RAKI

Collaborative Project

Rapid Explainable AI for Industrial Plants

Project Number: 32100687 **Start Date of Project:** 2019/09/01 **Duration:** 36 months

Deliverable 4.1
Conception of the Verbalization

Dissemination Level	Public
Due Date of Deliverable	M12
Actual Submission Date	M12
Work Package	A4 – Verbalization
Task	WS4.1
Type	document
Approval Status	final
Version	1.0
Number of Pages	11

Abstract: This deliverable presents the preliminary design, a conception, of the verbalization component for the RAKI project.

Gefördert durch:

Bundesministerium für Wirtschaft und Energie

aufgrund eines Beschlusses des Deutschen Bundestages

This project has received funding from the Federal Minister for Economic Affairs and Energy (BMWi) under grant agreement No 32100687.

History

Version	Date	Reason	Revised by
1.0	14/08/2020	Revision of first draft	René Speck
0.1	01/12/2019	Initial requirements	René Speck

Authors

In alphabetical order.

Organization	Name	Contact Information
UPB	Diego Moussallem	diego.moussallem@uni-paderborn.de
ULEI	René Speck	rene.speck@uni-leipzig.de

Contents

1 Introduction

This deliverable presents conceptions pertaining to the design of the verbalization components to be developed within the Rapid Explainable AI for Industrial Plants (RAKI) project. The verbalization is

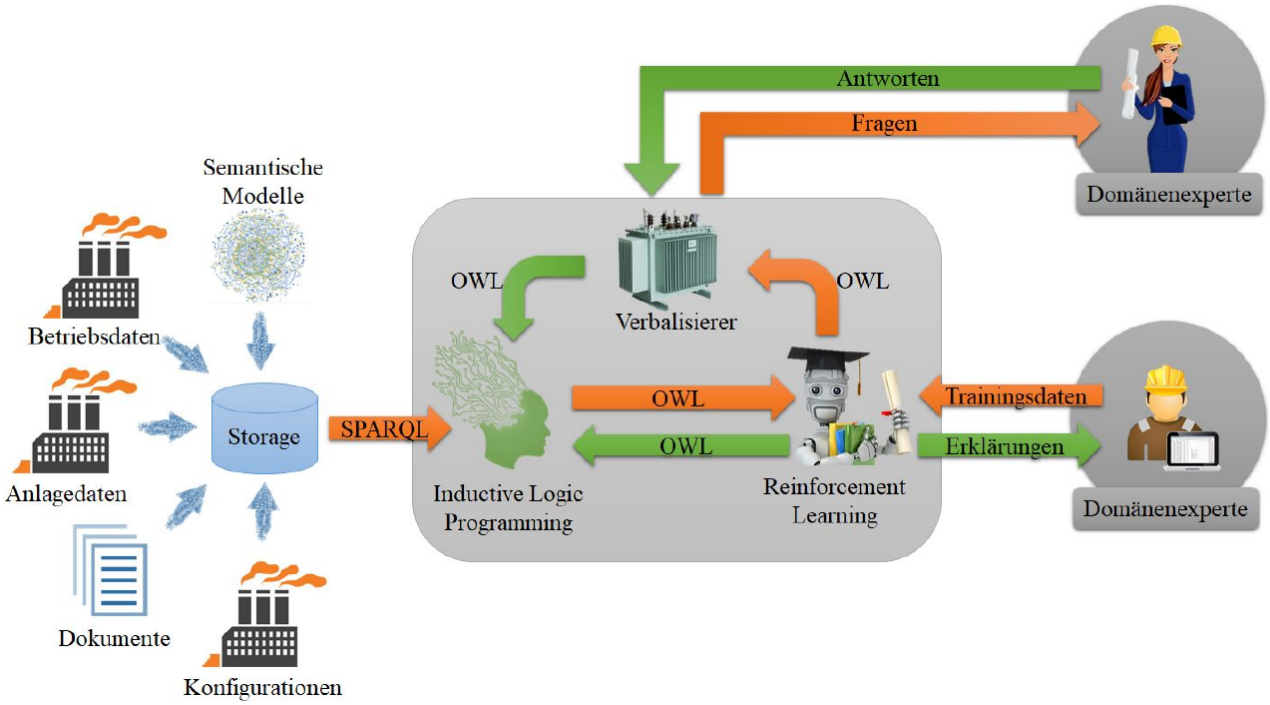


Figure 1: Overview of RAKI

one of the three main modules besides the Inductive Logic Programming (ILP) and the Reinforcement Learning (RL) module in RAKI (compare Figure 1).

