Explainable Integration of Knowledge Graphs using Large Language Models

Abdullah Fathi Ahmed[1], Asep Fajar Firmansyah[1,2](✉), Mohamed Ahmed Sherif[1], Diego Moussallem[1,3], and Axel-Cyrille Ngonga Ngomo[1]

[1] Paderborn University, Warburger Str. 100, 33098 Paderborn, Germany
afaahmed, mohamed.sherif, diego.moussallem, axel.ngonga@upb.de
[2] The State Islamic University Syarif Hidayatullah Jakarta, Jakarta, Indonesia
asep.fajar.firmansyah@upb.de, asep.airlangga@uinjkt.ac.id
[3] Jusbrasil, Brazil

Abstract. Linked knowledge graphs build the backbone of many data-driven applications such as search engines, conversational agents and e-commerce solutions. Declarative link discovery frameworks use complex link specifications to express the conditions under which a link between two resources can be deemed to exist. However, understanding such complex link specifications is a challenging task for non-expert users of link discovery frameworks. In this paper, we address this drawback by devising NMV-LS, a language model-based verbalization approach for translating complex link specifications into natural language. NMV-LS relies on the results of rule-based link specification verbalization to apply continuous training on T5, a large language model based on the Transformer architecture. We evaluated NMV-LS on English and German datasets using well-known machine translation metrics such as BLUE, METEOR, ChrF++ and TER. Our results suggest that our approach achieves a verbalization performance close to that of humans and outperforms state of the art approaches. Our source code and datasets are publicly available at `https://github.com/dice-group/NMV-LS`.

Keywords: KG Integration · Neural Machine Verbalization · Explainable AI · Semantic Web · Machine Learning Applications · Large Language Models

1 Introduction

Heterogeneous knowledge graphs that obey the principles of linked data are increasing in number. However, relatively few heterogeneous knowledge graphs are actually linked. The current Linked Open Data (LOD) statistic[1] shows that

Abdullah Fathi Ahmed and Asep Fajar Firmansyah contributed equally to this research.

[1] Release: 05.05.2021, accessed 24.11.2021 `https://lod-cloud.net/#about`, retrieved using `https://github.com/lod-cloud/lod-cloud-draw/blob/master/scripts/count-data.py`

there are 1301 knowledge graphs having 395.12 billion triples and only 2.72 billion links. Therefore, discovering links among these knowledge graphs is a major challenge to achieving the LOD vision[2]. Moreover, the linked knowledge graphs build the backbone of various data-driven applications, including information retrieval, recommender systems, search engines, question answering systems and digital assistants.

Declarative link discovery (LD) frameworks are used to link entities among knowledge graphs. These frameworks use complex link specifications (LSs) to express the conditions required to declare a link between two resources. For instance, state-of-the-art LD frameworks such as LIMES [15] and SILK [10] adopt a property-based computation of links between entities. For configuring link discovery frameworks, the user can either (1) manually enter a LS or (2) use machine learning for automatic generation of LSs. In both cases, a domain expert must manually write LS or set the configuration of machine learning algorithms that are used to generate LS. Furthermore, LD experts can easily understand the LS produced by such algorithms and modify it if needed. However, most lay users lack the expertise to proficiently interpret those LSs. Due to this lack of expertise, these users have difficulty when they i) check the correctness of the generated LS, ii) customize their LS, or iii) decide between possible interpretations of their input in an informed manner.

The aforementioned challenges can be seen as a bottleneck problem which degrade the effort and potential for ML algorithms to create such LSs automatically. Thus, addressing the explainability of link discovery-based artificial intelligence has become increasingly popular. For instance, the authors from [2] introduced a bilingual rule-based approach to verbalize the LS thus addressing the explainability of LD. In addition, Ahmed et al [1] extended the previous approach and devised a multilingual rule-based approach including English, German, and Spanish. They also presented a first attempt for creating neural architecture, which is a bidirectional RNN-LSTM 2 layers encoder-decoder model with an attention mechanism [14]. However, their neural model failed to generalize as the vocabulary was very small and not diverse.

