Data & Knowledge Engineering 133 (2021) 101874

Contents lists available at ScienceDirect

Data & Knowledge Engineering

journal homepage: www.elsevier.com/locate/datak

Multilingual Verbalization and Summarization for Explainable Link Discovery

Abdullah Fathi Ahmed *, Mohamed Ahmed Sherif, Diego Moussallem, Axel-Cyrille Ngonga Ngomo

Data Science Group, Department of Computer Science, Paderborn University, Pohlweg 51, 33098 Paderborn, Germany
Department of Computer Science, University of Leipzig, 04109 Leipzig, Germany

ARTICLE INFO

Keywords:
Link discovery
Verbalization
Link specification
NLP
NLG
Text summarization

ABSTRACT

The number and size of datasets abiding by the Linked Data paradigm increase every day. Discovering links between these datasets is thus central to achieving the vision behind the Data Web. Declarative Link Discovery (LD) frameworks rely on complex Link Specification (LS) to express the conditions under which two resources should be linked. Understanding such LS is not a trivial task for non-expert users. Particularly when such users are interested in generating LS to match their needs. Even if the user applies a machine learning algorithm for the automatic generation of the required LS, the challenge of explaining the resultant LS persists. Hence, providing explainable LS is the key challenge to enable users who are unfamiliar with underlying LS technologies to use them effectively and efficiently. In this paper, we extend our previous work (Ahmed et al., 2019) by proposing a generic multilingual approach that allows verbalization of LS in many languages, i.e., converts LS into understandable natural language text. In this work, we ported our LS verbalization framework into German and Spanish, in addition to English language. Our adequacy and fluency evaluations show that our approach can generate complete and easily understandable natural language descriptions even by lay users. Moreover, we devised an experimental neural approach for improving the quality of our generated texts. Our neural approach achieves promising results in terms of BLEU, METEOR and chrF++.

1. Introduction

With the rapid increase in the number and size of RDF datasets comes the need to link such datasets. Declarative Link Discovery frameworks rely on complex Link Specification to express the conditions necessary for linking resources within these datasets. For instance, state-of-the-art LD frameworks such as Limes [1] and Silk [2] adopt a property-based computation of links between entities. For configuring LD frameworks, the user can either (1) manually enter a LS or (2) use machine learning for automatic generation of LS.

There are a number of machine learning algorithms that can find LS automatically, using either supervised, unsupervised or active learning. For example, the Eagle algorithm [3] is a supervised machine-learning algorithm able to learn LS using genetic programming. In newer work, the Wombat algorithm [4] implements a positive-only learning algorithm for automatic LS finding based on generalization via an upward refinement operator. While LD experts can easily understand the generated LS from such algorithms, and even modify if necessary, most lay users lack the expertise to proficiently interpret those LSs. In addition, so far these

* Corresponding author at: Data Science Group, Department of Computer Science, Paderborn University, Pohlweg 51, 33098 Paderborn, Germany.
E-mail addresses: afaahmed@mail.upb.de (A.F. Ahmed), Mohamed.Sherif@upb.de (M.A. Sherif), diego.moussallem@upb.de (D. Moussallem), Axel.Ngonga@upb.de (A.-C.N. Ngomo).

https://doi.org/10.1016/j.datak.2021.101874
Received 24 February 2020; Received in revised form 23 November 2020; Accepted 14 January 2021
Available online 26 February 2021

```
1  OR(jaccard(x.name,y.name)|0.42,trigrams(x.name,y.description)|0.61)
```

Listing 1: Running example.

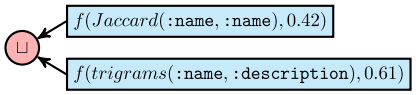

Fig. 1. Our running example complex LS. The filter nodes are rectangles while the operator node is a circle.

algorithms have been unable to explain the LS they generate to lay users. Consequently, these users will face difficulty when they (i) assess the correctness of the generated LS, (ii) adapt their LS, or (iii) choose in an informed manner between possible interpretations of their input. In this paper, we address the readability of LS in terms of natural language. To the best of our knowledge, this is the first work that shows how to verbalize LS and targets many languages such as English, German, and Spanish. As a result, our framework will help people who are unfamiliar with the underlying technology of LS to understand and update it efficiently. The contributions of this paper are as follows:

1. We propose the first (to the best of our knowledge) multilingual *template-based* approach to produce natural text from LS. Our approach is motivated by the pipeline architecture for natural language generation (NLG) systems performed by systems such as those introduced by Reiter & Dale [5]. Our evaluations show that our LS verbalization template-based approach can generate complete and easily understandable natural language descriptions even by lay users.
2. We propose a first version of our *neural-based* LS verbalization approach trained by automatically generated verbalization from our template-based LS approach. We also identify the challenges that the research community needs to address to carry out this task.
3. Finally, we propose a multilingual selectivity-based approach to generate a summarized verbalization of LS.

The rest of this paper is structured as follows: First, we introduce our basic notation in Section 2. Then we give an overview of our template-based approach underlying LS verbalization in Section 3. In Sections 3.1 and 3.2 we explain *document-planner* and the tasks carried out in the *micro-planner*, respectively. In sections 3.3 and 3.4, we explain the German and Spanish Verbalization and the modification carried out. In Section 4 we introduce our neural-based LS verbalization approach. Subsequently, we introduce our Summarization approach in Section 5. We then evaluate our approach with respect to the *adequacy* and *fluency* [6] of the natural language representations it generates in Section 6. After a brief review of related work in Section 7, we conclude our work with some final remarks in Section 8.

Throughout the rest of the paper, we use the LS shown in Listing 1 as our running example. It is generated by the Wombat [4] algorithm to link the ABT-BUY benchmark dataset from [7], where the source resource x will be linked to the target resource y if our running example's LS holds.

2. Preliminaries

In the following, we present the core of the formalization and notation necessary to implement our LS verbalization. We first give an overview of the grammar that underlies LS. Then we describe the notation of LS verbalization.

2.1. Link specification

The link discovery problem is formally defined as follows: Given an input relation ρ (e.g., `owl:sameAs`), a set of source resources S and a set of target resources T, the goal of link discovery is to discover the set $\{(s,t) \in S \times T : \rho(s,t)\}$. Declarative link discovery frameworks define the conditions necessary to generate such links using LS. Several grammars have been used for describing LS in previous work [2,4,8]. In general, these grammars assume that a LS consists of two types of atomic components: *similarity measures* m, which allow the comparison of property values of input resources; and *operators* op that can be used to combine these similarities to more complex specifications. Without loss of generality, we define a similarity measure m as a function $m : S \times T \rightarrow [0,1]$. We use *mappings* $M \subseteq S \times T$ to store the results of the application of a similarity function to $S \times T$ or subsets thereof. We define a *filter* as a function $f(m,\theta)$. We call a specification *atomic LS* when it consists of exactly one filtering function. A complex specification (*complex LS*) can be obtained by combining two specifications L_1 and L_2 through an *operator* op that allows the results of L_1 and L_2 to be merged. Here, we use the operators $\sqcap$, $\sqcup$ and $\setminus$ as they are complete and frequently used to define LS [4]. A graphical representation of our running example's complex LS from Listing 1 is given in Fig. 1.