Critical decisions detected by the Machine Learning (ML) components will be verbalized and handed out to domain experts. Those experts support the Machine Learning component by providing supplementary data, answering questions pertaining to learned models and hence supporting the automatic decision making in critical states of the machine learning process.

The verbalization components are designed to be generic, hence, to accommodate verbalization algorithms that are loosely coupled and reuse standards. The design of the verbalization components is based on the results of the RAKI requirements specification[1] and the requirements elicitation of the RAKI architecture[2].

Moreover, the components are designed to be compatible with extensions and other verbalization algorithms. Hence, it reuses standards (e.g., Resource Description Framework (RDF), Web Ontology Language (OWL)) as much as possible. The components are also designed to scale. Hence, a packaging concept and a distribution concept are part of the specification, even if the project does not plan for them to be be part of the final version of the prototype.

[1] https://raki-projekt.de/deliverables/D1.1.pdf
[2] https://raki-projekt.de/deliverables/D1.2.pdf

2 Sate of the Art

2.1 Natural Language Generation (NLG)

NLG is the process of automatically generating coherent Natural Language (NL) text from non-linguistic data [14]. Recently, the field has seen an increased interest in the development of NLG systems focusing on verbalizing resources from Semantic Web (SW) data [5]. The SW aims to make information available on the Web easier to process for machines and easier to understand for humans. However, the languages underlying this vision, i.e., RDF, SPARQL Protocol and RDF Query Language (SPARQL) and 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.

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, [1] generates sentences in English and Greek from OWL ontologies. Also, SPARQL2NL [9] uses rules to verbalize atomic constructs and combine their verbalization into sentences.

With this aim, we developed 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. LD2NL generates texts which can be easily understood by humans and is now considered the state-of-the-art approach to verbalize SW languages.[3]

An evaluation of a given Natural Language Generation (NLG) system may be carried out either automatically or manually. Generally, the NLG community has opted to use automatic metrics to decrease human efforts and time. A common process of automatic evaluation is composed of (1) the source data (input), in our case Class Expressions, (2) generated text (output produced by an NLG system, which is also called a hypothesis), and (3) the reference text of the source data, commonly named as human reference. The human reference is compared automatically against the generated text using a given evaluation metric.

There are plenty of automatic evaluation metrics that are used in the Machine Translation and Natural Language Generation tasks. However, we plan to use in RAKI the most effective metrics according to previous NLG work, which enables a fair scientific comparison between the quality of different NLG systems.

Automatic metrics:

BLEU was created by Papineni [11] widely chosen for evaluating automatically generated text outputs due to its low costs. BLEU uses a modified precision metric for comparing the output with the human reference. The precision is calculated by measuring the n-gram similarity (size 1-4) at word levels. BLEU also applies a brevity penalty by comparing the length of the output with the human reference. Additionally, some BLEU variations have been proposed to improve its evaluation quality. The most common variation deals with the number variability (frequency) of useless words commonly generated by NLG systems. However, the main weakness of BLEU is its difficulty handling semantic variations (i.e., synonyms) while performing the n-gram similarity.

[3] https://github.com/dice-group/LD2NL

METEOR was introduced by Banerjee and Lavie [3] to overcome some weaknesses of BLEU, for example, the lack of explicit word-matching between translation and reference, the lack of recall and the use of geometric averaging of n-grams. The goal of METEOR is to use semantic features to improve correlation with human judgments of translation quality. To this end, METEOR considers the synonymy overlap through a shared WordNet synset of the words.

chrF proposed by Popovic [12, 13] was initially for the use of character n-gram precision and recall (F-score). In addition, chrF also works with word n-grams. Although n-gram is already used in well-known and complex automatic metrics, the investigation of n-grams as an individual metric has not been exploited before. chrF has shown a good correlation with human rankings of different automatic outputs. chrF is simple and does not require any additional information. Additionally, chrF is language and tokenization independent.

TER is different from the metrics mentioned above. TER measures the number of necessary edits in an output (generated text) to match the human reference exactly. The goal of TER is to measure how much effort is needed to fix an automated translation to make it fluent and correct [15]. The edits consist of insertions, deletions, substitutions, and shifts of words, as well as capitalization and punctuation. The TER score is calculated by computing the number of edits divided by the average referenced words.

