Explaining Commonalities of Clusters of RDF Resources in Natural Language

Simona Colucci, Francesco M Donini and Eugenio Di Sciascio

April 30, 2024

Explaining commonalities of clusters of RDF resources in natural language

Simona Colucci$^{1[0000-0002-8774-4816]}$, Francesco M. Donini$^{2[0000-0003-0284-9625]}$, and Eugenio Di Sciascio$^{3[0000-0002-5484-9945]}$

[1] Politecnico di Bari, Via Orabona 4, 70125, Bari, Italy
[2] Università della Tuscia, Via S. Maria in Gradi, 4, 01100 Viterbo, Italy

Abstract. We introduce a system that provides explanations in Natural Language for individual clusters of RDF resources, where clusters are obtained using an external clustering tool. Our system is based on the theory of (Least) Common Subsumers (CS) in RDF. We propose an optimized algorithm for computing a CS, which allows us to compute the CS for up to 80 RDF resources (each with its own RDF-graph of linked data). We then generate a Natural Language sentence to describe each cluster. A unique aspect of our explanations is the use of relative sentences, including nested ones, to represent blank nodes in an RDF-path. We demonstrate the usefulness of our tool by describing the resulting clusters of a real, publicly available, dataset on Public Procurements.

Keywords: Explainable Artificial Intelligence (XAI) · Resource Description Framework (RDF) · Least Common Subsumer (LCS) · Natural Language Generation (NLG)

1 Introduction and Related Work

The success of subsymbolic approaches to the clusterization problem has increased the importance of describing the obtained clusters — *i.e.*, explaining why some items were put together — both to developers who tune parameters, and to end users [3]. For instance, a decision tree with k leaves has been recently proposed as an explanation of the clustering obtained by the well-known method of k-means [16], and by finding a minimal set of outliers to be excluded by the clusterization, one can approximate a "reasonably-sized" decision tree [4]. Yet, while decision trees and other data-structure-oriented descriptions could be well suited for Computer Science developers and practitioners, explanations using Natural Language — when available — are still among the most understandable ones for end users, given that an explanation is a *social interaction* [15].

A different perspective is given by the distinction between model-aware vs. model-agnostic explanation methods. Some clustering methods are self-explaining: conceptual clustering [14] was introduced in 1980 as the problem of returning, together with clustering results, a concept explaining the criterion for resources aggregation. A recent review [18] collected most influential proposals on the subject, but does not include any approach dealing with RDF resources. Again, while model-aware explanations describing the inner clustering mechanisms can be useful for developers' fine-tuning activities, model-agnostic explanations just describing the common characteristics of an obtained cluster seem to be preferable for communicating the results to lay users.

Motivated by the above considerations, in this paper we present a system that given any clusterization method for RDF resources in an RDF dataset, returns a description of the main commonalities in a cluster by a plain English sentence. Our system builds on a formal foundation of Least Common Subsumers of a set of RDF resources [6], but improves both on some computational aspects and the Natural Language generation (NLG) of descriptions.

After some preliminary notions (Section 2), we present in Section 3 the optimized algorithm computing the CS of several resources. An NLG tool for CS explanation is introduced in Section 4 and validated in Section 5 w.r.t. to a publicly available dataset for Public Procurement. Conclusion and future work close the paper.

2 Preliminaries

We assume the reader familiar with RDF and RDFS [17], fully covered by textbooks [11]. We recall here just some basic concepts and definitions about RDF Common Subsumers [6, 8] to make the paper self-contained.

When comparing RDF resources, it is indispensable to select the portion of Linked Open Data [20] more suitable to describe them. In fact, taking into account all triples unboundedly linked to a resource (also in a limited set of datasets) would make unscalable most of applications. Thus we refer to an RDF resource as a pair $\langle r, T_r \rangle$ — which we call *rooted* RDF-*graph* (in brief *r-graph*) — collecting the resource identifier r and a set of triples used for its description [6] with the condition that from r every other triple in T_r can be reached through an RDF-*path* (see below). Coherently, both Simple Entailment between r-graphs can be defined [6, Def.6], denoted as $\langle a, T_a \rangle \models \langle b, T_b \rangle$, and Common Subsumers (CS) in RDF [6, Def.7], which are themselves modeled as r-graphs of the form $\langle x, T_x \rangle$ (where x is a blank node unless $a = b$) such that both $\langle a, T_a \rangle \models \langle x, T_x \rangle$ and $\langle b, T_b \rangle \models \langle x, T_x \rangle$.