We define the semantics $[[L]]_M$ of a LS L w.r.t. a mapping M as given in Table 1. The semantics are similar to those used in languages like SPARQL, i.e., they are defined extensionally through the mappings they generate. The mapping $[[L]]$ of a LS L with respect to $S \times T$ contains the links that will be generated by L.

Table 1
Link specification syntax and semantics.

LS	$[[LS]]_M$
$f(m,\theta)$	$\{(s,t)\|(s,t) \in M \wedge m(s,t) \geq \theta\}$
$L_1 \sqcap L_2$	$\{(s,t)\|(s,t) \in [[L_1]]_M \wedge (s,t) \in [[L_2]]_M\}$
$L_1 \sqcup L_2$	$\{(s,t)\|(s,t) \in [[L_1]]_M \vee (s,t) \in [[L_2]]_M\}$
$L_1 \backslash L_2$	$\{(s,t)\|(s,t) \in [[L_1]]_M \wedge (s,t) \notin [[L_2]]_M\}$

Table 2
Dependencies used by LS verbalization.

Dependency	Explanation
`amod`	Represents the *adjectival modifier* dependency. For example, `amod(ROSE,WHITE)` stands for white rose.
`dobj`	Dependency between a verb and its *direct object*. For example, `dobj(EAT,APPLE)` expresses "to eat an/the apple".
`nn`	The *noun compound modifier* is used to modify a head noun by the means of another noun. For instance, `nn(FARMER,JOHN)` stands for farmer John.
`poss`	Expresses a possessive dependency between two lexical items. For example, `poss(JOHN,DOG)` express John's dog.
`prep_X`	Stands for the preposition `X`, where `X` can be any preposition, such as via, of, in and between.
`subj`	Relation between *subject* and verb. For example, `subj(PLAY,JOHN)` expresses John plays.
`coord`	Stands for the relation between a conjunct (many conjuncts) and a given conjunction (in most cases **a**nd or **o**r). For example in the sentence **a**n apple and a pear, `coord(PEAR, APPLE)` holds.

2.2. Link specification verbalization

Our definition of the realization function ζ relies on the formalization of the LS declared in the previous section. Let $\mathcal{A}$ be the set of all *atomic LS* that can be combined in a *complex LS* L. Let C^S respectively C^T be two sets of constraints that specify the sets S respectively. T. Let $\mathcal{M}$ be a set of similarity measures and $\mathcal{T}$ a set of thresholds. In general, a constraint C is a logical predicate. Constraints in LS could state, for example, the `rdf:type` of the elements of the set they describe, i.e., $C(x) \leftrightarrow x\texttt{rdf : typesomeClass}$, or the features that each element in the set must have, e.g., $C(x) \leftrightarrow (\exists y : x\ \texttt{someProperty}\ y)$. Each $s \in S$ must abide by each of the constraints $C_1^S \ldots C_m^S$, while each $t \in T$ must abide by each of the constraints $C_1^T \ldots C_k^T$. We call $z \in \mathcal{A} \cup C^S \cup C^T \cup \mathcal{M} \cup \mathcal{T}$ an *atom*. We define the realization function $\zeta : \mathcal{A} \cup C^S \cup C^T \cup \mathcal{M} \cup \mathcal{T} \rightarrow Language$, where $Language$ is our target language, and it can be English, German or Spanish. In turn, this realization function ζ maps each atom to a word or sequence of words in our target language. Formally, the goals of this paper are twofold: First, construct the extension of ζ to the entire LS so that all *atoms* z can be mapped to their realization $\zeta(x)$. Second, manage how these atomic realizations can be combined. For the sake of simplicity, we denote the extension of ζ by the same label ζ. We adopt a rule-based approach to achieve this goal, where the rule extending ζ to the entire LS is expressed in a conjunctive manner. This means that for premises $P_1, \ldots, P_n$ and consequences $K_1, \ldots, K_m$ we write $P_1 \wedge \cdots \wedge P_n \Rightarrow K_1 \wedge \cdots \wedge K_m$. The premises and consequences are clarified by using an extension of the Stanford dependencies.[1] Notably, we build on the constructs explained in Table 2. For example, dependency between a *verb* and its *object* is represented as $\texttt{dobj}(verb, object)$.

We have now introduced all components necessary for defining our approaches for LS verbalization and summarization. In the next section,we introduce our template-based approach for LS verbalization. We then use our template-based approach to generate *silver standard* verbalization, by which we train our *neural-based* LS verbalization approach. We introduce full details of our neural-based LS verbalization approach in Section 4.

3. Template-based LS verbalization approach

Our goal here is to generate a complete and correct natural language representation of an arbitrary LS. Our template-based LS verbalization approach is motivated by the pipeline architecture for Natural Language Generation (NLG) systems as introduced by Reiter & Dale [5]. The NLG architecture consists of three main stages: *document-planner*, *micro-planner* and *surface realizer*. Since this work is the first step towards the verbalization of LS, our efforts will be focused on *document-planner* (as explained in Section 3.1) with an overview of the tasks carried out in the *micro-planner* (Section 3.2). The *surface realizer* is used to create the output text.

[1] For a complete description of the vocabulary, see http://nlp.stanford.edu/software/dependencies_manual.pdf.

3.1. Document-planner

The *document-planner* consists of the content determination process to create messages and the document structuring process that combines those messages. We focus on document structuring to create independently verbalizable messages from the input LS and to decide on their order and structure. These messages are used for representing information. This part is carried out in the preprocessing and processing steps.

3.1.1. Preprocessing

The goal of the preprocessing step is to extract the central information of LS. This step mainly relies on the *atomic LS* where the necessary information can be extracted. The input for this step is the atomic LS while the output is the realization of each individual part of it. To this end, we break down the atomic LS into its individual parts:

- p_s property of the source resources $s \in S$,
- p_t property of the target resources $t \in T$,
- similarity threshold θ and
- similarity measure m.

The first property p_s comes from dataset S (source) and the second property p_t comes from dataset T (target). For example, $p_s = label$ while $p_t = name$ that means we have a pair of properties (p_s, p_t)=$(label, name)$. After that, on each part of the *atomic LS* we apply the dependency rule introduced in Table 1. We start with the realization of similarity measure m (e.g. $jaccard$ as stated in our running example in Listing 1) as follows:

1. $\zeta(\texttt{m}) \Rightarrow \texttt{nn(m,similarity)}$

Now we can combine ζ(m) and $\zeta(\theta)$.

2. $\zeta(m, \theta) = \zeta(\texttt{m}) \wedge \zeta(\theta) \Rightarrow \texttt{prep\_of}(\zeta(\theta), \zeta(m))$

Furthermore, if θ equals 1, we replace its value by *"exact match"* and in cases where θ is equal to 0, we replace it by *"complete mismatch"*. Otherwise, we keep the θ value (e.g., in the case of our running example). Regarding the properties p_s and p_t, we move the explanation into the processing step since they play an important role in the construction of a *subject* to be used later in sentence building.