ROUGE is essentially a set of metrics for evaluating automatic summarization of texts as well as MT and NLG[7]. It works by comparing an automatically produced summary or translation against a set of human reference summaries. ROUGE has some variations such as ROUGE-N, ROUGE-S, and ROUGE-L. ROUGE-N measures unigram, bigram, trigram, and higher-order n-gram overlap while ROUGE-L takes into account the longest matching sequence of words relying on Longest Common Subsequence (LCS). An advantage of using LCS is that it does not require consecutive matches but in-sequence matches that reflect sentence-level word order. ROUGE-S refers to any pair of words in a sentence considering a given order, usually called as skip-gram cooccurrence.

2.2 Neural Machine Translation

Transformer-based models consist of an encoder and a decoder, i.e., a two-tier architecture where the encoder reads an input sequence $x = (x_1, ..., x_n)$ and the decoder predicts a target sequence $y = (y_1, ..., y_n)$. The encoder and decoder interact via a soft-attention mechanism [2, 8], which comprises one or multiple attention layers. We follow the notations from Tang et al. [16] in the subsequent sections: Let m stand for the word embedding size and n for the number of hidden units. Further, let K be the vocabulary size of the source language. Then, h_i^l corresponds to the hidden state at step i of layer l. h_{i-1}^l represents the hidden state at the previous step of layer l while h_i^{l-1} means the hidden state at i of layer $l-1$. $E \in \mathbb{R}^{m \times K}$ is a word embedding matrix, $W \in \mathbb{R}^{n \times m}$, $U \in \mathbb{R}^{n \times n}$ are weight matrices, E_{x_i} refers to the embedding of x_i, and $e_{pos,i}$ indicates a positional embedding at position i.

Transformer models rely deeply on self-attention networks. Each token is connected to every other token in the same sentence directly via self-attention. Thus, the path length between any two tokens is 1. Due to lack of recurrence found in Recurrent Neural Network (RNN), Transformers implement *positional encoding* to input and output. Additionally, these models rely on multi-head attention to feature attention networks, which are more complex in comparison to the 1-head attention mechanism used in RNNs. In contrast to RNN, the positional information is also preserved in positional embeddings. Equation 1 describes the hidden state h_i^l, which is calculated from all hidden states of the previous layer. f represents a feed-forward network with the rectified linear unit (ReLU) as the activation function and layer normalization. The first layer is implemented as $h_i^0 = WE_{x_i} + e_{pos,i}$.

Moreover, the decoder has a multi-head attention over the encoder's hidden states:

$$h_i^l = h_i^{l-1} + f(\text{self-attention}(h_i^{l-1})). \tag{1}$$

2.3 OWL

OWL[4] [10] 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 [4]. 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 |` $\{\mathtt{a}_1,\ldots,\mathtt{a}_m\}$

---

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) [6].

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

3 Conception of the Verbalization

The conception of the Verbalization module with its inner components is briefly visualized in Figure 2. The Verbalization module consists of four inner components, a rule-based verbalizer, a post-editing

