A Holistic Natural Language Generation Framework for the Semantic Web

Axel-Cyrille Ngonga Ngomo[1] **Diego Moussallem**[1] **Lorenz Bühmann**[2]
[1]Data Science Group, University of Paderborn, Germany
[2]AKSW Research Group, University of Leipzig, Germany
{first.lastname}@upb.de
lastname@informatik.uni-leipzig.de

Abstract

With the ever-growing generation of data for the Semantic Web comes an increasing demand for this data to be made available to non-semantic Web experts. One way of achieving this goal is to translate the languages of the Semantic Web into natural language. We present LD2NL, a framework for verbalizing the three key languages of the Semantic Web, i.e., RDF, OWL, and SPARQL. Our framework is based on a bottom-up approach to verbalization. We evaluated LD2NL in an open survey with 86 persons. Our results suggest that our framework can generate verbalizations that are close to natural languages and that can be easily understood by non-experts. Therewith, it enables non-domain experts to interpret Semantic Web data with more than 91% of the accuracy of domain experts.

1 Introduction

Natural Language Generation (NLG) is the process of automatically generating coherent Natural Language (NL) text from non-linguistic data (Reiter and Dale, 2000a). Recently, the field has seen an increased interest in the development of NLG systems focusing on verbalizing resources from Semantic Web (SW) data (Gardent et al., 2017). The SW aims to make information available on the Web easier to process for machines and humans. However, the languages underlying this vision, i.e., Resource Description Framework (RDF), SPARQL Query Language (SPARQL) and Web Ontology Language (OWL), are rather difficult to understand for non-expert users. For example, while the meaning of the OWL class expression `Class: Professor SubClassOf: worksAt SOME University` is obvious to every SW expert, this expression ("Every professor works at a university") is rather difficult to fathom for lay persons.

Previous works such as SPARQL2NL (Ngonga Ngomo et al., 2013) and SPARTIQULATION (Ell et al., 2012) have already shown the usefulness of the verbalization of SPARQL [1] and RDF in areas such as question answering (Lehmann et al., 2012) and the explanation of the output of systems based onSW technologies (Ngonga Ngomo et al., 2013). However, other SW languages are rarely investigated, such as OWL.

In this paper, we present an open-source holistic NLG framework for the SW, named LD2NL, which facilitates the verbalization of the three key languages of the SW, i.e., RDF, OWL, and SPARQL into NL. Our framework is based on a bottom-up paradigm for verbalizing SW data. Additionally, LD2NL builds upon *SPARQL2NL* as it is open-source and the paradigm it follows can be reused and ported to RDF and OWL. Thus, LD2NL is capable of generating either a single sentence or a summary of a given resource, rule, or query. To validate our framework, we evaluated LD2NL using experts 66 in Natural Language Processing (NLP) and SW as well as 20 non-experts who were lay users or non-users of SW. The results suggest that LD2NL generates texts which can be easily understood by humans. The version of LD2NL used in this paper, all experimental results will be publicly available.

2 Related Work

According to Gatt and Krahmer (2017), there has been a plenty of works which investigated the generation of NL texts from Semantic Web Technologies (SWT) as an input data (Cimiano et al., 2013;

[1]SPARQL is the query language for RDF data.

Proceedings of Recent Advances in Natural Language Processing, pages 819–828,
Varna, Bulgaria, Sep 2–4, 2019.
https://doi.org/10.26615/978-954-452-056-4_095

Duma and Klein, 2013; Ell and Harth, 2014; Biran and McKeown, 2015). However, the subject of research has only recently gained significant momentum due to the great number of published works in the WebNLG (Colin et al., 2016) challenge along with deep learning techniques (Sleimi and Gardent, 2016; Mrabet et al., 2016). RDF has also been showing promising benefits to the generation of benchmarks for evaluating NLG systems (Gardent et al., 2017; Perez-Beltrachini et al., 2016).

Despite the plethora of recent works written on handling RDF data, only a few have exploited the generation of NL from OWL and SPARQL. For instance, Androutsopoulos et al. (2013) generates sentences in English and Greek from OWL ontologies. Also, SPARQL2NL (Ngonga Ngomo et al., 2013) uses rules to verbalize atomic constructs and combine their verbalization into sentences. Therefore, our goal with LD2NL is to provide a complete framework to verbalize SW concepts rather than become the state of the art on the respective tasks.

3 Background

3.1 OWL

OWL[2] (OWL Working Group, 2009) is the de-facto standard for machine processable and interoperable ontologies on the SW. In its second version, OWL is equivalent to the description logic $\mathcal{SROIQ}(D)$. Such expressiveness has a higher computational cost but allows the development of interesting applications such as automated reasoning (Bühmann et al., 2016). OWL 2 ontologies consist of the following three different syntactic categories:

Entities, such as *classes*, *properties*, and *individuals*, are identified by IRIs. They form the primitive terms and constitute the basic elements of an ontology. Classes denote sets of individuals and properties link two individuals or an individual and a data value along a property. For example, a class `:Animal` can be used to represent the set of all animals. Similarly, the object property `:childOf` can be used to represent the parent-child relationship and the data property `:birthDate` assigns a particular birth date to an individual. Finally, the individual `:Alice` can be used to represent a particular person called "Alice".