3.1.2. Processing

In this step, we aim to map all *atoms* z into their realization function ζ(z) and to define how these atomic realizations are to be combined. The input for this step is the LS and the output is the verbalization of the LS at hand. Given our formalization of LS in Section 2.1, any LS is a binary tree, where the root of the tree is an operator op and each of its two branches are LSs. Therefore, we recursively in-order apply our processing step at the LS tree at hand. As the complete verbalization of an atomic LS mainly depends on the properties p_s and p_t, we distinguish two cases here: the *first case* where p_s and p_t are equal, so we only need to verbalize p_s. In this case, the realization function of an atomic LS $a \in \mathcal{A}$ is constructed as follows:

3. $\zeta(\texttt{a}) \Rightarrow \texttt{subj(have,nn(prep\_of}(\zeta(p_s), \zeta(\texttt{source and target})), \zeta(\texttt{resources}))) \wedge \texttt{dobj(have,}\ \zeta(m, \theta))$

The *second case* is where p_s and p_t are not equal. Here, both properties need to be verbalized as follows:

4. $\zeta(p_s, p_t) \Rightarrow \zeta(p_s) \wedge \zeta(p_t)$

3.2. Micro-planer

The micro-planner is divided into three processes: *lexicalization, referring expression generation*, and *aggregation*. We explain each process in the following.

3.2.1. Lexicalization

Within the lexicalization process, we decide what specific words should be used to express the content. In particular, we choose the actual nouns, verbs, adjectives, and adverbs to appear in the text from a lexicon. Also, we decide which particular syntactic structures to use, for example, whether to use the phrase `the name of the resource` or **resource's name**.

5. $\zeta(p_s) \Rightarrow \texttt{prep\_of(poss}(\zeta(\texttt{resource}), p_s), \zeta(\texttt{source}))$
6. $\zeta(p_t) \Rightarrow \texttt{prep\_of(poss}(\zeta(\texttt{resource}), p_t), \zeta(\texttt{target}))$
7. $\zeta(\texttt{a}) \Rightarrow \texttt{subj(have,}\ \zeta(p_s, p_t)) \wedge \texttt{dobj(have,}\ \zeta(m, \theta))$

Applying *preprocessing* and *processing* steps followed by the *Lexicalization* step on our running example from Listing 1 generates the following verbalization: `The name of source and target resources has a 42% of Jaccard similarity or the resource's name of the source and the resource's description of the target have a 61% of Trigrams similarity.` Note that our running example contains both cases.

```
1 OR(jaccard(x.name,y.name)|0.42,qgrams(x.name,y.name)|0.61)
```

Listing 2: Grouping example.

3.2.2. Referring expression generation

Here we carry out the task of deciding which expressions should be used to refer to entities. Considering the example, `the source and the target have a resource's name and they have a 45% of Jaccard similarity`, `they` is referring to the expression `the source and the target`. However, we avoid such a construction in our verbalization because we aim to generate a simple yet readable text that contains the central information of the LS at hand.

3.2.3. Aggregation

The goal of aggregation in NLG is to avoid duplicating information that has already been presented. In our LS verbalization, we mainly focus on the *subject collapsing*, defined in [9] as the process of *"collecting clauses with common elements and then collapsing the common elements"*. Formally, we define *subject* $\texttt{subj}(v_i, s_i)$ as s_i, *object* $\texttt{dobj}(v_i, o_i)$ as o_i

8. $\zeta(s_1)= \zeta(s_2)=\ldots=\zeta(s_n) \Rightarrow \texttt{subj}(v_1, s_1) \wedge \texttt{dobj}(v_1, \texttt{coord}(o_1, o_2,\ldots, o_n))$

In Listing 2, we present a second example LS where grouping is applicable.

The original verbalization of LS from Listing 2 is: `The name of source and target resources has a 42% of Jaccard similarity or the name of source and target resources has a 61% of Qgrams similarity.` And after applying grouping, our verbalization will become more compact as follows: `The name of source and target resources has a 42% of Jaccard similarity or a 61% of Qgrams similarity.`

3.3. German verbalization

In regard to the German language, we make use of the genitive case instead of using possessive case since it is widely used in the German language. For instance, the words ``Name'' and ``Datenquelle'' will be verbalized into ``Namens der Datenquelle''. We generated German text using the implementation of [10].

Considering the complexity of the German language, our running example from Listing 1 will be converted into ``Name von der Datenquelle und dem Datenziel Ressourcen hat 42% einer Jaccards Ähnlichkeit oder die Ressource Namens der Datenquelle und die Ressource Descriptions des Datenziels haben 61% einer Trigramss Ähnlichkeit''. The same LS can be verblized in German as follows: ``der Link passiert, wenn Name von der Datenquelle und dem Datenziel Ressourcen 42% einer Jaccards Ähnlichkeit hat oder der Link passiert, wenn die Ressource Namens der Datenquelle und die Ressource Descriptions des Datenziels 61% einer Trigramss Ähnlichkeit haben''. This is due to the fact that there are many possible word orders for the same sentence in German. However, the position of the verb should be changed according to the way of ordering the words. For instance, in the first verbalization the verb ``hat'' in the middle of the sentence, while in the second verbalization after changing the order of words, the verb ``hat'' moved to the end of the sentence. From a surface realization perspective, many inflection rules make the German language more complex than English. While table look-ups for inflected forms can be performed reasonably in the English language, it is not feasible for German. The German language has ``ein'' and ``eine'' as indefinite articles and ``das, der'' and ``die'' as definite articles, whereas in the English language `the` as definite article and `a/an` as indefinite articles satisfy. Accordingly, all articles and pronouns must be inflected according to gender, number, person and grammatical case (nominative, genitive, dative, accusative) resulting in more article forms, for instance for indefinite articles in ``einen'', ``einem'',``einer'', and ``eines''. E.g., ``die Ressource Descriptions des Datenziels 61% einer Trigramss Ähnlichkeit'', where indefinite article ``eine'' changed to ``einer'' obeying the grammar of genitive case. This can be seen for definite articles as well. For example, ``die Ressource Descriptions des Datenziels'' is inflection for the noun ``das Datenziel'' in genitive case (i.e. ``... des Datenziels'')

3.4. Spanish verbalization

For the verbalization in the Spanish language, we rely on the implementation proposed in [11], in which the authors extended the bilingual English–French SimpleNLG-EnFr realizer. The Spanish realizer is named SimpleNLG-ES. In SimpleNLG-EnFr, English and French share most of the basic framework, such as document elements and some grammar rules which are common for both English and Spanish. The Spanish language is not rich in terms of morphology rules compare to other close languages. There are two main contractions between the prepositions ``a'' and ``de'', and the masculine singular determinant ``el'' that are always used as follows: ``a el'' → ``al'' meaning `to the`, and the second case is ``de el'' → ``del'' meaning `of the`. As a result, SimpleNLG-ES provides only support for ``al'' and ``del'' .

The verbalization in Spanish was adapted in a straightforward manner. In our running example in Listing 1, the Spanish verbalization was generated as follows: ``El enlace será generado se La propiedad ``name'' de la fuente y los recursos de destino tiene un 42% de Jaccard similitud o el recurso name de la fuente y el recurso description de destino tiene un 61% de Trigrams similitud''.