In this work, we alleviate the vocabulary problem found in Ahmed et al [1] by proposing a language-based LS approach, named NMV-LS. To this end, we propose a pipeline architecture consisting of two stages. The first stage is a rule-based verbalizer to generate the necessary data to feed the second stage. The second stage relies on a few-shot learning approach by fine-tuning a large language model (LLM), in our case, T5. The underlying idea of using a language model is to verbalize LS from different types of systems only by using few examples. For example, LSs from LIMES [15] differ from the ones used in SILK [10]. In addition, the second stage contains a standard seq2seq 2 layers encoder-decoder architecture using different RNN cells such as GRU, LSTM, BiLSTM, and transformer trained with more diverse data. Figure 2 depicts the proposed architecture.

[2] https://www.w3.org/DesignIssues/LinkedData.html

To evaluate NMV-LS, we designed two settings. In the first setting, we used two datasets for assessing the first part of our approach (i.e., standard encoder-decoder architectures). The first dataset contains 107 thousand English pairs and the second dataset contains 73 thousand German pairs. It should be noted that each pair is nothing but an LS and its verbalization. In the second setting, we used human annotated data for evaluating our second part of our approach (i.e., few-shot learning using T5 model). We created a human annotated data from LIMES with only 100 pairs, human annotated manipulated data from LIMES with only 8 pairs, and human annotated data from SILK with only 8 pairs. It is important to note that we evaluated our second part only on English.

Our main contributions are as follows:

- We present NMV-LS, a language model-based LS verbalization approach which relies on a few-shot learning strategy.
- We propose an approach which is capable of verbalizing different types of LS thus mitigating the high efforts for creating linguistic rules to each system.
- We propose an approach which is easily extensible to other languages.

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 approach underlying neural machine verbalization LS in Section 3, followed by the evaluation of our approach with respect to the automatic evaluation standard metrics BLEU, METEOR, ChrF++, and TER. We used BENG [12] to automatically measure the performance of our approach in Section 4. After a brief review of related work in Section 5, we conclude with some final remarks in Section 6.

2 Preliminaries

2.1 Link Specification LS

LS consists of two types of atomic components: *similarity measures* m, which are used to compare the property values of input resources, and *operators* ω that are used to combine these similarities into more complex specifications. We define a similarity measure m as a function $m : S \times T \rightarrow [0, 1]$, where S and T are the sets of source and target resources, respectively. We use *mappings* $M \subseteq S \times T$ to store the results of the application of a similarity function m to $S \times T$.

We also define a *filter* as a function $f(m, \theta)$. A specification is named *atomic LS* when it consists of exactly one filtering function. Although a complex specification (*complex LS*) can be obtained by merging two specifications L_1 and L_2 through an *operator* ω that combines the results of L_1 and L_2, here we use the operators $\sqcap$, $\sqcup$ and $\backslash$ as they are complete and frequently used to define LS [20]. A graphical representation of a complex LS is given in Figure 1. We define the semantics $[[L]]_M$ of an LS L w.r.t. a mapping M as given in Table 1. The mapping $[[L]]$ of an LS L with respect to $S \times T$ contains the links that will be generated by L.

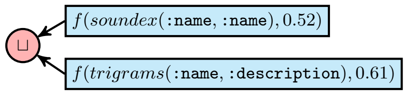
$\sqcup$
$f(soundex(\texttt{:name}, \texttt{:name}), 0.52)$
$f(trigrams(\texttt{:name}, \texttt{:description}), 0.61)$

Fig. 1: A complex LS. The filter nodes are rectangles while the operator node is a circle.

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 1: Link Specification Syntax and Semantics.

2.2 Neural Machine Verbalization

Given a source sentence $\mathbf{x}$ and a target sentence $\mathbf{y}$, verbalization is tasked with finding $\mathbf{y}$ that maximizes the conditional probability of $\mathbf{y}$ (i.e., $\arg\max_{\mathbf{y}} p(\mathbf{y} \mid \mathbf{x})$). In neural machine verbalization, an encoder-decoder model with a set of parameters is trained to maximize the conditional probability of sentence pairs using a parallel training dataset. Accordingly, a verbalization model that learned the conditional distribution can generate a corresponding verbalization of a given sentence by searching for the sentence that maximizes the conditional probability.