RDF-graphs have some peculiarities w.r.t. usual graphs (including Knowledge Graphs), which ask for the extension of some basic notions of Graph Theory. First, RDF admits also paths connected through the predicate (*i.e.*, the predicate of a triple stands as subject in another one). Thus, we define an RDF-*path* of length n from r to s as a sequence of triples $t_0, t_1, \ldots, t_n$ in which (1) the subject of t_0 is r, (2) for $i = 1, ..., n-1$, either the predicate or the object of t_i is the subject of t_{i+1}, and (3) *either* the predicate *or* the object of t_n is s [6]. Such a peculiarity changes also the notion of connectedness, which may occur also through the predicate; a resource r is RDF-*connected* to a resource s if there exists an RDF-path from r to s [6]. The RDF-*distance* between two resources is the length of the shortest RDF-path between them [6]. The distance between a resource r and a full triple t is the RDF-distance between r and the subject of t; note that triples whose subject is r have zero-RDF-distance from r itself.

The representation of RDF resources as r-graphs $\langle r, T_r \rangle$ asks for some criteria for the selection of triples to include in T_r describing a resource r. We encapsulate all selection criteria in a Boolean predicate ϕ [6]: only triples satisfying ϕ are selected. Our current implementation asks for the specification of three parameters determining the value of ϕ: i) the datasets to analyse; ii) the RDF-distance for exploration; iii) the list of so-called *stop-patterns* (analogous to stop-words to be discarded in search engines). In our proposal, ϕ aims at focusing on triples which are significant for the computation of a CS: tuning ϕ may lead to r-graphs which are more representative of the main commonalities in a CS. To further increase the significance of a CS, we iteratively eliminate from it triples that provide little information, called *uninformative triples*. Stop-patterns and uninformative triples include both general patterns/triples discarded in every application domain, and some domain-dependent patterns/triples.

3 An optimized algorithm for the CS of several resources

We recall [6, 8] that the operation of computing a (Least) Common Subsumer is associative — that is, to compute the LCS of three or more r-graphs, one can start from computing the subsumer of any pair of them, and use the result to add the third one, etc. in any order. Hence below we first present Algorithm 1 for the computation of a CS of just two r-graphs, whose iteration to several r-graphs is just hinted at the end of this section.

Algorithm 1 is an optimized version of a previously published algorithm [6], which computes a CS $\langle x, T_x \rangle$ of two r-graphs $\langle a, T_a \rangle$, $\langle b, T_b \rangle$. Mimicking the depth-first search in n-ary trees, $\langle x, T_x \rangle$ is computed incrementally with respect to the triples (filtered by ϕ) directly outgoing from each resource, and recursively when the predicates and/or the objects of such triples are subjects of other triples. More precisely, for each pair of triples $t_1 = \ll a\ p\ c \gg, t_2 = \ll b\ q\ d \gg$, both satisfying $\phi(t_1), \phi(t_2)$, Algorithm 1 first recursively computes a CS $\langle y, T_y \rangle$ of the two predicates p, q, and a CS $\langle z, T_z \rangle$ of objects c, d, then it forms a provisional r-graph $\langle x, T_w \rangle$ with $T_w = \{\ll x\ y\ z \gg\} \cup T_y \cup T_z\}$. Then it adds the triples in T_w to T_x *only if* $\langle x, T_x \rangle$ does *not* entail $\langle x, T_w \rangle$ (Line 21).