Expressions represent complex notions in the domain being described. For example, a *class expression* describes a set of individuals in terms of the restrictions on the individuals' characteristics. OWL offers existential (**SOME**) or universal (**ONLY**) qualifiers and a variety of typical logical constructs, such as negation (**NOT**), other Boolean operators (**OR**, **AND**), and more constructs such as cardinality restriction (**MIN**, **MAX**, **EXACTLY**) and value restriction (**VALUE**), to create class expressions. Such constructs can be combined in arbitrarily complex class expressions `CE` according to the following grammar

```
CE = A | C AND D | C OR D | NOT C | R
     SOME C | R ONLY C | R MIN n | R MAX
      n | R EXACTLY n | R VALUE a | {a1
     ,...,am}
```

where `A` is an atomic class, `C` and `D` are class expressions, `R` is an object property, `a` as well as $\mathtt{a}_1$ to $\mathtt{a}_m$ with $\mathtt{m} \geq 1$ are individuals, and $\mathtt{n} \geq 0$ is an integer.

Axioms are statements that are asserted to be true in the domain being described. Usually, one distinguish between (1) *terminological* and (2) *assertional* axioms. (1) terminological axioms are used to describe the structure of the domain, i.e., the relationships between classes resp. class expressions. For example, using a subclass axiom (**SubClassOf:**), one can state that the class `:Koala` is a subclass of the class `:Animal`. Classes can be subclasses of other classes, thus creating a taxonomy. In addition, axioms can arrange properties in hierarchies (**SubPropertyOf:**) and can assign various characteristics (**Characteristics:**) such as transitivity or reflexivity to them. (2) Assertional axioms formulate facts about individuals, especially the classes they belong to and their mutual relationships. OWL can be expressed in various syntaxes with the most common computer readable syntax being RDF/XMLA more human-readable format is the Manchester OWL Syntax (MOS) (Horridge et al., 2006). For example, the class expression that models people who work at a university that is located in Spain could be as follows in MOS:

```
Person AND worksAt SOME (University AND
    locatedIn VALUE Spain)
```

Likewise, expressing that every professor works at a university would read as

```
Class: Professor
  SubClassOf: worksAt SOME University
```

[2] www.w3.org/TR/owl2-overview/

3.2 RDF

RDF (RDF Working Group, 2014) uses a graph-based data model for representing knowledge. Statements in RDF are expressed as so-called triples of the form `(subject, predicate, object)`. RDF subjects and predicates are Internationalized Resource Identifierss (IRIs) and objects are either IRIs or literals.[3] RDF literals always have a datatype that defines its possible values. A predicate denotes a *property* and can also be seen as a binary relation taking subject and object as arguments. For example, the following triple expresses that Albert Einstein was born in Ulm:

```
:Albert_Einstein :birthPlace :Ulm .
```

3.3 SPARQL

Commonly, the selection of subsets of RDF is performed using the SPARQL query language.[4] SPARQL can be used to express queries across diverse data sources. *Query forms* contain variables that appear in a solution result. They can be used to select all or a subset of the variables bound in a pattern match. They exist in four different instantiations, i.e., *SELECT*, *CONSTRUCT*, *ASK* and *DESCRIBE*. The *SELECT* query form is the most commonly used and is used to return rows of variable bindings. Therefore, we use this type of query in our explanation. *CONSTRUCT* allows to create a new RDF graph or modify the existing one through substituting variables in a graph templates for each solution. *ASK* returns a Boolean value indicating whether the graph contains a match or not. Finally, *DESCRIBE* is used to return all triples about the resources matching the query. For example, 1 represents the following query "Return all scientists who were born in Ulm".

```
SELECT ?person
WHERE {
    ?person a dbo:Scientist;
        dbo:birthPlace dbr:Ulm.
}
```

Listing 1: All scientists who were born in Ulm

4 LD2NL Framework

The goal of LD2NL is to provide an integrated system which generates a complete and correct NL representation for the most common used SW modeling languages RDF and OWL, and SPARQL. In terms of the standard model of NL generation proposed by Reiter & Dale (Reiter and Dale, 2000b), our steps mainly play the role of the micro-planner, with focus on aggregation, lexicalization, referring expressions and linguistic realization. In the following, we present our approach to formalizing NL sentences for each of the supported languages.

4.1 From RDF to NL

4.1.1 Lexicalization

The lexicalization of RDF triples must be able to deal with resources, classes, properties and literals.