3 Approach

NMV-LS consists of two stages. The first stage is rule-based verbalizer introduced in [1] to generate silver data for the second stage, blue colored background in Figure 2 . The second stage is with green colored background in Figure 2. The second stage contains two independent parts. The first part of stage 2 is based on standard encoder-decoder architectures such as two layers seq2seq with GRU, LSTM and BiLSTM, and transformer. The second part of stage 2 applies the concept of few-shot learning and is based on T5 model. In Figure 2, ① means that the data is from Limes silver data, ② means that the training data is a combination of Limes silver data and human annotated Limes silver data, ③ is a combination of ② and humane annotated manipulated Limes LS, and ④ is a combination of ③ and humane annotated Silk LS. In ②, the human annotation is applied only on the verbalization of LS without changing LS. Manipulated Limes LS means that we altered the structure of Limes LS. Listing 1.1 shows an example of Limes silver data, Listing 1.2 is an example of Limes human annotated data, Listing 1.3 is an example of Limes human annotated manipulated data, and Listing 1.4 is an example of Silk human annotated data.

3.1 Rule-based verbalizer

The rule-based verbalizer in [1] is based on Reiter & Dale NLG architecture [19]. In [1] , real datasets (knowledge graphs) are used to generate LSs using Wombat [20]. Since the number of properties used in [1] is limited, it results in less diverse LSs. Our goal is to add more proprieties into each generated LSs.

```
Source =
OR(mongeElkan(x.title,y.title)|0.45,
   cosine(x.title,y.streetName)|0.37)

Target =
A link will be generated if
- the titles of the source and the target have a Mongeelkan similarity of 45%
      or
- the title of the source and the streetName of the target have a Cosine
     similarity of 37%.
```

Listing 1.1: Limes silver data: A pair that contains an LS and its verbalization in English.

```
Source=
AND(AND(ratcliff(x.givenName,y.givenName)|0.0,AND(OR(jaroWinkler(x.givenName,
     y.authors)|0.37,cosine(x.givenName,y.givenName)|0.0)|0.0,ratcliff(x.
     givenName,y.givenName)|0.37)|0.37)|0.0,jaroWinkler(x.givenName,y.
     givenName)|0.37)

Target= a link will be produced supposing that the givenNames of the source
     and the target have a Ratcliff similarity of 0% or the givenName of the
     source and the author of the target have a Jarowinkler similarity of 37%
     or the givenNames of the source and the target have a Cosine similarity
     of 0% and a Ratcliff similarity and a Jarowinkler similarity of a 37%
```

Listing 1.2: Limes human annotated data: A pair that contains an LS and its verbalization in English.

Therefore, in this work we create 10 templates to generate LSs relying on the rules defined in Wombat. The complexity of an LS is formally defined as the number of the combined atomic LS so that an LS is more complex when it contains a higher number of the combined atomic LS. For example, the template $(A_1 \sqcup A_2) \sqcap (A_3 \sqcup A_4)$ is less complex than $(A_1 \sqcup A_2) \sqcap (A_3 \sqcup A_4) \sqcap (A_5 \sqcup A_6)$, where A_i is atomic LS.

3.2 Standard Encoder-Decoder Architectures

As we can see in Figure 2, the first part of the second stage in our approach deploys a set of standard encoder-decoder architectures. Our first part of the second stage is motivated by the advance in sequence-to-sequence neural translation, which belongs to a family of encoder-decoder architecture [22]. The encoder neural network reads and encodes a source sentence into a vector. The decoder translates from the encoded vectors to a sequence of symbols (i.e., words). The goal here is to maximize the probability of a correct translation by jointly training the whole encoder-decoder system using source sentences. We rely on a Recurrent Neural Network (RNN) for both encoding and decoding [7], with the attention mechanism introduced in [3]. We deploy RNN-GRU-2 layer and RNN-Bi/LSTM-2 layer architectures to perform the verbalization. The first architecture is based on Long Short-Term Memory (LSTM) [9], while the second architecture is based on Gated Recurrent Unit (GRU) [7]. Given a sequence of tokens (i.e., words) $\mathbf{x} = (x_1, \cdots, x_T)$ as input at time step t and a sequence of

```
Source=
trigrams(x.givenName, y.name)|0.8 AND cosine(x.title, y.label)|0.7 Or
     levenshtein(x.streetAdress, y.locationAdress)|0.9

Target=
The link will be generated when the givenname of the source and the name of
     the target have a trigrams similarity of 80% and the title of the source
     and the label of the target have a cosine similarity of 70% or the
     streetAdressenname of the source and the locationAdress of the target
     have a levenshtein similarity of 90%
```