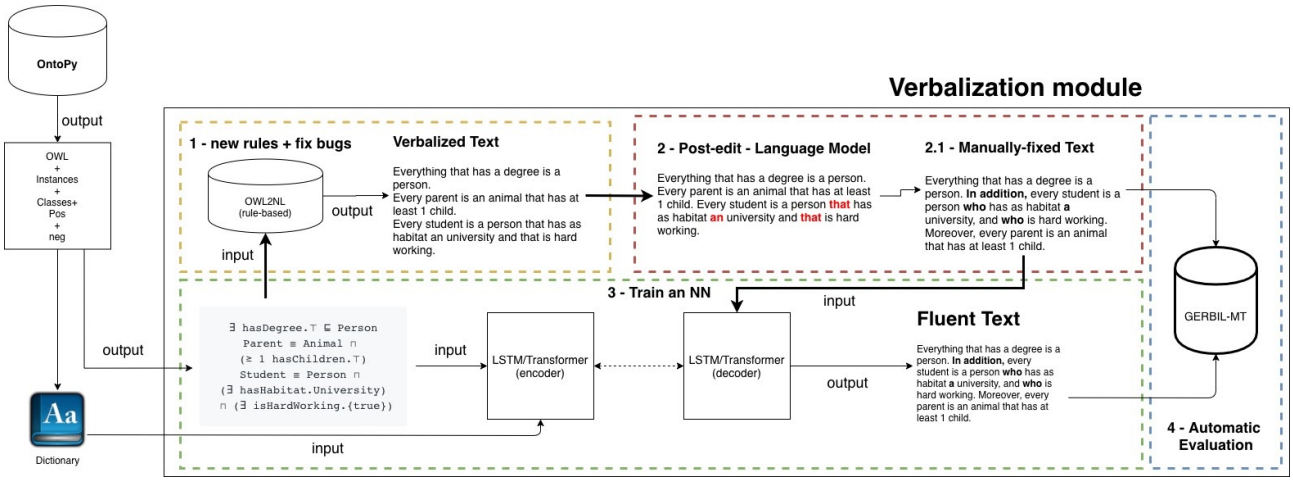


Figure 2: Overview of the conception of the verbalization

component, a neural network and an evaluation component.

This module verbalizes the set of concepts, along with their scores, produced by the ILP module in a given language $\mathcal{L}$. Hence, the required input to generate the verbalizations will be

1. A reference to a domain dictionary
2. A reference to a knowledge graph which can be queried for labels using a standard language, e.g., SPARQL
3. A reference to a concept summarization algorithm for the language $\mathcal{L}$, which can take a concept in $\mathcal{L}$ and return a shorter concept which minimize the classification error.

The module returns two types of outputs, (1) an automatic generated explanation pertaining to the decisions and actions taken by each of the previous modules and (2) an automatic generated question along with possible answers in a multiple-choice format to the domain expert for a human-in-the-loop scenario. Additionally, a sanity-check option will be provided, where the domain expert can register a non-sensical question-answer construct.

For the deployment of the platform, this module should be built into packaged images, e.g., Docker and deployable locally, e.g., with docker-compose. This module should notify users when it has an irrecoverable failure. In addition, it should give access to its status to support to display the status in the UI.

3.1 Sentence Planner

The task of the sentence planner within the RAKI project is to generated English sentences $\mathcal{S}$, i.e., natural language expressions, based on given expressions $\mathcal{E}$ in a formal language $\mathcal{L}$. This task is basically a bilingual translation by translating $\mathcal{E}$ to $\mathcal{S}$. For instance, consider the following expression:

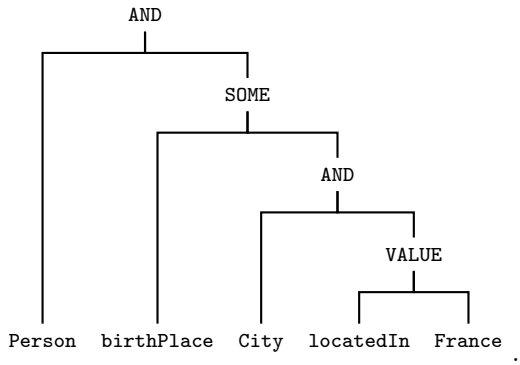
$Person \sqcap \exists birthPlace.(City \sqcap locatedIn.\{France\})$. This expression might be expressed in natural language by: "Persons whose birth place is a city located in France". The sentence planner applies the rule-based verbalization (Section 3.1.1) and the post-editing component (Section 3.1.2).

3.1.1 Rule-based Verbalization

This component extends the LD2NL framework to verbalize concepts provided by the ILP module. LD2NL generates texts which can be understood by humans (see Section 2). 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. The advantages of a rule-based approach are high adequacy and fluency but the disadvantages are no treatment of long expressions and no storytelling.

The verbalized concepts by the rule-based approach then feet to the post-editing component to prepare the training datasets for the neural network in the document planner (Section 3.2).

3.1.2 Post-Editing

In the post-editing component the verbalizations produced by the rule-based verbalization are manually fixed to improve the quality if needed. Those improved verbalizations are part of the training data for the neural network (see Section 3.2.1).

3.2 Document Planner

The task of the document planner within the RAKI project is to decide which information should be used to verbalize and to increase fluency of the generated text. The document planner implements methods to decide the order in which the verbalized expressions should be processed and how to present the verbalizations, i.e., in which format. The document planner relies on the idea and encoder-decoder architecture of NABU to increase the fluency of the verbalizations.

3.2.1 Neural Network

This component is based on the idea and results of NABU, a multilingual graph-based neural RDF verbalizer[5]. The architecture of NABU is based on a encoder and decoder (see Figure 3).