Classes and resources The lexicalization of classes and resources is carried out as follows: Given a URI u we ask for the English label of u using a SPARQL query.[5] If such a label does not exist, we use either the fragment of u (the string after #) if it exists, else the string after the last occurrence of `/`. Finally this NL representation is realized as a noun phrase, and in the case of classes is also pluralized. As an example, `:Person` is realized as `people` (its label).

Properties The lexicalization of properties relies on the insight that most property labels are either nouns or verbs. While the mapping of a particular property `p` can be unambiguous, some property labels are not as easy to categorize. For examples, the label `crosses` can either be the plural form of the noun `cross` or the third person singular present form of the verb `to cross`. To automatically determine which realization to use, we relied on the insight that the first and last word of a property label are often the key to determining the type of the property: properties whose label begins with a verb (resp. noun or gerund) are most to be realized as verbs (resp. nouns). We devised a set of rules to capture this behavior, which we omit due to space restrictions. In some cases (such as `crosses`) none of the rules applied. In these cases, we compare the probability of $P(p|\texttt{noun})$ and $P(p|\texttt{verb})$ by measuring

$$P(p|X) = \frac{\sum_{t \in synset(p|X)} \log_2(f(t))}{\sum_{t' \in synset(p)} \log_2(f(t'))}, \quad (1)$$

where $synset(p)$ is the set of all synsets of p, $synset(p|X)$ is the set of all synsets of p that are of

[3] For simplicity, we omit RDF blank nodes in subject or object position.

[4] http://www.w3.org/TR/sparql11-query

[5] Note that it could be any property which returns a NL representation of the given URI, see (Ell et al., 2011).

the syntactic class $X \in \{\texttt{noun}, \texttt{verb}\}$ and $f(t)$ is the frequency of use of p in the sense of the synset t according to WordNet. For

$$\frac{P(p|\texttt{verb})}{P(p|\texttt{noun})} \geq \theta, \tag{2}$$

we choose to realize p as a noun; else we realized it as a verb. For $\theta = 1$, for example, dbo:crosses is realized as a verb.

Literals Literals in an RDF graph usually consist of a *lexical form* LF and a *datatype IRI* DT, represented as "LF"^^<DT>. Optionally, if the datatype is rdf:langString, a non-empty *language tag* is specified and the literal is denoted as *language-tagged string*[6]. The realization of language-tagged strings is done by using simply the lexical form, while omitting the language tag. For example, "Albert Einstein"@en is realized as Albert Einstein. For other types of literals, we further differentiate between built-in and user-defined datatypes. For the former, we also use the lexical form, e.g. "123"^^xsd:int ⇒ 123, while the latter are processed by using the literal value with its representation of the datatype IRI, e.g., "123"^^dt:squareKilometre as 123 square kilometres.

4.1.2 Realizing single triples

The realization ρ of a triple (s p o) depends mostly on the verbalization of the predicate p. If p can be realized as a noun phrase, then a possessive clause can be used to express the semantics of (s p o), more formally

1. ρ(s p o) ⇒
   poss(ρ(p),ρ(s)) ∧ subj(BE,ρ(p)) ∧ dobj(BE,ρ(o))

For example, if ρ(p) is a relational noun like birth place e.g. in the triple (:Albert_Einstein :birthPlace :Ulm), then the verbalization is Albert Einstein's birth place is Ulm. Note that BE stands for the verb "to be". In case p's realization is a verb, then the triple can be verbalized as follows:

2. ρ(s p o) ⇒
   subj(ρ(p),ρ(s)) ∧ dobj(ρ(p),ρ(o))

For example, in (:Albert_Einstein :influenced :Nathan_Rosen) ρ(p) is the verb influenced, thus, the verbalization is Albert Einstein influenced Nathan Rosen.

[6] In RDF 1.0 literals have been divided into 'plain' literals with no type and optional language tags, and typed literals.

4.2 Realization - RDF Triples to NL

The same procedure of generating a single triple can be applied for the generation of each triple in a set of triples. However, the NL output would contain redundant information and consequently sound very artificial. Thus, the goal is to transform the generated description to sound more natural. To this end, we focus on two types of transformation rules (cf. (Dalianis and Hovy, 1996)): *ordering and clustering* and *grouping*. In the following, we describe the transformation rules we employ in more detail. Note that clustering and ordering (4.2.1) is applied before grouping (4.2.2).

4.2.1 Clustering and ordering rules

We process the input trees in descending order with respect to the frequency of the variables they contain, starting with the projection variables and only after that turning to other variables. As an example, consider the following triples about two of the most known people in the world:

```
:William_Shakespeare rdf:type :Writer .
:Albert_Einstein :birthPlace :Ulm .
:Albert_Einstein :deathPlace :Princeton
:Albert_Einstein rdf:type :Scientist .
:William_Shakespeare :deathDate
"1616-04-23"^^xsd:date .
```

The five triples are verbalized as given in 3a–3e. Clustering and ordering first take all sentences containing the subject :Albert_Einstein, i.e. 3b –3d, which are ordered such that copulative sentences (such as Albert Einstein is a scientist) come before other sentences, and then takes all sentences containing the remaining subject :William_Shakespeare in 3a and 3e resulting in a sequence of sentences as in 4.