Listing 1.3: LIMES human annotated manipulated data: A pair that contains an LS and its verbalization in English.

```
Source=
 min( mongeelkanSimilarity(?x/p:producer, ?y/p:producer), ratclifDisitance(x/
      p:city,y/p:city))

Target=
 The link will be generated if the labels of the source and the target have
      minimum mongeelkan similarity or the cities of the source and the target
       have minimum ratclif distance
```

Listing 1.4: SILK human annotated data: A pair that contains an LS and its verbalization in English.

tokens $\mathbf{y} = (y_1, \ldots, y_T)$ as output, our encoder-decoder is jointly trained to maximize the probability of a correct verbalization.Where $\mathbf{x}$ is the representation of LS and $\mathbf{y}$ is the representation of natural text verbalized by a trained decoder. The length of $\mathbf{x}$ may differ from the length of $\mathbf{y}$. For our proposed NMV-LS (i.e., part one of stage two), we use additive attention (as in [3]) with the conditional probability $p(y_i|y_1, \ldots, y_{i-1}, \mathbf{x}) = g(y_{i-1}, s_i, \mathbf{c}_i)$, where s_i is an RNN decoder's hidden state at time i. Formally, $s_i = f(s_{i-1}, y_{i-1}, \mathbf{c}_i)$ (see [7] for more details). In this part of our approach, we also deploy transformer. Transformer is a sequence-to-sequence architecture that relies entirely on the attention mechanism to transform one sequence into another with the help of two parts encoder and decoder without implying any recurrent networks (GRU, LSTM, etc). The architecture consists of multiple identical encoders and decoders stacked on top of each other (more details in [25]). We use our rule-based verbalizer to generate silver data to train our models. However, before feeding these data, we need to apply some preprocessing techniques.

3.3 Few-shot Learning using T5 model

As depicted in Figure 2, the second part of the second stage in our approach is based on a few-shot learning strategy that involves fine-tuning a large language model (LLM).

To address the vocabulary issue in Ahmed et al [1], we base our approach on a few-shot learning approach by fine-tuning a large language model (LLM) such T5 model [18]. T5 is a pre-trained model for text-to-text generative multitasking based on transformer encoder-decoder and it is pre-treained on a large pre-

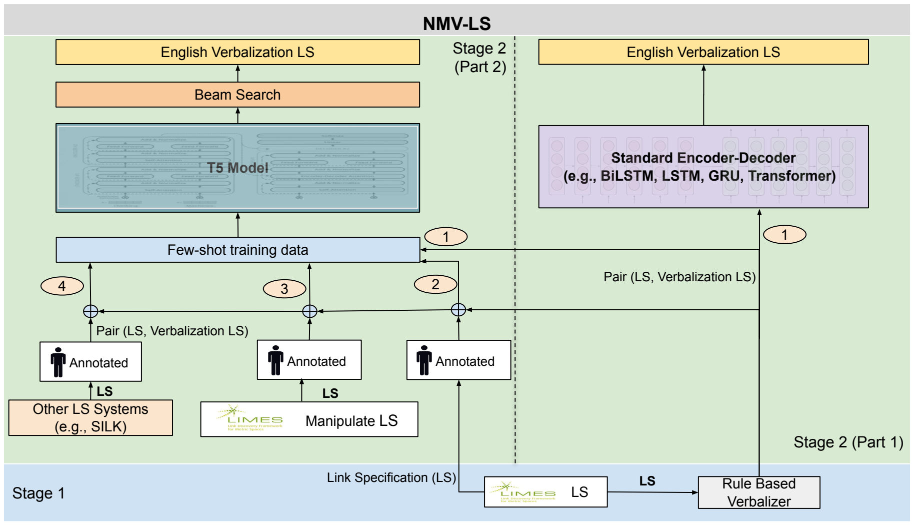

Fig. 2: LS Neural Machine Verbalization System

training dataset (C4) [18]. Using T5 pre-trained model allows the model to learn the structure and pattern of natural language from a vast quantity of diverse, real-world text data. This can assist the model in learning to comprehend and generate high-quality, human-like text, which is useful for a variety of natural language processing tasks. In addition, T5 pre-trained model can frequently be fine-tuned for specific tasks, particularly in our case to learn the complexity of LS and generate verbalizations of LS with additional data and training time.