Of course, Line 21 is only an optimization step, since in general it was proved [2] for Description Logics that the size of the *Least* Common Subsumer of several tree-shaped concepts can be exponential in the number of concepts, and that proof could also be repeated for r-graphs[3]. Also regarding time consumption, observe that Algorithm 1 could be launched with the two arguments equal to an r-graph $\langle x, G \rangle$ (with x any subject of a triple that reaches all other resources), and it is well known that finding a *lean*[4] equivalent of an RDF-graph G is NP-complete [19]. Since Algorithm 1 runs in time polynomial in the sizes of its

[3] The same conclusion was recently reached [1] using cycles instead of trees.

[4] A lean graph G [17] is an RDF-graph which is $\subseteq$-minimal with respect to all other RDF-graphs logically equivalent to G.

arguments, we cannot expect it to yield a minimal r-graph (unless $P = NP$), but only a reduced version of it. Nevertheless, on the real RDF datasets we tested, this optimization greatly reduces the size of the CS.

By iterating Algorithm 1 — *i.e.*, starting from an initial pair of r-graphs and using its returned CS as the input of the next iteration with another r-graph — we can compute a CS of the whole cluster. By using the optimization of Line 21, we were able to compute a CS of clusters up to 80 resources — *i.e.*, the size of the largest cluster returned in our experiments – and cascade such a CS to the verbalization module.

4 Explaining commonalities of sets of RDF resources

The triple set computed by iterating Algorithm 1 may be serialized in different formats and visualized as a graph, but none of these transformations is intelligible by unacquainted readers. Thus, we designed and developed a Natural Language Generation (NLG) tool able to explain in natural language the content of a CS.

Traditional NLG approaches for RDF are based on the application of rules and templates, *i.e.*, solutions highly domain-dependent and demanding manual intervention. Recently, the advancements in deep learning gave a boost to neural network-based NLG models (see [23] and [13], among others). In both approaches, the generated text considers only triples already present in the input RDF graphs and not including blank nodes.

We here propose a template-based approach, generating text from r-graphs logically computed from input RDF-graphs: CSs; such r-graphs, by construction, include blank nodes, that may also occur in positions other than the root. As an example, consider a group of contracting processes described in the dataset TheyBuyForYou [21] (further introduced in Section 5). The CS collecting their commonalities may include the following triples:

```
_:x <http://data.tbfy.eu/ontology/ocds#hasRelease>  _:y .
_:y <http://data.tbfy.eu/ontology/tbfy#releaseDate> "2019-01-14T00:00:00+00:00" .
```

which means that all contracting processes in the group (abstracted in the blank node `_:x`) have some release (`_:y`) dated 14 January 2019. Notably, `_:y` does not occur by itself in the input r-graphs but represents the fact that different contracting processes refer to different releases, yet all dated 14 January 2019.

To the best of our knowledge, the tool we present is the only NLG tool able to verbalize RDF triples involving blank nodes in any position. Other available tools[5] do not mention such a capability. Furthermore, Bouayad-Agha *et al.* [5] surveyed 11 NLG approaches working on RDF graphs in 2014; none of them was able to manage anonymous resources or generate text from triples not explicitly in the input RDF graph (*i.e.*, derived triples).

In the design of our NLG tool, we follow the approach of Gatt and Krahmer [10], synthesized in the following six tasks:

Content Determination. This step aims at identifying the portion of available informative content to include in the text to generate. In our tool, such a