4. Neural-based LS verbalization approach

To improve our verbalization module quality, we devised a neural approach that relies on standard sequence-to-sequence [12] models for generating text. It is well known that neural models require a considerable amount of training data to achieve good performance [13]. However, to the best of our knowledge, our work is the pioneer in generating verbalization, i.e., explanations in natural language from link specifications data. Therefore, there is a lack of training data for this task. Thus, our neural approach's intuition was to train the neural network on the texts generated by our template-based approach (Section 3). Using our template-based approach, we create parallel training data, the source is composed of the link specifications, and the target refers to the natural language sentences. Every link specification is aligned to one natural language sentence. We generated 35,000 parallel data. Our neural architecture is a bidirectional RNN-LSTM 2-layer encoder–decoder model with an attention mechanism [14]. The training uses a batch size of 32 and the stochastic gradient descent with an initial learning rate of 0.0002. We set a source and target word embeddings size of 500, and hidden layers to size 500, dropout = 0.3 (naïve). We used a maximum sentence length of 80, a vocabulary of 50,000 words, and a beam size of 5.

5. Selectivity-based LS summarization approach

We define the *selectivity score* of a sub-LS $L_s \in L$ as a function $\sigma(L)$ that returns the F-Measure achieved by the mapping $[[L_s]]$ of L_s by considering the mapping $[[L]]$ generated by the original LS L as its reference mapping.

We propose a sentence-scoring-based LS summarization approach. The basic idea behind our summarization approach is to simplify the original LS tree by pruning LS sub-trees that achieve the minimum *selectivity score*, i.e., keep the information loss minimum. Given an input LS L_i, our summarization approach first generates an ordered list $\mathbf{L}$ of simplified LSs of L_i, where $\mathbf{L}$ is ordered by the selective score of each of its elements in descending order. This step is carried out by iteratively pruning the sub-tree of L_i with the minimum selectivity score.

In cases where a summarization threshold $\tau \in [0, 1]$ is given, the output of our summarization algorithm will be generated by applying our LS verbalization approach to the LS $L \in \mathbf{L}$ with the highest selectivity score $\sigma(L) \leq \tau$. Otherwise, the output of our summarization approach will be a list of the verbalization of the whole list $\mathbf{L}$. For instance, assume we have the LS `OR(jaccard(x.name,y.name)|0.45, qgrams(x.name,y.name)|0.67)`, from which our summarization approach generates the ordered list $\mathbf{L} = \{L_1, L_2\}$, where L_1 = `(jaccard(x.name,y.name)|0.45)` and L_2 = `(qgrams(x.name, y.name)|0.67)`. Accordingly, we compute the scores of $\sigma(L_1) = 0.8$ and $\sigma(L_2) = 0.6$. Assume we have the summarization threshold $\tau = 0.9$. Our approach will thus verbalize the link specification L_1 with highest selectivity score $\sigma(L) \leq \tau$.

6. Evaluation

We evaluated our approaches for LS verbalization and summarization in order to elucidate the following questions:

- Q_1: Does the LS verbalization help the user to better understand the conditions sufficient to link the resources in comparison to the original LS?
- Q_2: How fluent is the generated LS verbalization? I.e., how good is the natural language description of the LS verbalization in terms of comprehensibility and readability?
- Q_3: How adequate is the generated LS verbalization? I.e., how well does the verbalization capture the meaning of the underlying LS?
- Q_4: How much information do we lose by applying our summarization approach?

6.1. Experimental setup

To answer the first three questions, we conducted a *user study* to evaluate our LS verbalization. We used our approach to verbalize a set of five LSs automatically generated by the Eagle algorithm [8] for the benchmark datasets of `Amazon-GP`, `ABT-BUY`, `DBLP-ACM`, and `DBLP-Scholar` from [7]. Our user study consists of four tasks, and each task consists of five multiple choice questions.[2] Altogether, we have a group of 18 participants in our user study from the DICE[3] and AKSW[4] research groups. In the following, we explain each task:

Task 1: This task consists of five identical sub-tasks. For each, we present the survey participant a LS and three pairs of source and target resources represented by their respective concise bounded descriptions (CBD)[5] graph. These pairs are matched based on the provided LS with different degrees of confidence. To this end, the participant is asked to find the best matched pair, and we measure the response time for each participant.

Task 2: This task also consists of five identical sub-tasks. We again follow the same process in *Task 1* of presenting the participant with the CBDs of matched resources, but this time we give the survey participant the verbalization of the LSs. Again, we record the response time of each participant.

[2] The survey interface for English can be accessed at https://umfragen.uni-paderborn.de/index.php/186916?lang=en.

[3] https://dice-research.org/.

[4] http://aksw.org/About.html.

[5] https://www.w3.org/Submission/CBD/.

Task 3: Within this task, a survey participant is asked to judge the fluency of the provided verbalization. Here, we follow the machine translation standard introduced in [6]. Fluency captures how good the natural language description is, in terms of comprehensibility and readability, according to the following six ratings: (6) Perfectly clear and natural; (5) Sounds a bit artificial, but is clearly comprehensible (may contain minor grammatical flaws); (4) Sounds very artificial, but is understandable (although may contain significant grammatical flaws); (3) Barely comprehensible, but can be understood with some effort; (2) Only a loose and incomplete understanding of the meaning can be obtained, and (1) Completely incomprehensible.

Task 4: In this task, we provide a survey participant with a LS and its verbalization. They are then asked to judge the adequacy of the verbalization. Here we follow the machine translation standard from [6]. Adequacy addresses how well the verbalization captures the meaning of the LS, according to the following six ratings: (6) Perfect; (5) Mostly correct, although maybe some expressions do not match the concepts very well; (4) Close, but some information is missing or incorrect; (3) There is significant information missing or incorrect; (2) Natural Language (NL) description and LS are only loosely connected; and (1) NL description and LS are in no conceivable way related.

For answering the last question, we conducted an experiment on the benchmark datasets from [7]. We ran the supervised version of the Wombat algorithm to generate an automatic LS for each dataset. We again used [4] to configure Wombat. Afterwards, we applied our summarization algorithm to each of the generated LSs. Because of the space limitation, we present only the verbalization of the original LS (the ones generated by Wombat) as well as the first summarization of it for the `Amazon-GP` and `DBLP-Scholar` datasets in Table 3. The complete results are available on the project website.[6] For evaluating the verbalization of German, we reformulate the same 4 *Tasks* but the participants were only 8 participants. The survey can be accessed via.[7] To evaluate our verbalization approach for Spanish, we conducted a survey similar to the ones created for English and German; however, we have only two experts in Link Discovery who can speak Spanish.

6.2. Results and discussion of English language verbalization

After collecting all the responses from our user study, we filtered out those survey participants who were unlikely to have thoroughly executed the survey (i.e., the ones who took notably less time than the average response time of all other participants) or who were likely distracted while executing it (i.e., the ones who took notably more time than the average time of all other participants). This process reduced the number of valid participants to 16. Our final accepted time window was $3.5 - 38$ minutes for *Tasks 1* & *2*. Accordingly, we start our evaluation by comparing the user time required to find the best matched source–target pair using LS (*Task 1*) against using the verbalization of the provided LS (*Task 2*).

As shown in Fig. 2, the average user response time with LS verbalization is less than the response times for LS in the 5 LSs in our user study. On average, using verbalization is 36% faster than using LS. Additionally, we also compared the error rates of participants in *Tasks 1* & *2*, i.e. the number of incorrect answers per question. As shown in Fig. 3, we have a higher error rate using verbalization (5% mean squared error) than when using LS. These results show that using LS verbalization decreases the average response time, which is an indicator that our participants were able to better understand underlying LS using verbalization. Still, using the LS verbalization does not always lead our participants to select the correct answer. This is due to the complexity involved in the underlying LSs, which leads to verbalization that is too long. This answers Q_1. Using our simplification approach on the same LS verbalization leads our participants to achieve better results.