To use the T5 pre-trained model for few-shot learning in our model, as shown in figure 2, we fine-tune it on four different small training datasets, as detailed in section 4.3, where those datasets were designed based on the LSs of Limes. The model's goal is to generalize the verbalization of a wide range of LSs. Given a sequence LS tokens as input represented by $ls = \{w_1, w_2, ..., w_N\}$ and mapped into sequence embeddings before being fed into the encoder of T5, which outputs a sequence of embedding vectors. Furthermore, the decoder of T5 accepts as inputs both encoder outputs and previously generated tokens from the decoder during the auto-regressive decoding. Moreover, linear transformation and softmax functions are applied to the decoder outputs. In addition, beam search decoding [23] is utilized to generate the verbalization LS from the model outputs.

4 Evaluation

4.1 Data

Since there are no gold standard datasets for an NM verbalization task to translate link specification to natural languages, we generated silver standard datasets

Table 2: Statistics about our datasets used in the experiments, where V_{max} is the maximum verbalization length (in words) and V is the verbalization length.

Data	# Records	$V < 50$	$51 < V < 100$	$V > 100$
EN	107424	3744 (3.49%)	88320 (82.22%)	15360 (14.30%)
DE	73008	3888 (5.33%)	48384 (66.27%)	20736 (28.40%)

using the rule-based approach introduced in [2] and [1]. For evaluating the standard encoder-decoder architecture, we generated three datasets with the following sizes: 107k pairs (English) and 73k pairs (German). Table 2 shows statistical information about the data. For evaluating the fine-tuning of T5, we combined 10k pairs (English) from 107k pairs (English) with 100 pairs human annotated data from LIMES, 8 pairs human annotated manipulated data from LIMES, and only 8 pairs human annotated data from SILK.

4.2 Evaluation Metrics.

To ensure consistent and clear evaluation, we evaluate our approach with respect to the automatic evaluation standard metrics BLEU [16], METEOR [4], ChrF++ [17] and TER [21].

We used BENG [12] to evaluate our approach automatically. BENG is an evaluation tool for text generation that abides by the FAIR principles and is built upon the successful benchmarking platform GERBIL [24].

4.3 Experimental Setup

As we can see in Figure 2, our approach consists of two stages. The first stage is the rule-based verbalizer and the second stage contains two parts. The first part is based on standard encoder-decoder architectures and the second part is based on few-shot learning method by fine-tuning a large language model (LLM) such T5. However, the first stage feeds the two parts of the second stage. For instance, ① means that the data is from LIMES silver data generated by the first stage of NMV-LS which is the rule-based verblizer. ① feeds the both two parts of the second stage in our pipeline architecture. For evaluating our first part of of the second stage in our approach(i.e., standard encoder-decoder architectures), we conducted three sets of experiments for both English and German to answer the following research question:

Q_1. Does the complexity of LS impact the performance of our NMV-LS in case of training standard encoder-decoder architectures?

For evaluating our second part of the second stage in our approach(i.e., few-shot learning using T5 model), we conducted one set of experiments for English to answer the following research questions:

Q_2. Does fine-tuning a LLM improve the verbalization of our NMV-LS system ?

Q_3. Does fine-tuning a LLM help to the generalization of different LS for verbalization.

Q_4 How large is the impact of using human annotated data on the quality of verbalization in comparison with using silver data?

Experiment Set 1, English Language (107k dataset). We evaluated a GRU/LSTM/BiLSTM-2 layers encoder-decoder on an English dataset consisting of 107k pairs (each pair contains an LS and its verbalization in English or German), split into 70% for training, 20% for validation, and 10% for testing. For all experiments, we set the parameters as follows: The learning rate is {0.1, 0.01, and 1}, the dropout is 0.1, the embedding dimensionality is 256, the epochs number is {100, 1000, and 10000}, the clipping value is 0.25, SGD optimizer with negative log-likelihood loss function, and the max length of a sentence is {107 and 187 tokens}. The max length of a sentence means that model can filter out all the pairs that have a length greater than the max length. For LSTM/BiLSTM, the batch size is 256. The selection of parameters is manually tuned. We run all GRU on colab and LSTM/BiLSTM on a local server with 1 GPU, 16 CPUs and 32 GB of memory. We use `Pytorch` library to implement our model. The results are listed in Table 3. For these results, we set the learning rate to 0.01 in case for using GRU and to 0.1 in case with using LSTM/BiLSTM. We conducted additional experiments with the learning rate set to 1 to study the impact of learning rate on the results using LSTM/BiLSTM. The results are provided in the Table 5.