3. (a) William Shakespeare is a writer.
   (b) Albert Einstein's birth place is Ulm.
   (c) Albert Einstein's death place is Princeton.
   (d) Albert Einstein is a scientist.
   (e) William Shakespeare's death date is 23 April 1616.

4. Albert Einstein is a scientist. Albert Einstein's birth place is Ulm. Albert Einstein's death place is Princeton. William Shakespeare's is a writer. William Shakespeare's death date is 23 April 1616.

4.2.2 Grouping

Dalianis and Hovy (1996) describe grouping as a process "collecting clauses with common elements and then collapsing the common elements". The common elements are usually subject noun

phrases and verb phrases (verbs together with object noun phrases), leading to *subject grouping* and *object grouping*. To maximize the grouping effects, we collapse common prefixes and suffixes of sentences, irrespective of whether they are full subject noun phrases or complete verb phrases. In the following we use $X_1, X_2, \ldots X_N$ as variables for the root nodes of the input sentences and Y as variable for the root node of the output sentence. Furthermore, we abbreviate a subject $\texttt{subj}(X_i, s_i)$ as $\texttt{s}_i$, an object $\texttt{dobj}(X_i, o_i)$ as $\texttt{o}_i$, and a verb $\texttt{root}(\textit{ROOT}_i, v_i)$ as $\texttt{v}_i$.

Subject grouping collapses the predicates (i.e. verb and object) of two sentences if their subjects are the same, as specified in 5 (abbreviations as above).

5. $\rho(\texttt{s}_1) = \rho(\texttt{s}_2) \wedge \texttt{cc}(\texttt{v}_1, \textit{coord})$
$\Rightarrow \texttt{root}(Y, \texttt{coord}(\texttt{v}_1, \texttt{v}_2)) \wedge \texttt{subj}(\texttt{v}_1, \texttt{s}_1) \wedge \texttt{dobj}(\texttt{v}_1, \texttt{o}_1) \wedge \texttt{subj}(\texttt{v}_2, \texttt{s}_1) \wedge \texttt{dobj}(\texttt{v}_1, \texttt{o}_2)$

An example are the sentences given in 6, which share the subject Albert Einstein and thus can be collaped into a single sentence.

6. Albert Einstein is a scientist and Albert Einstein is known for general relativity.
$\Rightarrow$ Albert Einstein is a scientist and known for general relativity.

Object grouping collapses the subjects of two sentences if the realizations of the verbs and objects of the sentences are the same, where the $\textit{coord} \in \{\text{and}, \text{or}\}$ is the coordination combining the input sentences X_1 and X_2, and $\texttt{coord} \in \{\texttt{conj}, \texttt{disj}\}$ is the corresponding coordination combining the subjects.

7. $\rho(\texttt{o}_1) = \rho(\texttt{o}_2) \wedge \rho(\texttt{v}_1) = \rho(\texttt{v}_2) \wedge \texttt{cc}(\texttt{v}_1, \textit{coord})$
$\Rightarrow \texttt{root}(Y, \texttt{PLURAL}(\texttt{v}_1)) \wedge \texttt{subj}(\texttt{v}_1, \texttt{coord}(\texttt{s}_1, \texttt{s}_2)) \wedge \texttt{dobj}(\texttt{v}_1, \texttt{o}_1)$

For example, the sentences in 8 share their verb and object, thus they can be collapsed into a single sentence. Note that to this end the singular auxiliary was needs to be transformed into its plural form were.

8. Benjamin Franklin was born in Boston. Leonard Nimoy was born in Boston. $\Rightarrow$ Benjamin Franklin and Leonard Nimoy were born in Boston.

4.3 From OWL to NL

OWL 2 ontologies consist of Entities, Expressions and Axioms as introduced in subsection 3.1. While both expressions and axioms can be mapped to RDF[7], i.e. into a set of RDF triples, using this mapping and applying the triple-based verbalization on it would lead to a non-human understandable text in many cases. For example, the intersection of two classes `:A` and `:B` can be represented in RDF by the six triples

```
_:x rdf:type owl:Class .
_:x owl:intersectionOf _:y1 .
_:y1 rdf:first :A .
_:y1 rdf:rest _:y2 .
_:y2 rdf:first :B .
_:y2 rdf:rest rdf:nil .
```