The results of *Task 3* (see Fig. 4) show that the majority of the generated verbalizations (i.e., the natural language descriptions) were fluent. In particular, 87% of the cases achieved a rating of 3 or higher. On average, the fluency of the natural language descriptions is 4.7 ± 0.4. This answers Q_2.

For *Task 4*, the average adequacy rating of our verbalization was 4.75 ± 0.6 (see Fig. 5), which we consider to be a positive result. In particular, 40% of all verbalizations were judged to be perfectly adequate and 83% of the cases achieved a rating of 3 or higher. This answers Q_3.

As we can see in Table 3, applying our summarization approach reduces the verbalization of the original LS to more that half of its original size. At most, our summarization approach loses an F-Measure of 12% of the original description, which we consider a fair price given the high summarization rate. This clearly answers our last question.

6.3. Results and discussion of German language verbalization

As shown in Fig. 6, the average user response time with LS verbalization is less than the ones for LS in 80% of the LSs used in our users study. On average, using verbalization is 16% faster than using LS. In addition, we measured the error rates of participants in *Tasks 1* & *2*, i.e. the number of incorrect answers per question. Since there are fewer participants, we did not filter out any survey participants, for instance those who are unlikely to have executed the survey in a thorough manner (i.e., the ones who might take clearly less time than the average response time of all other participants) or who are likely distracted while running it (i.e., the ones who might spend notably more time than the average time of all other participants). As shown in Fig. 7, using verbalization we have a higher error rate (1.6% mean squared error) than when using LS. These results indicate that our participants were able to

[6] https://bit.ly/2XKDpKZ.

[7] The survey for German language verbalizationhttps://umfragen.uni-paderborn.de/index.php/288119?lang=en.

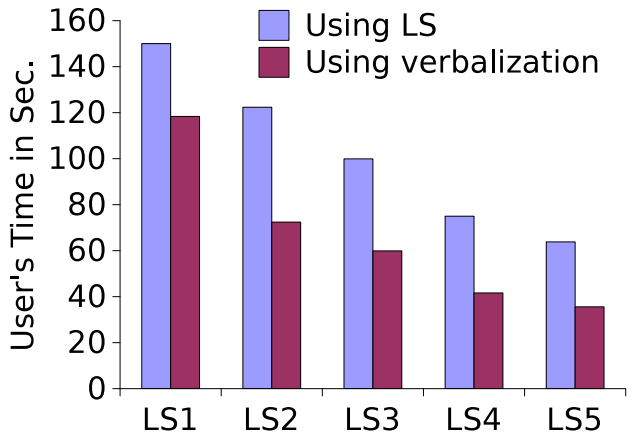

Fig. 2. Average response time of our user study for English language.

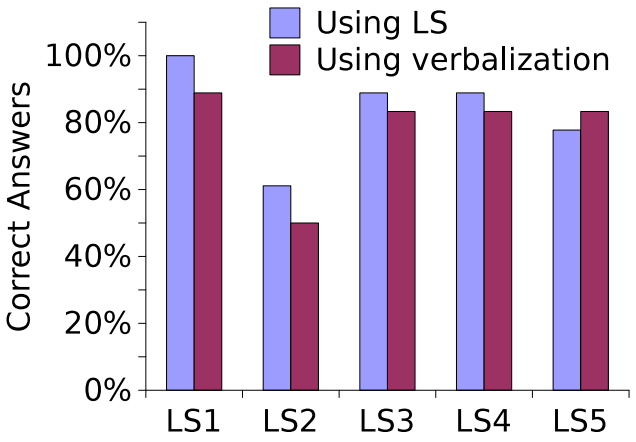

Fig. 3. Correct answers of our user study for English language.

Table 3
Verbalization of different summarization of a LS for the DBLP-SCHOLAR and Amazon-GP dataset with respective **F**-measures in the English language.

Dataset	**F**	Verbalization
DBLP-SCHOLAR	1	The link will be generated if the title of the source and the target resources has a 66% of Cosine similarity or the resource's title of the source and the resource's author of the target has a 43% of Jaccard similarity or the resource's author of the source and the resource's title of the target has a 43% of Trigram similarity
DBLP-SCHOLAR	0.88	The link will be generated if the title of the source and the target resources has a 66% of Cosine similarity
Amazon-GP	1	The link will be generated if the resource's title of the source and the resource's name of the target has a 48% of Cosine similarity or the description of the source and the target resources has a 43% of Cosine similarity or the resource's title of the source and the resource's description of the target has a 43% of Jaccard similarity
Amazon-GP	0.97	The link will be generated if the resource's title of the source and the resource's name of the target has a 48% of Cosine similarity

better understand underlying LS using verbalization. However, using the LS verbalization does not always lead our participants to select the correct answer. This is due to the fact that the underlying LSs are very complex, which makes the verbalization too long. This answers Q_1.

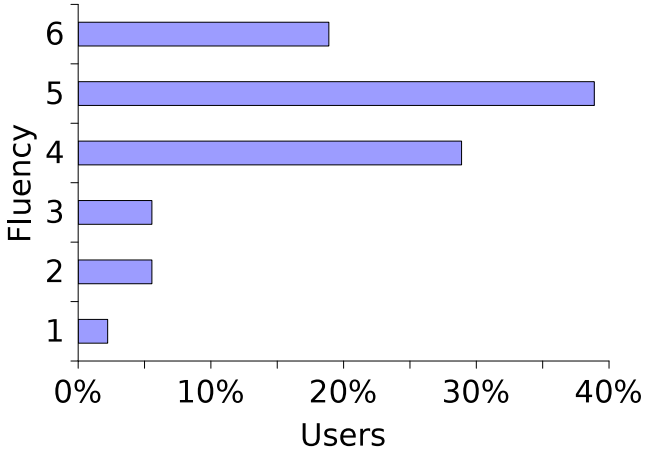

Fig. 4. Fluency results for the English language.

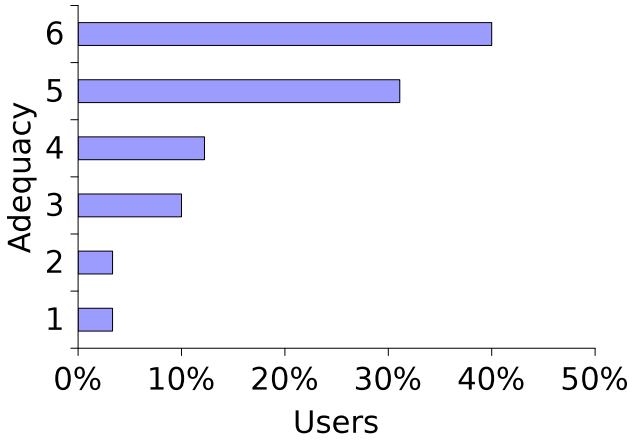

Fig. 5. Adequacy results for the English language.

For *Task 3* (see Fig. 8), the results show that the bulk of the generated verbalizations (i.e., the natural language text) was fluent. In particular, 90% of the cases achieved a rating of 3 or higher. The average fluency of the natural language descriptions is 4.3 ± 0.06. This answers Q_2.