[5] `https://aclweb.org/aclwiki/Downloadable_NLG_systems`

```
Algorithm $Find\_ReducedCS(\langle a, T_a\rangle, \langle b, T_b\rangle, n)$
Input      : $\langle a, T_a\rangle, \langle b, T_b\rangle$ : a pair of r-graphs;
                       $n$ : the RDF-distance for RDF-graphs exploration;
Output     : $\langle cs, T_{cs}\rangle$ : r-graph s.t. both $\begin{cases} \langle a, T_a\rangle \models \langle cs, T_{cs}\rangle \\ \langle b, T_b\rangle \models \langle cs, T_{cs}\rangle \end{cases}$
Subroutine : Simple entailment between r-graphs $\langle x, T_x\rangle \models \langle y, T_y\rangle$ [6]
Global     :            $\phi$ : boolean predicate selecting triples;
variables    $uninf\_triples$ : triple patterns to eliminate from the results
                        $S$ : set of records $[a, b, \langle w, T_w\rangle]$ s.t. $\begin{cases} (a, b) \text{ was already examined} \\ \langle w, T_w\rangle \text{ is their CS} \end{cases}$
Local      : $\langle x, T_x\rangle$ : the r-graph to be returned, incrementally built
variables

if there is already the record $[a, b, \langle w, T_w\rangle] \in S$ then
    let $\langle x, T_x\rangle = \langle w, T_w\rangle$
else
    add $[a, b, \langle x, T_x\rangle]$ to $S$;
    if $a = b$ then
        let $\langle x, T_x\rangle = \langle a, T_a\rangle$
    else
        let $x$ be a new blank node not occurring in $S$;
        let $T_x = \emptyset$
    end
    if $n > 0$ then
        foreach $t_1 = \ll a\ p\ c \gg$ such that $\phi(t_1) = true$ do
            foreach $t_2 = \ll b\ q\ d \gg$ such that $\phi(t_2) = true$ do
                let $\langle y, T_y\rangle = Find\_ReducedCS(\langle p, T_p\rangle, \langle q, T_q\rangle, n - 1)$;
                let $\langle z, T_z\rangle = Find\_ReducedCS(\langle c, T_c\rangle, \langle d, T_d\rangle, n - 1)$;
                let $\langle x, T_w\rangle = \langle x, \{\ll x\ y\ z \gg\} \cup T_y \cup T_z\}\rangle$;
                while $\exists t \in T_w$ s.t. $t$ matches a pattern in $uninf\_triples$ do
                    delete $t$ from $T_w$
                end
                if $\langle x, T_x\rangle \not\models \langle x, T_w\rangle$ then
                    add $T_w$ to $T_x$
                end
            end
        end
    end
end
return $\langle x, T_x\rangle$;
```

Algorithm 1: Optimized construction of a CS of $\langle a, T_a\rangle$ and $\langle b, T_b\rangle$

content coincides with a computed CS, which by construction holds a compact representation of all commonalities we consider significant. In fact, we: i) include in the input set only triple patterns which are relevant to the problem of finding similarities (Algorithm 1, Row 12); ii) exclude from the CS all triples too generic to be informative (Algorithm 1, Row 17).

Text Structuring. This task deals with deciding how to structure the content above in a readable text. In our case, the content is organized in an r-graph, in which different paths are RDF-connected to the root and include triples at variable RDF-distance from the root. The tool generates one sentence (with subject in the root) for each full path. Such sentences are presented in the order the paths appear in the CS construction, supposing they are equally informative.

Sentence Aggregation. The RDF-connection is a native criterion for sentence aggregation: each path RDF-connected to the root becomes one sentence.

Lexicalisation. This step aims at finding the right words to express information. Our approach exploits the inherent structure of paths for the lexicalization of triples, that depends on their RDF-distance form the root. At every RDF-distance, the subject of a triple is lexicalized with a (different) pronoun, the predicate with a verb in present tense and the object with a noun. Such nouns and predicates are collected in a dictionary. If the predicate and/or the object is a blank node, it is generated a phrase ("some generic resource"), further explained through a relative sentence when the resource has successors in the r-graph.

Referring Expression Generation. This task deals with selecting the words and phrases to identify domain objects. In our case, such entities are triple subjects at any level of the RDF path to consider, represented by blank nodes. We refer to them through generic pronouns, followed by (possibly recursive) relative sentences lexicalizing paths connected to them. All such information are organized around the root: the tool generates sentences whose main subject is the phrase corresponding to the root ("They all").