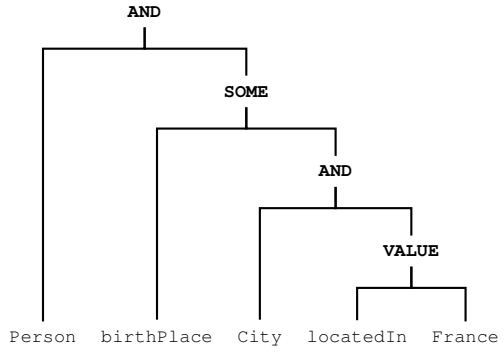
The verbalization of these triples would result in `Something that is a class and the intersection of something whose first is A and whose rest is something whose first is B and whose rest ist nil.`, which is obviously far away from how a human would express it in NL. Therefore, generating NL from OWL requires a different procedure based on its syntactic categories, OWL expressions and OWL axioms. We show the general rules for each of them in the following.

4.3.1 OWL Class Expressions

In theory, class expressions can be arbitrarily complex, but as it turned out in some previous analysis (Power and Third, 2010), in practice they seldom arise and can be seen as some corner cases. For example, an ontology could contain the following class expression about people and their birth place:

```
Person AND birthPlace SOME (City AND
    locatedIn VALUE France)
```

Class expressions do have a tree-like structure and can simply be parsed into a tree by means of the binary OWL class expressions constructors contained in it. For our example, this would result in the following tree:



Such a tree can be traversed in post-order, i.e. sub-trees are processed before their parent nodes recursively. For the sake of simplicity, we only process sub-trees that represent proper

[7] http://bit.ly/2Mc0vIw

class expression in our example, i.e. we omit birthPlace, locatedIn, and France. Moreover and again for simplicity, we'll explain the transformation process by starting from the right-hand side of the tree. Thus, in our example we begin with the class expression City which is transformed to everything that is a city and locatedIn VALUE France resulting in everything that is located in France by application of a rule. Both class expressions are used in the conjunction City AND locatedIn VALUE France. Thus, the next step would be to merge both phrases. An easy way is to use the coordinating conjunction and, i.e. everything that is a city and everything that is located in France. Although the output of this transformation is correct, it still contains unnecessarily redundant information. Therefore, we apply the aggregation procedure described in Section 4.2.2, i.e. we get everything that is a city and located in France. Yet, the aggregation can still be improved: if there is any atomic class in the conjunction, we know that this is more specific than the placeholder everything. Thus, we can replace it by the plural form of the class, finally resulting in cities that are located in France. The same procedure is applied for its parent class expression being the existential restriction

```
birthPlace SOME (City AND locatedIn
    VALUE France)
```

This will be transformed to everything whose birth place is a city that is located in France. Note, that we used the singular form here, assuming that the property birthPlace is supposed to be functional in the ontology. In the last step, we process the class expression Person, which gives us everything that is a person. Again, due to the conjunction we merge this result with with the previous one, such that in the end we get people whose birth place is a city that is located in France.

4.3.2 OWL Axioms

As we described in Section 4.3, OWL axioms can roughly be categorized into terminological and assertional axioms. Therefore, we have different procedures for processing each category:

Assertional Axioms (ABox Axioms) - Most assertional axioms assert individuals to atomic classes or relate individuals to another individual resp. literal value. For example, axioms about the type as well as birth place and birth date of Albert Einstein can be expressed by

```
Individual: Albert_Einstein
  Types: Person
  Facts: birthPlace Ulm, birthDate "
      1879-03-14"^^xsd:date
```

Those axioms can simply be rewritten as triples, thus, we can use the same procedure as we do for triples (Section 4.1.2). Converting them into NL gives us Albert Einstein is a person whose birth place is Ulm and whose birth date is 14 March 1879. OWL also allows for assigning an individual to a complex class expression. In that case we'll use our conversion of OWL class expressions as described in Section 4.3.1.

Terminological Axioms (TBox Axioms) - According to Power and Third (2010), most of the terminological axioms used in ontologies are subclass axioms. By definition, subclass and superclass can be arbitrarily complex class expressions CE_1 and CE_2, i.e. CE_1 **SubClassOf** CE_2, but in praxis it is quite often only used with atomic classes as subclass or even more simple with the superclass also beeing an atomic class. Nevertheless, we support any kind of subclass axiom and all other logical OWL axioms in LD2NL. For simplicity, we outline here how we verbalize subclass axioms in LD2NL. The semantics of a subclass axiom denotes that every individual of the subclass also belongs to the superclass. Thus, the verbalization seems to be relatively straightforward, i.e. we verbalize both class expressions and follow the template : every $\rho(CE_1)$ is a $\rho(CE_2)$. Obviously, this works pretty well for subclass axioms with atomic classes only. For example, the axiom

```
Class: Scientist
  SubClassOf: Person
```

is verbalized as every scientist is a person.