The results of *Task 4* show that the average adequacy rating of our verbalization is 4.8 ± 0.52 (see Fig. 9), which we recognize as a positive result. Precisely, 42.5% of all verbalizations were judged to be perfectly adequate and 92.5% of the cases achieved a rating of 3 or higher. This answers Q_3. In general, the verbalization for the German language is more difficult compared to English or Spanish because of the natural difficulty of the German language itself; however our approach is able to generate text with an average adequacy rating of 4.8 ± 0.52 and the average of the fluency is 4.3 ± 0.06.

To answer the last question Q_4, in Table 4 we can see that applying our summarization approach on the German language that can reduce the verbalization of the original LS to more than half of its original size. An F-Measure of 12% of the original description which is at the biggest loses of our summarization approach. We consider a fair cost given the high summarization rate. Table 5 shows the results of summarization applied to the Spanish language. From our summarization results, we can conclude that our summarization approach works independently from the language.

6.4. Results and discussion of neural-based verbalization

As no previous work (to the best of our knowledge) investigated the generation of natural language from link specifications, our goal was to identify the challenges for this new line of research. We reckon that the training data is a *silver standard* because it was not peer-reviewed manually before creating the test data. However, it provided many insights and challenges that the research community needs to address this task. Table 6 displays the results of our model in the automatic evaluation standard metrics, BLEU, METEOR, and chrf++.

The results of the automatic metrics show that our model was capable of generating fluent texts. However, we looked more in depth at the generated texts and investigated the sentences manually. We perceived that some metrics such as `JaroWinkler` and

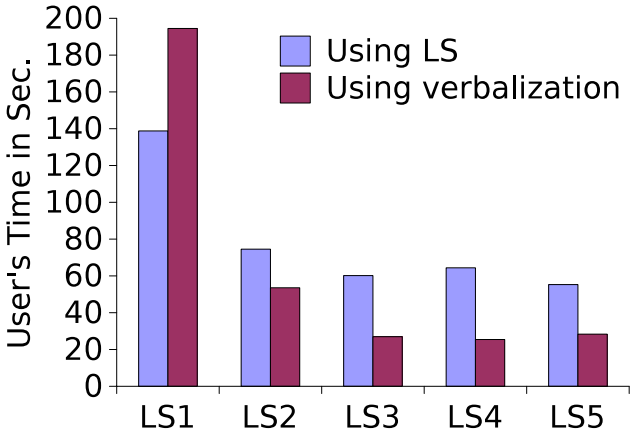

Fig. 6. Average response time of our user study for the German language.

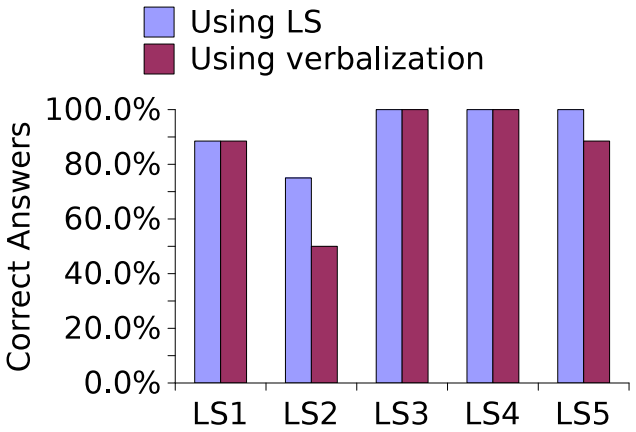

Fig. 7. Correct answers of our user study for the German language.

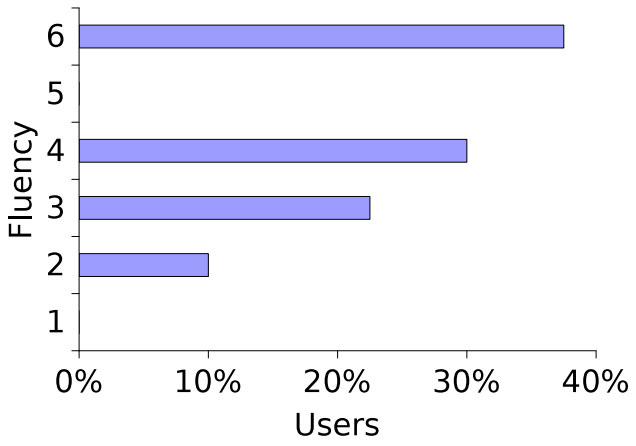

Fig. 8. Fluency results for the German language.

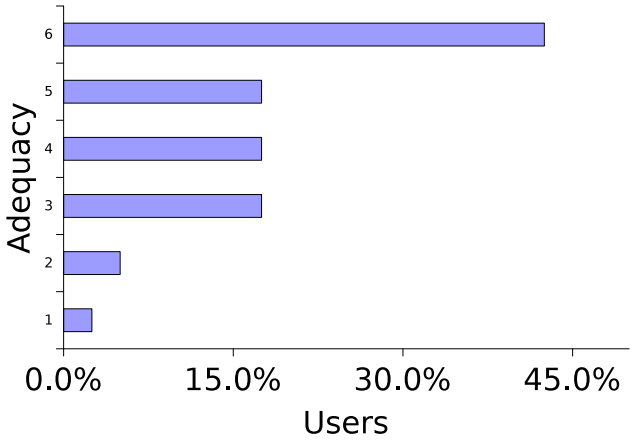

Fig. 9. Adequacy results for German language.

Table 4
Summarization of a LS for the DBLP-SCHOLAR and Amazon-GP datasets together with respective **F**-measures in German language.

Dataset	**F**	Verbalization
DBLP-SCHOLAR	1	"Der Link passiert, wenn Title von der Datenquelle und dem Datenziel Ressourcen 66% einer Cosines Ähnlichkeit hat oder der Link passiert, wenn die Ressource Titles der Datenquelle und die Ressource Authorss des Datenziels 43% einer Jaccards Ähnlichkeit haben oder der Link passiert, wenn die Ressource Authorss der Datenquelle und die Ressource Titles des Datenziels 43% einer Trigrams Ähnlichkeit hat"
DBLP-SCHOLAR	0.88	"Der Link passiert, wenn Title von der Datenquelle und dem Datenziel Ressourcen 66% einer Cosines Ähnlichkeit hat"
Amazon-GP	1	Der Link passiert, wenn die Ressource Titles der Datenquelle und die Ressource Namens des Datenziels 48% einer Cosines Ähnlichkeit haben oder der Link passiert, wenn Description von der Datenquelle und dem Datenziel Ressourcen 43% einer Cosines Ähnlichkeit hat oder der Link passiert, wenn die Ressource Titles der Datenquelle und die Ressource Descriptions des Datenziels 43% einer Jaccards Ähnlichkeit hat
Amazon-GP	0.97	Der Link passiert, wenn die Ressource Titles der Datenquelle und die Ressource Namens des Datenziels 48% einer Cosines Ähnlichkeit haben

Table 5
Summarization of a LS for the DBLP-SCHOLAR and Amazon-GP datasets together with respective **F**-measures in the Spanish language.