Linguistic Realisation. The final step consists in combining all words and phrases into well-formed sentences following a human-crafted grammar-based approach. We give below the main rules of the grammar (terminal symbols are quoted and vertical bar represents a choice between two forms of a rule).

$$\begin{aligned} CS &\rightarrow \text{"They all"}\ \mathit{Predicate}\ (\mathit{Noun} \mid \mathit{Noun}\ RC) \\ RC &\rightarrow \text{"which"}\ \mathit{Predicate}\ (\mathit{Noun} \mid \mathit{Noun}\ RC) \end{aligned}$$

In the grammar above: *RC* is a nonterminal representing a relative clause; *Predicate* (respectively *Noun*) is a nonterminal describing the text linguistically realizing a term in the position of triple predicate (respectively, subject/object). If the term is a blank node, the phrase "some resource" linguistically realizes it. We notice that *Predicate* and *Noun* produce a finite number of terminals; thus, the substitution of all such terminals in the above rules yields a (very lengthy) right-recursive Type-3 Grammar. Such a grammar guides the implementation of the linguistic realisation, that follows a breadth-first strategy. In particular, if the CS includes multiple (say, n) RDF-paths, we first generate n phrases, one for the first triple of each path. Such phrases are complete if the path has length 0 (just one triple), or refer to a nonterminal *RC* when the path has length ≥ 1.

The approach proceeds by browsing each path one triple further: their verbalization is added by substituting all non terminals RC once. The exploration ends when no path extends further.

5 Use case: explaining clusters in TheyBuyForYou

We validate the NLG tool presented in Section 4 w.r.t. the public procurement knowledge domain, modeled in the "TheyBuyForYou" dataset [21]. TheyBuyForYou knowledge graph includes an ontology for procurement data, based on the Open Contracting Data Standard (OCDS) [22] . The OCDS data model is built around the concept of a contracting process, which models the procedures followed by a business entity when purchasing services or products.

Our use case shows the commonalities shared by different contracting processes, collected in the same cluster by the well-known clustering algorithm k-Means [12]. We implemented k-Means with $k = 250$ by *Scikit learn*[6] on the set of contracting processes released on 14 Jan. 2019[7], which includes 3,198 resources. The first returned cluster is made up of 14 resources, to which we iteratively applied Algorithm 1, finding a CS of them all. Figure 1 shows the text generated by our NLG tool to explain the commonalities in the above cluster.

We stress that we do not evaluate here clustering results: our approach is agnostic w.r.t. to the clustering process and aims at generating human-readable text corresponding to a set of RDF triples modeling the commonalities of a group of resources.

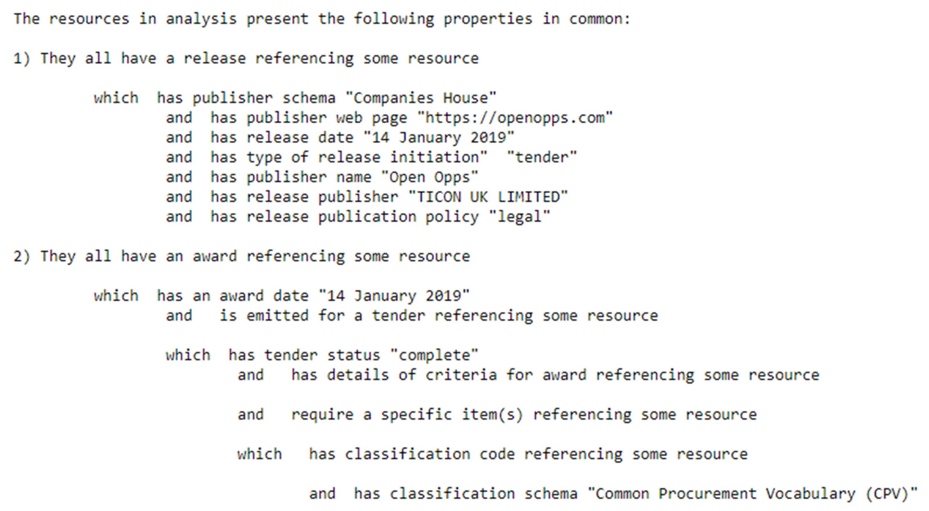
Intuitively, when finding commonalities in a set of resources, the addition of a new item may only reduce the CS, if the fresh resource does not share the full content collected up to its addition. In fact, in Figure 2 we show the explanation of the commonalities of the first seven resources in the analyzed cluster. The reader may notice that the explanation in Figure 2 is more informative (*i.e.*, specific) than the one in Figure 1, because it describes a smaller set of resources.