4.4 From SPARQL to NL

A SPARQL SELECT query can be regarded as consisting of three parts: (1) a *body section* B, which describes all data that has to be retrieved, (2) an *optional section* O, which describes the data items that can be retrieved by the query if they exist, and (3) a *modifier section* M, which describes all

solution sequences, modifiers and aggregates that are to be applied to the result of the previous two sections of the query. Let *Var* be the set of all variables that can be used in a SPARQL query. In addition, let R be the set of all resources, P the set of all properties and L the set of all literals contained in the target knowledge base of the SPARQL queries at hand. We call $x \in Var \cup R \cup P \cup L$ an *atom*. The basic components of the body of a SPARQL query are triple patterns $(\mathtt{s}, \mathtt{p}, \mathtt{o}) \in (Var \cup R) \times (Var \cup P) \times (Var \cup R \cup L)$. Let W be the set of all words in the dictionary of our target language. We define the realization function $\rho : Var \cup R \cup P \cup L \rightarrow W^*$ as the function which maps each atom to a word or sequence of words from the dictionary. The extension of ρ to all SPARQL constructs maps all atoms x to their realization $\rho(x)$ and defines how these atomic realizations are to be combined. We denote the extension of ρ by the same label ρ for the sake of simplicity. We adopt a rule-based approach to achieve this goal, where the rules extending ρ to all valid SPARQL constructs are expressed in a conjunctive manner. This means that for premises $P_1, \ldots, P_n$ and consequences $K_1, \ldots, K_m$ we write $P_1 \wedge \ldots \wedge P_n \Rightarrow K_1 \wedge \ldots \wedge K_m$. The premises and consequences are explicated by using an extension of the Stanford dependencies[8].

For example, a possessive dependency between two phrase elements e_1 and e_2 is represented as $\mathtt{poss}(e_1, e_2)$. For the sake of simplicity, we slightly deviate from the Stanford vocabulary by not treating the copula to be as an auxiliary, but denoting it as `BE`. Moreover, we extend the vocabulary by the constructs `conj` and `disj` which denote the conjunction resp. disjunction of two phrase elements. In addition, we sometimes reduce the construct `subj(y,x)` $\wedge$ `dobj(y,z)` to the triple `(x,y,z)` $\in W^3$.

5 Experiments

We evaluated our approach in three different experiments based on human ratings. We divided the volunteers into two groups—domain experts and non-experts. The group of domain experts comprised 66 persons while there were 20 non-experts forming the second group. In the first experiment, an OWL axiom and its verbalization were shown to the experts who were asked to rate the verbalization regarding the two following measures according to Gardent et al. (2017): (1) Adequacy: Does the text contain only and all the information from the data? (2) Fluency: Does the text sound fluent and natural?. For both measures the volunteers were asked to rate on a scale from 1 (Very Bad) to 5 (Very Good). The experiment was carried out using 41 axioms of the Koala ontology.[9] Because of the complexity of OWL axioms, only domain experts were asked to perform this experiment.

In the second experiment, a set of triples describing a single resource and their verbalization were shown to the volunteers. The experts were asked to rate the verbalization regarding adequacy, fluency and *completeness*, i.e., whether all triples have been covered. The non-experts were only asked to rate the fluency. The experiment was carried out using 6 DBpedia resources.In the third experiment, the verbalization of an OWL class and 5 resources were shown to the human raters. For non-experts, the resources have been verbalized as well, while for domain experts the resources were presented as triples. The task of the raters was to identify the resource that fits the class description and, thus, is an instance of the class. We used 4 different OWL axioms and measured the amount of correct identified class instances.

Results In our first series of experiments, the verbalization of OWL axioms, we achieved an average adequacy of 4.4 while the fluency reached 4.38. In addition, more than 77% of the verbalizations were assigned the maximal adequacy (i.e., were assigned a score of 5, see Fig. 1). The maximal score for fluency was achieved in more than 69% of the cases (see Fig. 1). This clearly indicates that the verbalization of axioms generated by LD2NL can be easily understood by domain experts and contains all the information necessary to access the input OWL class expression.

Experiments on the verbalization of summaries for RDF resources revealed that verbalizing resource summaries is a more difficult task. While the adequacy of the verbalization was assigned an average score of 3.92 by experts (see Fig. 2), the fluency was assigned a average score of 3.47 by experts and 3.0 by non-experts (see Fig. 2). What these results suggest is that (1) our framework generates sentences that are close to that which a domain expert would also generate (adequacy). However (2) while the sentence is grammatically

[8] For a complete description of the vocabulary, see https://stanford.io/2EzMjmo.

