge in the Age of the Semantic Web*, ser. Lecture Notes in Computer Science, E. Motta, N. Shadbolt, A. Stutt, and N. Gibbins, Eds. Springer Berlin Heidelberg, 2004, vol. 3257, pp. 63–81.

[11] G. Rizzo, M. van Erp, and R. Troncy, "Benchmarking the extraction and disambiguation of named entities on the semantic web," in *LREC*, 05 2014.

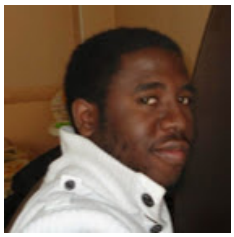
[12] I. Terziev, A. Kiryakov, and D. Manov, "D. 1.8. 1 Base upper-level ontology (BULO) Guidance," *Deliverable of EU-IST Project IST*, 2005.

[13] J. Völker, D. Fleischhacker, and H. Stuckenschmidt, "Automatic acquisition of class disjoint-

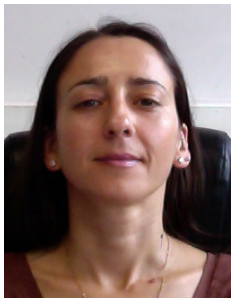
ness," *Web Semantics: Science, Services and Agents on the World Wide Web*, 2015.

[14] J. Völker, P. Hitzler, and P. Cimiano, "Acquisition of OWL DL axioms from lexical resources," in *The Semantic Web: Research and Applications*. Springer, 2007, pp. 670–685.

AUTHOR BIOGRAPHIES



Cheikh Kacfah Emani is currently PhD student in the IT department of the *Centre Scientifique et Technique du Bâtiment* in Sophia Antipolis and in computer science department of University of Lyon I. He started his PhD in November 2013. Previously, he obtained an engineering degree in 2011 in University of Yaounde I and a Master degree in 2013 in University of Burgundy in Dijon. His current research interests are Natural Language Processing, Business Rules Management, Semantic Web and Knowledge Representation.



Dr. Catarina Ferreira Da Silva is Associate Professor at the Computer Science Department of the University Institute of Technology of the Claude Bernard Lyon 1 University, and joined the Service Oriented Computing team of the Research Center for Images and Intelligent Information Systems (France) in 2012. She is also member of the Information System Group of the Centre for Informatics and Systems of the University of Coimbra (Portugal) since 2009. She obtained her PhD thesis in computer science (2007) from the University of Lyon 1. Previously she worked for the Scientific and Technical Centre for Building at Sophia-Antipolis (France). Her main current research interests are Knowledge Representation and Reasoing, Business Rules, Semantic Web, Service Science and Cloud Computing.

Dr. Bruno Fiès is a research engineer in the field of Information and Communication Technologies. He started his contribution with CSTB (French research center in Building and Construction) working on Electronic Data Interchange related to the Construction Sector. He naturally moved from information exchange to knowledge management. His main field of interest is now on Semantic web technologies applied to the Construction Sector with a specific focus on energy efficiency issues from the Building to the City scale. He has been and is still involved in National and European Research Projects.

Dr. Parisa Ghodous is currently full professor in computer science department of University of Lyon I. She is head of cloud computing theme of LIRIS UMR 5205 (Laboratory of Computer Graphics, Images and Information Systems). Her research expertise is in the following areas: Cloud Computing, Interoperability, Web semantic, Web services, Collaborative modeling, Product data exchange and modeling and Standards. She is in editorial boards of CERA, ICAE and IJAM journals and in the committees of many relevant international associations such as concurrent engineering, ISPE, Interoperability.