We now discuss two distinguishing features of our NLG approach.

First, the ability of managing blank nodes is crucial for abstracting several triples with common predicate/object. In both figures, the phrase "some resource" translates blank nodes with successors in the CS r-graph; triples rooted in such blank nodes are further explained in relative sentences whose common subject is the pronoun "which". As an example, consider Commonality 1) in Figure 1. Each contracting process in the cluster has a release that references one resource, which, although different from all other contracting processes, shares with them 7 features (publisher name, release initiation, etc.). All the commonalities recursively expressed in relative sentences would be lost without considering blank nodes. Instead, our method uses blank nodes to chain triples reaching the same known object, until the maximum exploration depth.

[6] `https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html`

[7] The full Knowledge Graph is downloadable at `https://tbfy.github.io/data/`



```
The resources in analysis present the following properties in common:

1) They all have a release referencing some resource

        which  has publisher schema "Companies House"
               and  has publisher web page "https://openopps.com"
               and  has release date "14 January 2019"
               and  has type of release initiation"  "tender"
               and  has publisher name "Open Opps"
               and  has release publisher "TICON UK LIMITED"
               and  has release publication policy "legal"

2) They all have an award referencing some resource

        which  has an award date "14 January 2019"
               and   is emitted for a tender referencing some resource

               which  has tender status "complete"
                      and   has details of criteria for award referencing some resource

                      and   require a specific item(s) referencing some resource

                      which   has classification code referencing some resource

                              and  has classification schema "Common Procurement Vocabulary (CPV)"
```

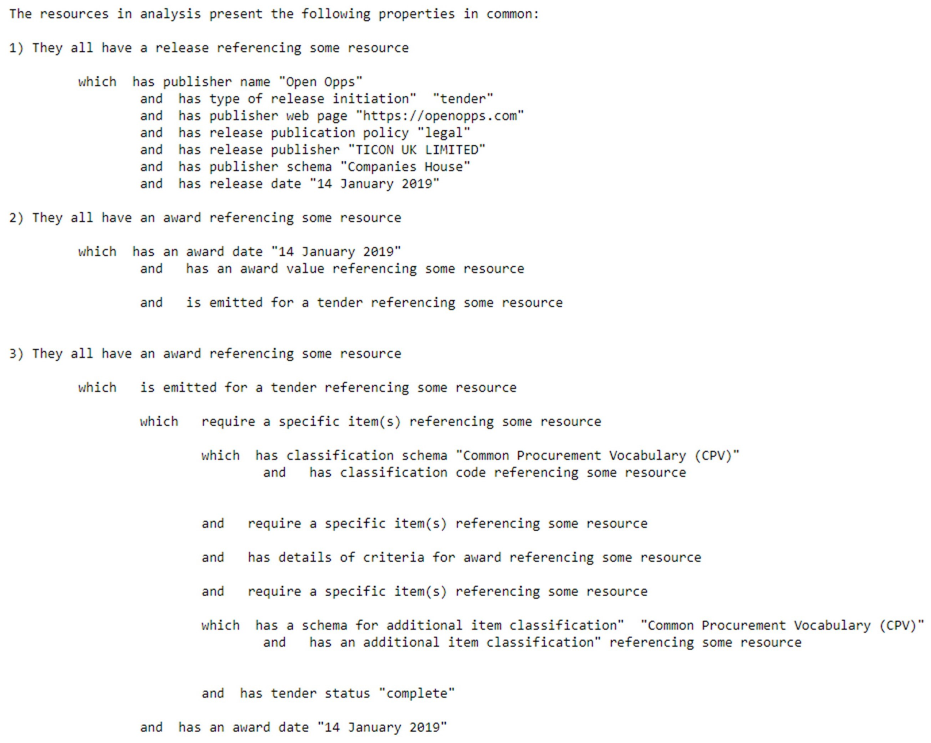
Fig. 1. Explanation provided by our NLG tool of a cluster of 14 contracting processes, returned by running k-means (k=250) over the whole set (3,198 resources) of contracting processes emitted on 14 January 2019.