Dataset	**F**	Verbalization
DBLP-SCHOLAR	1	"El enlace será generado se La propiedad title de la fuente y los recursos de destino tiene un 66% de Cosine similitud o el recurso title de la fuente y el recurso authors de destino tiene un 43% de Jaccard similitud o el recurso authors de la fuente y el recurso title de destino tiene un 43% de Trigram similitud"
DBLP-SCHOLAR	0.88	"El enlace será generado se La propiedad title de la fuente y los recursos de destino tiene un 66% de Cosine similitud"
Amazon-GP	1	El enlace será generado se el recurso title de la fuente y el recurso name de destino tiene un 48% de Cosine similitud o La propiedad description de la fuente y los recursos de destino tiene un 43% de Cosine similitud o el recurso title de la fuente y el recurso description de destino tiene un 43% de Jaccard similitud
Amazon-GP	0.97	El enlace será generado se el recurso title de la fuente y el recurso name de destino tiene un 48% de Cosine similitud

Table 6
Results of BLEU, METEOR and chrf++.

Model	BLEU	METEOR	chrf++
LSNeural	32.05	23.41	43.38

`Ngram` were rarely verbalized, while `Jaccard` and `Levenstein` were generated frequently as they appear several times on the training data. This shows the need to generate diverse training data with a balanced distribution of tokens for training the neural models.

Additionally, the target side's vocabulary was small, with only 48 different tokens, while the source side, which is regarded as the link specification, contains 10,000 different tokens. Therefore, we envisage that the text generation for link specification is regarded as a *low-resource translation problem*. Thus we can apply well-known Machine Translation (MT) strategies for dealing with the data sparsity problem in future work such as *Byte-per-encoding* (BPE) [15] and *copy-mechanism* [16]. Copy-mechanism tries to substitute the Out-of-vocabulary (OOV) words with target words that have the highest attention weight according to their source words. When the words are not found, it copies the source words to the position of the not-found target word. BPE is a form of data compression that iteratively replaces the most frequent pair of bytes in a sequence with a single, unused byte.

7. Related work

While we believe that this is the first work that shows how to verbalize LS, related work comes from three research areas: declarative link discovery approaches, verbalization of Semantic data, and text summarization.

Declarative Link Discovery frameworks rely on complex LS to express the conditions necessary for linking resources within RDF datasets. For instance, state-of-the-art LD frameworks, such as Limes [1] and Silk [2], adopt a property-based computation of links between entities. All such frameworks enable their users to manually write LS and execute it against source–target resources. In recent years, the problem of using machine learning for the automatic generation of accurate LS has been addressed by most of the link discovery frameworks. For example, the Silk framework [2] implements a batch learning approach for the discovery LS based on genetic programming, which is similar to the approach presented in [17]. For the Limes framework, the Raven algorithm [18] is an active learning approach that treats the discovery of specifications as a classification problem. In RAVEN, the discovery of LS is done by first finding class and property mappings between knowledge bases automatically. It then uses these mappings to compute linear and boolean classifiers that can be used as LS. Eagle [8] has addressed the readability of LS alongside accuracy and efficiency. However, the generated LS is still expressed in a declarative manner. Recently, the Wombat algorithm [4] has implemented a machine leaning algorithm for automatic LS finding by using generalization via an upward refinement operator.

With the recent demand on new explainable machine learning approaches comes the need for *the verbalization of semantic data* involved within such approaches. For example, [19] expands on an approach for converting RDF triples into Polish. The authors of [20] espouse a reliance on the Linked Data Web being created by reversing engineered structured data into natural language. In their work [21], the same authors show how this approach can be used to produce text out of RDF triples. Yet another work, [22], generated natural language out of RDF by depending on the BOA framework [23,24] to compute the trustworthiness of RDF triples using the Web as background knowledge. Other approaches and concepts for verbalizing RDF include [25] and [26]. Moreover, approaches to verbalizing first-order logics [27] are currently being devised. In very recent work [28], the authors have addressed the limitations of adapting rule-based approaches to generate text from semantic data by proposing a statistical model for NLG using neural networks.

The second part of our approach is the summarization of LS, which is related to work in the area of *text summarization* with a focus on sentence scoring techniques. The work [29] surveys many sentence scoring techniques. Furthermore, the survey [30] addresses many text summarization methods. However, in our summarization technique the summarization score is user-defined.

8. Conclusions and future work

In this paper, we extend our previous work [31] by introducing a multilingual template-based approach for verbalizing LS. Our approach produces both a direct literal verbalization of the content of the LS and a more natural aggregated version of the same content. We presented the key steps of our approach and evaluated it with a user study in the English, Spanish and German languages. Our evaluation shows that the verbalization generated by our approach is both complete and easily understandable. Our approach not only accelerates the understanding of LS by expert users but also enables non-expert users to understand the content of LS. However, our evaluation shows that when the LS is more complex and contains different operators, the fluency of our approach decreases. We also introduce the first version of a neural-based LS verbalization approach trained by automatically generated verbalization from our template-based LS approach. In addition, we summarize the produced multilingual verbalization using selectivity-based LS summarization approach.

In future work, we will improve upon our verbalization in such a way that further increases the fluency without altering the rule underlying LS. Moreover, we will devise a consistency checking algorithm to improve the correctness of the natural language generated by our approach. We will also extend our current neural network approach by addressing the challenges that we identified in our current version. By addressing such challenges, we envisage that the quality of the generated texts for both verbalization and summarization will improve substantially.

CRediT authorship contribution statement

Abdullah Fathi Ahmed: Conceptualization, Methodology, Software, Data curation, Writing - original draft, Visualization, Investigation, Validation, Writing - review & editing. **Mohamed Ahmed Sherif:** Conceptualization, Methodology, Software, Data curation, Writing - original draft, Visualization, Investigation, Supervision, Validation, Writing - review & editing, Project administration, Funding acquisition. **Diego Moussallem:** Conceptualization, Methodology, Software, Data curation, Writing - original draft, Visualization, Investigation, Validation, Writing - review & editing. **Axel-Cyrille Ngonga Ngomo:** Supervision, Project administration, Funding acquisition.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work has been supported by the German Federal Ministry of Economics and Technology (BMWI) project RAKI (no. 01MD19012D), the EU H2020 project KnowGraphs (GA no. 860801) as well as the BMVI projects LIMBO, Germany (GA no. 19F2029C) and OPAL, Germany (GA no. 19F2028A).