Experiment Set 2, German Language. We evaluated the GRU/LSTM/-BiLSTM-2 layers encoder-decoder on the German dataset containing only 73k pairs. LSs are also complex in terms of atomic LSs. For instance, an LS can contain up to 6 atomic LSs A_i combined using operators $\sqcup$ and $\sqcap$. The results of the experiments are shown in the Table 4. The results in Table 4 are obtained with the learning rate set to 0.01 for GRU and to 0.1 for LSTM/BiLSTM with a batch size of 265. We ran more experiments with the learning rate set to 1 to study the impact of learning rate on the results. The results are presented in Table 6.

Table 3: BLEU, METEOR, ChrF++, and TER scores for the English language, evaluated on the 107k dataset; the learning rate is 0.01 for GRU and 0.1 for LSTM/BIlSTM.

Model	Length	Iter.	BLEU	BLEU-NLTK	METEOR	ChrF++	TER
GRU	107	100	35.92	0.36	0.36	0.66	0.69
GRU	187	100	21.73	0.22	0.33	0.58	1.69
GRU	107	1000	41.05	0.41	0.39	0.71	0.63
GRU	187	1000	22.07	0.22	0.22	0.44	0.56
GRU	107	10000	**99.22**	**0.99**	**0.78**	**0.99**	**0.01**
GRU	187	10000	88.81	0.89	0.60	0.93	0.05
LSTM	107	100	82.61	0.83	0.65	0.92	0.27
LSTM	187	100	77.31	0.77	0.58	0.87	0.40
BiLSTM	107	100	85.37	0.85	0.64	0.91	0.26
BiLSTM	187	100	79.23	0.79	0.59	0.89	0.34

Table 4: BLEU, METEOR, ChrF++, and TER scores for the German language evaluated on the 73K dataset. The learning rate is 0.01 for GRU and 0.10 for LSTM/BiLSTM.

Model	Length	Iter.	BLEU	BLEU-NLTK	METEOR	ChrF++	TER
GRU	107	100	41.84	0.42	0.43	0.74	0.66
GRU	187	100	28.67	0.29	0.4	0.67	1.41
GRU	107	1000	49.75	0.5o	0.47	0.79	0.59
GRU	187	1000	54.01	0.54	0.40	0.71	0.38
GRU	107	10000	**99.98**	**1.00**	**0.90**	**1.00**	**0.00**
GRU	187	10000	79.52	0.80	0.54	0.84	0.32
LSTM	107	100	60.40	0.60	0.44	0.70	0.45
LSTM	187	100	76.67	0.77	0.63	0.86	0.49
BiLSTM	107	100	81.90	0.82	0.59	0.86	0.21
BiLSTM	187	100	81.30	0.81	0.59	0.85	0.30

Table 5: BLEU, METEOR, ChrF++, and TER scores for English language evaluated on the 107K dataset with learning rate set to 1.00.

Model	length	Iter.	BLEU	BLEU-NLTK	METEOR	ChrF++	TER
LSTM	107	100	83.01	0.83	0.61	0.86	0.22
LSTM	187	100	68.06	0.68	0.67	0.89	0.55
BiLSTM	107	100	**94.45**	**0.94**	**0.70**	**0.96**	**0.08**
BiLSTM	187	100	86.18	0.86	0.66	0.89	0.16

Experiment Set 3, Transformer. We implemented our transformer model using the `Pytorch` framework with the default parameters, i.e., the number of epochs is 30, the batch size is 256, and the max sentence length is {107, 187}. The results are listed in Table 7.