Second, unlike NLG-based summarization approaches, our method does not just verbalize the set of triples explicitly included in input RDF-graphs; it generates human-readable text from a *newly computed* set of RDF triples that logically represent the commonalities shared by groups of resources. As a consequence, the informative potential of the returned explanation is rather significant and double-tied to the logic-based nature of computation.

6 Conclusion and Future Work

We implemented a *post hoc*, model-agnostic explanation system that provides natural language descriptions of single clusters of RDF resources, previously obtained by any external clusterization tool. Our system is based on the theory of (Least) Common Subsumers (CS) in RDF [6, 8]. We presented an optimized version of an algorithm for computing a CS, that allowed us to compute the CS of the largest cluster in our experiments (80 RDF resources, each one with its own RDF-graph of linked data), to which we pipeline the generation of a Natural Language sentence describing it. An original feature of our explanations is the use of (possibly nested) relative sentences to represent blank nodes in an RDF-path. The application of our tool to a real dataset regarding Public Procurements shows both its usefulness in describing the obtained clusters and the possibility to use it in an interactive process to find more meaningful, fine-grained clusters.

The results of our tool show that although all the characteristics described by the CS are actually common to all resources, some of them could be considered more relevant than others for an end user. Hence, in the near future, we aim to



```
The resources in analysis present the following properties in common:

1) They all have a release referencing some resource

         which  has publisher name "Open Opps"
                 and  has type of release initiation"  "tender"
                 and  has publisher web page "https://openopps.com"
                 and  has release publication policy "legal"
                 and  has release publisher "TICON UK LIMITED"
                 and  has publisher schema "Companies House"
                 and  has release date "14 January 2019"

2) They all have an award referencing some resource

         which  has an award date "14 January 2019"
                 and   has an award value referencing some resource

                 and   is emitted for a tender referencing some resource


3) They all have an award referencing some resource

         which   is emitted for a tender referencing some resource

                 which   require a specific item(s) referencing some resource

                         which  has classification schema "Common Procurement Vocabulary (CPV)"
                                 and   has classification code referencing some resource


                         and   require a specific item(s) referencing some resource

                         and   has details of criteria for award referencing some resource

                         and   require a specific item(s) referencing some resource

                         which  has a schema for additional item classification"  "Common Procurement Vocabulary (CPV)"
                                 and   has an additional item classification" referencing some resource


                         and  has tender status "complete"

                 and  has an award date "14 January 2019"
```

Fig. 2. Explanation provided by our NLG tool of the commonalities of 7 contracting processes selected in the cluster of 14 previously explained in Figure 1.

add to our tool a filter by relevance of the characteristics in the CS, considering that there are two types of relevance [7, 9]: one relative to the general context, and one relative to the user's previous knowledge.

Data availability: Data publicly available at `https://tbfy.github.io/data/`

References

1. Amendola, G., Manna, M., Ricioppo, A.: Characterizing nexus of similarity within knowledge bases: A logic-based framework and its computational complexity aspects, `https://arxiv.org/pdf/2303.10714.pdf`
2. Baader, F., Küsters, R., Molitor, R.: Computing least common subsumers in description logics with existential restrictions. IJCAI. vol. 99, pp. 96–101 (1999)
3. Bae, J., Helldin, T., Riveiro, M., Nowaczyk, S., Bouguelia, M.R., Falkman, G.: Interactive clustering: A comprehensive review. ACM Comput. Surv. **53**(1) (2020)
4. Bandyapadhyay, S., Fomin, F.V., Golovach, P.A., Lochet, W., Purohit, N., Simonov, K.: How to find a good explanation for clustering? Artif. Intell. **322** (2023)