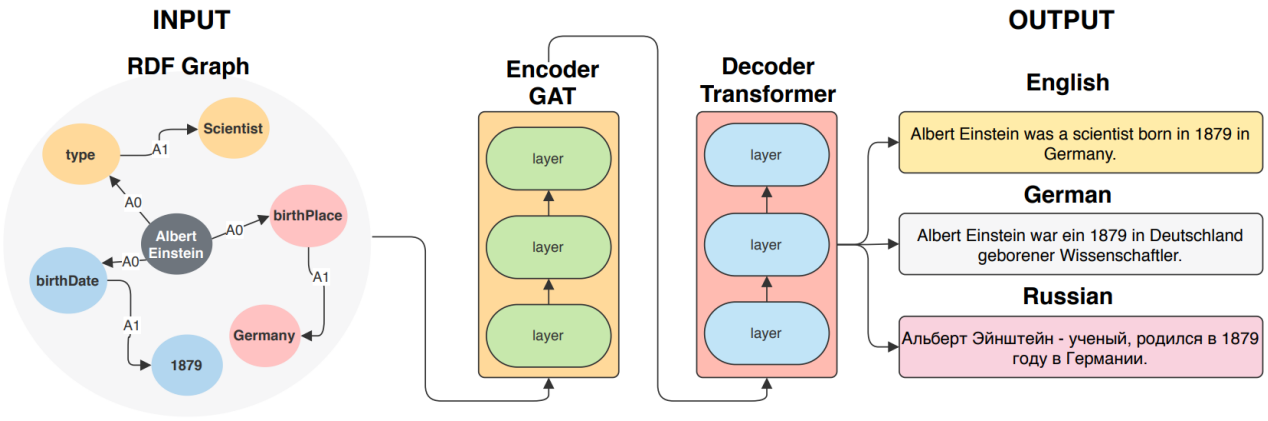


Figure 3: NABU architecture

The input of our neural network is a set of provided concepts and the output are natural language expression. The target training data for learning how the network translates is provided by the post-edited verbalizations (see Section 3.1.2) while the source training data are the concepts.

3.3 Evaluation

For our experiments we will extend the benchmarking framework Hobbit/Gerbil for benchmarking NLG systems.

3.3.1 Framework Key Performance Indicators

The following KPIs will be used for our experiments:

Precision, Recall, Accuracy and F1-Score We apply Precision $P = \frac{TP}{(TP+FP)}$, Recall $R = \frac{TP}{(TP+FN)}$, Accuracy $A = \frac{TP+TN}{(TP+FP+FN+TN)}$ and F1-Score $F1 = \frac{2P}{PR}$. TP, FP, FN, and TN are determined as defined in the DL-learner framework [4].

Runtime The runtime of algorithms or queries is a central metric for evaluating the performance of a system. It measures the time that passes between starting and finishing a task.

We are aiming to consider the following additional KPIs in our experiments:

- BLEU [11]
- METEOR [3]
- chrF++[12, 13]

[5] https://raki-projekt.de/publications/

- TER [15]
- Error Count

References

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

[2] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[3] Satanjeev Banerjee and Alon Lavie. Meteor: An automatic metric for mt evaluation with improved correlation with human judgments. In *Proceedings of the acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*, pages 65–72, 2005.

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

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

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

[7] Chin-Yew Lin. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain, July 2004. Association for Computational Linguistics.

[8] Thang Luong, Hieu Pham, and Christopher D. Manning. Effective approaches to attention-based neural machine translation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421. ACL, 2015.

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

[10] W3C OWL Working Group. *OWL 2 Web Ontology Language: Document Overview*. W3C Recommendation, 2009.

[11] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA, July 2002. Association for Computational Linguistics.

[12] Maja Popović. chrf: character n-gram f-score for automatic mt evaluation. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 392–395, 2015.

[13] Maja Popović. chrf deconstructed: beta parameters and n-gram weights. In *Proceedings of the First Conference on Machine Translation: Volume 2, Shared Task Papers*, pages 499–504, 2016.

[14] Ehud Reiter and Robert Dale. *Building natural language generation systems*. Cambridge university press, 2000.

[15] Matthew Snover, Bonnie Dorr, Richard Schwartz, Linnea Micciulla, and John Makhoul. A study of translation edit rate with targeted human annotation. In *Proceedings of association for machine translation in the Americas*, volume 200, 2006.

[16] Gongbo Tang, Mathias Müller, Annette Rios, and Rico Sennrich. Why self-attention? a targeted evaluation of neural machine translation architectures. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4263–4272, 2018.