[9] https://bit.ly/2K8BWts

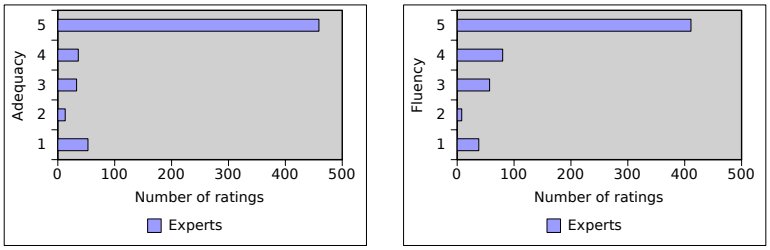


Figure 1: Experiment I: adequacy (left) and fluency (right) ratings

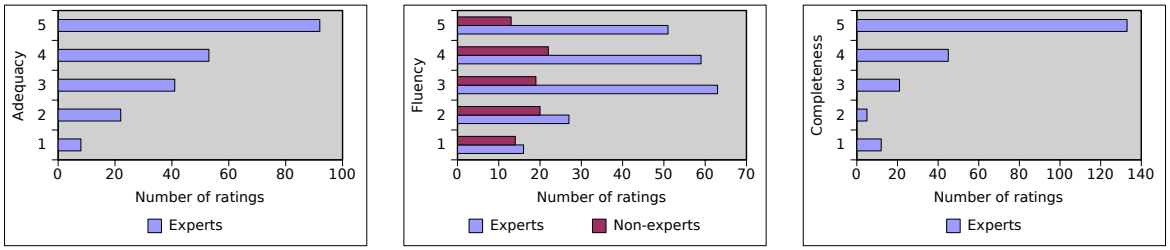


Figure 2: Experiment II: adequacy (left), fluency (middle) and completeness (left) results

sufficient for the experts, it is regarded by non-domain experts (which were mostly linguists, i.e., the worst-case scenario for such an evaluation) as being grammatically passably good but still worthy of improvement. The completeness rating achieves a score of 4.31 on average (see Fig. 2). This was to be expected as we introduced a rule to shorten the description of resources that contain more than 5 triples which share a common subject and predicate. Finally, we measured how well the users and experts were able to understand the meaning of the text generated by our approach. As expected, the domain experts outperform the non-expert users by being able to find the answers to 87.2% of the questions. The score achieved by non-domain experts, i.e., 80%, still suggest that our framework is able to bridge the gap pertaining to understand RDF and OWL for non-experts from 0% to 80%, which is more than 91.8% of the performance of experts.

Discussion Our evaluation results suggest that the verbalization of these languages is a non-trivial task that can be approached by using a bottom-up approach. As expected, the verbalization of short expressions leads to sentences which read as if they have been generated by a human. However, due to the complexity of the semantics that can be expressed by the languages at hand, long expressions can sound mildly artificial. Our results however also suggest that although the text generated can sound artificial, it is still clear enough to enable non-expert users to achieve results that are comparable to those achieved by experts. Hence, our first conclusion is that our framework clearly serves its purpose. Still, potential improvements can be derived from the results achieved during the experiments. In particular, we will consider the used of attention-based encoder-decoder networks to improve the fluency of complex sentences.

6 Conclusion and Future Work

In this paper, we presented LD2NL, a framework for verbalizing SW languages, especially on RDF and OWL while including the SPARQL verbalization provided by SPARQL2NL. Our evaluation with 86 persons revealed that our framework generates NL that can be understood by lay users. While the OWL verbalization was close to NL, the RDF was less natural but still sufficient to convey the meaning expressed by the corresponding set of triples. In future work, we aim to extend LD2NL to verbalize the languages SWRL (Horrocks et al., 2004) and SHACL (Knublauch and Kontokostas, 2017).

Acknowledgments

This work was supported by the German Federal Ministry of Transport and Digital Infrastructure (BMVI) through the projects LIMBO (no. 19F2029I) and OPAL (no. 19F2028A). This work was supported by the German Federal Ministry of Economics and Technology (BMWI) in the projects RAKI (no. 01MD19012D) as well as by the BMBF project SOLIDE (no. 13N14456).

References

Ion Androutsopoulos, Gerasimos Lampouras, and Dimitrios Galanis. 2013. Generating natural language descriptions from OWL ontologies: The natural owl system. *J. Artif. Int. Res.* 48(1):671–715.

Or Biran and Kathleen McKeown. 2015. Discourse planning with an n-gram model of relations. In *EMNLP*. pages 1973–1977.

Lorenz Bühmann, Jens Lehmann, and Patrick Westphal. 2016. Dl-learnera framework for inductive learning on the semantic web. *Journal of Web Semantics* 39:15–24.

Philipp Cimiano, Janna Lüker, David Nagel, and Christina Unger. 2013. Exploiting ontology lexica for generating natural language texts from rdf data. In *Proceedings of the 14th European Workshop on Natural Language Generation*. ACL, Sofia, Bulgaria, pages 10–19.

Emilie Colin, Claire Gardent, Yassine Mrabet, Shashi Narayan, and Laura Perez-Beltrachini. 2016. The webnlg challenge: Generating text from dbpedia data. In *Proceedings of the 9th INLG conference*. pages 163–167.