5. Bouayad-Agha, N., Casamayor, G., Wanner, L.: Natural language generation in the context of the semantic web. Semantic Web **5**(6), 493–513 (2014)
6. Colucci, S., Donini, F., Giannini, S., Di Sciascio, E.: Defining and computing least common subsumers in RDF. Web Semantics: Science, Services and Agents on the World Wide Web **39**, 62 – 80 (2016)
7. Colucci, S., Di Noia, T., Donini, F.M., Pomo, C., Di Sciascio, E.: Irrelevant explanations: a logical formalization and a case study. Proc. of the 4th Italian Workshop on Explainable Artificial Intelligence. CEUR Workshop Proceedings, vol. 3518 (2023)
8. Colucci, S., Donini, F.M., Di Sciascio, E.: Common subsumbers in RDF. Proceedings of AI*IA-2013. LNCS, vol. 8249, pp. 348–359. Springer (2013)
9. Colucci, S., Donini, F.M., Di Sciascio, E.: On the relevance of explanation for RDF resources similarity. Model-Driven Organizational and Business Agility - Third International Workshop, MOBA 2023. LNBIP, vol. 488, pp. 96–107. Springer (2023)
10. Gatt, A., Krahmer, E.: Survey of the state of the art in natural language generation: Core tasks, applications and evaluation. J. Artif. Int. Res. **61**(1), 65–170 (jan 2018)
11. Hitzler, P., Krötzsch, M., Rudolph, S.: Foundations of Semantic Web Technologies. Chapman & Hall/CRC (2009)
12. Jain, A.K., Dubes, R.C.: Algorithms for Clustering Data. Prentice-Hall, Inc., Upper Saddle River, NJ, USA (1988)
13. Li, J., Cheng, G., Liu, Q., Zhang, W., Kharlamov, E., Gunaratna, K., Chen, H.: Neural entity summarization with joint encoding and weak supervision. Proceedings of IJCAI-2020. pp. 1644–1650. ijcai.org (2020)
14. Michalski, R.S.: Knowledge acquisition through conceptual clustering: A theoretical framework and an algorithm for partitioning data into conjunctive concepts. Int. Journal of Policy Analysis and Information Systems **4**, 219—-244 (1980)
15. Miller, T.: Explanation in artificial intelligence: Insights from the social sciences. Artificial Intelligence **267**, 1 – 38 (2019)
16. Moshkovitz, M., Dasgupta, S., Rashtchian, C., Frost, N.: Explainable k-means and k-medians clustering. Proc. of the 37th Int. Conf. on Machine Learning. Proc. of Machine Learning Research, vol. 119, pp. 7055–7065. PMLR (13–18 Jul 2020)
17. Patel-Schneider, P., Arndt, D., Haudebourg, T.: RDF 1.2 semantics, W3C recommendation (2023)
18. Pérez-Suárez, A., Martínez-Trinidad, J.F., Carrasco-Ochoa, J.A.: A review of conceptual clustering algorithms. Art. Intell. Review **52**(2), 1267–1296 (2019)
19. Pichler, R., Polleres, A., Skritek, S., Woltran, S.: Complexity of redundancy detection on RDF graphs in the presence of rules, constraints, and queries. Semantic Web **4**(4), 351–393 (2013)
20. Shadbolt, N., Hall, W., Berners-Lee, T.: The semantic web revisited. Intelligent Systems, IEEE **21**(3), 96–101 (2006)
21. Soylu, A., Corcho, O., Elvesater, B., Badenes-Olmedo, C., Blount, T., Yedro Martinez, F., Kovacic, M., Posinkovic, M., Makgill, I., Taggart, C., Simperl, E., Lech, T.C., Roman, D.: TheyBuyForYou platform and knowledge graph: Expanding horizons in public procurement with open linked data. Semantic Web **13**(2) (2022)
22. Soylu, A., Elvesæter, B., Turk, P., Roman, D., Corcho, O., Simperl, E., Konstantinidis, G., Lech, T.C.: Towards an ontology for public procurement based on the open contracting data standard. Proc. of 18th IFIP WG 6.11 Conference on e-Business, e-Services, and e-Society, I3E 2019. Springer-Verlag (2019)
23. Vougiouklis, P., Elsahar, H., Kaffee, L.A., Gravier, C., Laforest, F., Hare, J., Simperl, E.: Neural wikipedian: Generating textual summaries from knowledge base triples. Journal of Web Semantics **52-53**, 1–15 (2018)