References

[1] A.-C.N. Ngomo, S. Auer, Limes - A time-efficient approach for large-scale link discovery on the web of data, in: IJCAI, 2011.
[2] R. Isele, A. Jentzsch, C. Bizer, Efficient multidimensional blocking for link discovery without losing recall, in: WebDB, 2011.
[3] A.-C. Ngonga Ngomo, K. Lyko, EAGLE: Efficient active learning of link specifications using genetic programming, in: E. Simperl, P. Cimiano, A. Polleres, O. Corcho, V. Presutti (Eds.), The Semantic Web: Research and Applications, Springer Berlin Heidelberg, Berlin, Heidelberg, 2012, pp. 149–163.
[4] M. Sherif, A.-C. Ngonga Ngomo, J. Lehmann, WOMBAT - A Generalization Approach for Automatic Link Discovery, Springer, 2017.
[5] E. Reiter, R. Dale, Building Natural Language Generation Systems, Cambridge University Press, New York, NY, USA, 2000.
[6] G. Doddington, Automatic evaluation of machine translation quality using n-gram co-occurrence statistics, in: Proceedings of HLT, 2002, pp. 138–145.
[7] H. Köpcke, A. Thor, E. Rahm, Comparative evaluation of entity resolution approaches with fever, Proc. VLDB Endow. 2 (2) (2009) 1574–1577.
[8] A.-C. Ngonga Ngomo, K. Lyko, EAGLE: Efficient Active Learning of Link Specifications Using Genetic Programming, Springer Berlin Heidelberg, 2012.
[9] H. Dalianis, E. Hovy, Aggregation in natural language generation, in: G. Adorni, M. Zock (Eds.), Trends in Natural Language Generation an Artificial Intelligence Perspective, Springer Berlin Heidelberg, Berlin, Heidelberg, 1996, pp. 88–105.
[10] D. Braun, K. Klimt, D. Schneider, F. Matthes, Simplenlg-de: Adapting simplenlg 4 to german, in: Proceedings of the 12th International Conference on Natural Language Generation, Association for Computational Linguistics, Tokio, Japan, 2019.
[11] A. Ramos-Soto, J. Janeiro-Gallardo, A. Bugarín, Adapting simplenlg to spanish, in: 10th International Conference on Natural Language Generation, Association for Computational Linguistics, 2017, pp. 142–146.
[12] D. Bahdanau, K. Cho, Y. Bengio, Neural machine translation by jointly learning to align and translate, in: 3rd International Conference on Learning Representations, ICLR, 2015.
[13] A.F. Ahmed, M.A. Sherif, A.-C.N. Ngomo, Lsvs: Link specification verbalization and summarization, in: International Conference on Applications of Natural Language To Information Systems, Springer, 2019, pp. 66–78.
[14] D. Moussallem, A.-C. Ngonga Ngomo, P. Buitelaar, M. Arcan, Utilizing knowledge graphs for neural machine translation augmentation, in: Proceedings of the 10th International Conference on Knowledge Capture, 2019, pp. 139–146.
[15] R. Sennrich, B. Haddow, A. Birch, Neural machine translation of rare words with subword units, in: Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), 2016, pp. 1715–1725.
[16] M.-T. Luong, H. Pham, C.D. Manning, Effective approaches to attention-based neural machine translation, in: Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, 2015, pp. 1412–1421.
[17] M.G. Carvalho, A.H.F. Laender, M.A. Gonçalves, A.S. da Silva, Replica Identification Using Genetic Programming, ACM, 2008.
[18] A.-C. Ngonga Ngomo, J. Lehmann, S. Auer, K. Höffner, Raven – Active learning of link specifications, in: Proceedings of OM@ISWC, 2011.
[19] A. Pohl, The polish interface for linked open data, in: Proceedings of the ISWC 2010 Posters & Demonstrations Track, 2011, pp. 165–168.
[20] C. Mellish, X. Sun, The semantic web as a linguistic resource: opportunities for natural language generation, 2006.
[21] X. Sun, C. Mellish, An experiment on "Free Generation" from Single RDF Triples, Association for Computational Linguistics, 2007.
[22] J. Lehmann, D. Gerber, M. Morsey, A.-C. Ngonga Ngomo, Defacto - deep fact validation, in: ISWC, 2012.
[23] D. Gerber, A.-C. Ngonga Ngomo, Bootstrapping the linked data web, in: 1st Workshop on Web Scale Knowledge Extraction @ ISWC 2011, 2011.
[24] D. Gerber, A.-C.N. Ngomo, Extracting multilingual natural-language patterns for RDF predicates, in: EKAW, 2012, pp. 87–96.
[25] H. Piccinini, M.A. Casanova, A.L. Furtado, B.P. Nunes, Verbalization of rdf triples with applications, in: ISWC - Outrageous Ideas Track, 2011.
[26] G. Wilcock, K. Jokinen, Generating responses and explanations from RDF/XML and DAML+OIL, 2003, http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.14.4745.
[27] N.E. Fuchs, First-order reasoning for attempto controlled english, 2010.
[28] P. Vougiouklis, H. Elsahar, L.-A. Kaffee, C. Gravier, F. Laforest, J. Hare, E. Simperl, Neural wikipedian: Generating textual summaries from knowledge base triples, J. Web Semant. 52–53 (2018) 1–15, http://dx.doi.org/10.1016/j.websem.2018.07.002.
[29] Assessing sentence scoring techniques for extractive text summarization, Expert Syst. Appl. (2013).
[30] M. Allahyari, S.A. Pouriyeh, M. Assefi, S. Safaei, E.D. Trippe, J.B. Gutierrez, K. Kochut, Text summarization techniques: A brief survey, CoRR (2017).
[31] A.F. Ahmed, M.A. Sherif, A.-C.N. Ngomo, Lsvs: Link specification verbalization and summarization, in: E. Métais, F. Meziane, S. Vadera, V. Sugumaran, M. Saraee (Eds.), Natural Language Processing and Information Systems, Springer International Publishing, Cham, 2019, pp. 66–78.

Abdullah Fathi Ahmed (m), *Homepage:* https://dice-research.org/AbdullahFathiAhmed. Abdullah Fathi Ahmed is a Ph.D. researcher at the Data Science research group (DICE), Computer Science, Paderborn University. His research interests are in the area of Semantic Web technologies with the main focus on data integration and link discovery. It also includes the AI explainability of link discovery using natural language processing techniques. He earned his B.Sc. in Computer Engineering from the University of Mosul (Iraq), then M.Sc. in Computer Engineering from Paderborn University.

Dr. Mohamed Ahmed Sherif (m), *Homepage:* https://dice-research.org/MohamedAhmedSherif. Mohamed Ahmed Sherif is a postdoctoral researcher at the Data Science research group (DICE), at University of Paderborn. Mohamed's main research fields is applying machine learning techniques to data integration tasks (e.g., linking, fusion and enrichment) specially for geospatial data. Mohamed developed number of algorithms for link specification learning, data repair, load balancing and spatial/temporal relation discovery. Currently, Mohamed is leading the integration tasks of the many research projects.

Dr. Diego Moussallem (m), *Homepage:* https://dice-research.org/DiegoMoussallem. Diego Moussallem is a postdoctoral researcher at the Data Science research group (DICE), at University of Paderborn. He worked on different NLP tasks ranging from basic research in computational linguistics to Entity Linking, Machine Translation, Natural Language Generation, and Question Answering. His work resulted in the first neural machine translation model augmented with knowledge

graphs and one state-of-the-art framework in respect of multilingualism and knowledge-graph-based algorithms. Additionally, He contributes actively to the scientific community, being responsible for the Portuguese chapter of DBpedia as well as mentoring students at the Google Summer of Code.

Prof. Dr. Axel-Cyrille Ngonga Ngomo (m), *Homepage:* https://dice-research.org/AxelCyrilleNgongaNgomo. Axel Ngonga is the Data Science chair at the Computer Science department at University of Paderborn. He also leads the Data science research group (DICE). His research interests revolve around Semantic Web technologies, especially link discovery, federated queries, machine learning and natural-language processing. Axel (co-)authored more than 200 reviewed publications and has developed several widely used frameworks such as LIMES, FOX and GERBIL.