H. Dalianis and E.H. Hovy. 1996. Aggregation in natural language generation. In G. Adorni and M. Zock, editors, *Trends in natural language generation: an artificial intelligence perspective*, Springer, volume 1036 of *Lecture Notes in Artificial Intelligence*, pages 88–105.

Daniel Duma and Ewan Klein. 2013. Generating natural language from linked data: Unsupervised template extraction. In *IWCS*. pages 83–94.

Basil Ell and Andreas Harth. 2014. A language-independent method for the extraction of rdf verbalization templates. In *INLG*. pages 26–34.

Basil Ell, Denny Vr, and Elena Simperl. 2012. SPARTIQULATION Verbalizing SPARQL queries. In *In Proceedings of ILD Workshop, ESWC*.

Basil Ell, Denny Vrandecic, and Elena Paslaru Bontas Simperl. 2011. Labels in the web of data. In *Proceedings of ISWC*. Springer, volume 7031, pages 162–176.

Claire Gardent, Anastasia Shimorina, Shashi Narayan, and Laura Perez-Beltrachini. 2017. Creating training corpora for nlg micro-planning. In *Proceedings of ACL*.

Albert Gatt and Emiel Krahmer. 2017. Survey of the state of the art in natural language generation: Core tasks, applications and evaluation. *arXiv preprint arXiv:1703.09902* .

Matthew Horridge, Nick Drummond, John Goodwin, Alan L Rector, Robert Stevens, and Hai Wang. 2006. The Manchester OWL syntax. In *OWLed*. volume 216.

Ian Horrocks, Peter F. Patel-Schneider, Harold Boley, Said Tabet, Benjamin Grosofand, and Mike Dean. 2004. SWRL: A semantic web rule language combining OWL and RuleML. W3C Member Submission.

Holger Knublauch and Dimitris Kontokostas. 2017. Shapes constraint language (shacl). *W3C Candidate Recommendation* 11(8).

Jens Lehmann, Tim Furche, Giovanni Grasso, Axel-Cyrille Ngonga Ngomo, Christian Schallhart, Andrew Jon Sellers, Christina Unger, Lorenz Bühmann, Daniel Gerber, Konrad Höffner, David Liu, and Sören Auer. 2012. DEQA: Deep Web Extraction for Question Answering. In *The Semantic Web - ISWC 2012 - 11th International Semantic Web Conference, Boston, MA, USA, November 11-15, 2012, Proceedings, Part II*. pages 131–147.

Yassine Mrabet, Pavlos Vougiouklis, Halil Kilicoglu, Claire Gardent, Dina Demner-Fushman, Jonathon Hare, and Elena Simperl. 2016. Aligning texts and knowledge bases with semantic sentence simplification. *WebNLG 2016* .

Axel-Cyrille Ngonga Ngomo, Lorenz Bühmann, Christina Unger, Jens Lehmann, and Daniel Gerber. 2013. Sorry, I Don't Speak SPARQL: Translating SPARQL Queries into Natural Language. In *Proceedings of the 22Nd International Conference on World Wide Web*. ACM, New York, NY, USA, pages 977–988.

Axel-Cyrille Ngonga Ngomo, Lorenz Bühmann, Christina Unger, Jens Lehmann, and Daniel Gerber. 2013. Sorry, i don't speak sparql: translating sparql queries into natural language. In *Proceedings of the 22nd international conference on World Wide Web*. ACM, pages 977–988.

Axel-Cyrille Ngonga Ngomo, Lorenz Bühmann, Christina Unger, Jens Lehmann, and Daniel Gerber. 2013. SPARQL2NL: Verbalizing SPARQL queries. In *22nd International World Wide Web Conference, Rio de Janeiro, Brazil, May 13-17*. pages 329–332.

W3C OWL Working Group. 2009. *OWL 2 Web Ontology Language: Document Overview*. W3C Recommendation.

Laura Perez-Beltrachini, Rania Sayed, and Claire Gardent. 2016. Building rdf content for data-to-text generation. In *COLING*. pages 1493–1502.

Richard Power and Allan Third. 2010. Expressing OWL Axioms by English Sentences: Dubious in Theory, Feasible in Practice. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*. ACL, Stroudsburg, PA, USA, pages 1006–1013.

W3C RDF Working Group. 2014. *RDF 1.1 Concepts and Abstract Syntax*. W3C Recommendation.

Ehud Reiter and Robert Dale. 2000a. *Building natural language generation systems*. Cambridge university press.

Ehud Reiter and Robert Dale. 2000b. *Building natural language generation systems*. Cambridge University Press, New York, NY, USA.

Amin Sleimi and Claire Gardent. 2016. Generating paraphrases from dbpedia using deep learning. *WebNLG 2016* page 54.