Experiment Set 4, Few-shot learning on T5. To address the issues raised by employing conventional architectures, such as overfitting and limited vocabulary size as we have seen in previous experiments, we implemented few-shot learning of text generation on the T5 model with a small number of training samples. This experiment is designed with four distinct sets of few-shot training data and three distinct sets of testing data, as shown in Table 8. In the first experiment, we fine-tuned the T5 model using a training dataset of 10k pairs, each consisting of a LS and its English verbalization from LIMES silver data ①. In the second experiment, we fine-tuned T5 using the previous training dataset in combination with 70 pairs of LS and their human-annotated verbalizations from the LIMES silver data ②. In the third experiment, the training dataset

Table 6: BLEU and METEOR, ChrF++, and TER scores for the German language evaluated on the 73K pairs dataset with learning rate set to 1.

Model	Length	Iter.	BLEU	BLEU-NLTK	METEOR	ChrF++	TER
LSTM	107	100	87.19	0.87	0.66	0.90	0.13
LSTM	187	100	96.67	0.97	0.82	0.99	0.06
BiLSTM	107	100	91.74	0.92	0.71	0.93	0.07
BiLSTM	187	100	**99.58**	**1.00**	**0.85**	**1.00**	**0**

Table 7: BLEU, METEOR, ChrF++, and TER scores for German and English evaluated on the 73K and 107K pairs datasets using Transformers.

Data	Length	Iter.	BLEU	BLEU-NLTK	METEOR	ChrF++	TER
107K (En)	107	30	90.89	**0.91**	**0.67**	**0.98**	**0.12**
107K (En)	187	30	**90.92**	**0.91**	**0.67**	**0.98**	**0.12**
73K (De)	107	30	89.98	0.90	0.66	0.97	0.15
73K (De)	187	30	79.11	0.79	0.60	0.93	0.29

Table 8: BLEU, METEOR, ChrF++, and TER scores for English language using Fine-tuned T5 model leveraging few-shot learning.

Train set	Test set	BLEU	BLEU-NLTK	METEOR	ChrF++	TER
①	Limes original LS	76.27	0.76	0.54	0.87	0.15
①	Silk LS	34.26	0.35	0.26	0.54	0.71
②	Limes original LS	77.91	0.78	0.54	0.89	0.13
②	Limes Manipulated LS	45.76	0.46	0.37	0.68	0.55
③	Limes Manipulated LS	63.64	0.64	0.43	0.80	0.48
③	Silk LS	34.93	0.35	0.27	0.54	0.67
④	Silk LS	36.58	0.37	0.34	0.59	0.62

from the second experiment is combined with human-annotated manipulated Limes LS ③. By modifying the formula, the manipulated Limes LS is defined differently than the previous LS. In addition, we fine-tuned the T5 model on the training dataset in an effort to determine whether the model can improve the verbalization of manipulated Limes LS. In the last experiment, we fine-tuned the T5 model using the training data from the previous experiment in combination with Silk LS ④. All experiments are built using the Pytorch lightning framework, with following hyper-parameters: We set the number of epochs to five and the learning rate to 3e-5 and use beam search decoding to generate verbalization LS with the parameters *max length*=256, *num beams*=15, *no repeat ngram size*=6. In addition, *t5-base* is utilized as a pre-trained model. All models based on few-shot learning are evaluated using Table 8's test set, which is designed to investigate the effect of each training dataset on the model's ability to improve the generalization quality of LS verbalization.

4.4 Results & Analysis

To answer Q_1, we set the maximum length of a sentence to be either 107 words (tokens) or 187 words (tokens) based on the statistics in Table 2. This means we filtered out all verbalized sentences that have a length greater than 107 words for experiments where the maximum length of a sentence is set to 107 words. We also removed all verbalized sentences that exceed 125 words for experiments where the maximum sentence length is set to 125 words. In Table 3, we can observe that NMV-LS using GRU achieves a better BLEU score, up to 99.22 (see Table 3). In Table 4, we can observe that the BLEU score is up to 99.98 obtained from our model using GRU when the length of a verbalized sentence is also less than or equal to 107. In Table 5, the BLEU score is 94.45 using BiLSTM with a max length of 107. Furthermore, Table 4 and 6 show that the NMV-LS model achieves better scores when the length of a verbalized sentence is 187. For instance, the BLEU score on the $73k$ German dataset is 76.67 using LSTM and 99.58 using BiLSTM (see Table 4 and 6). The reason is that the $73k$ German dataset contains complex LSs and 28.40% of their verbalizations have sentence lengths greater than 100 words, and these sentences are filtered out, which in turn affects the training process. Especially since the size of the dataset is only $73k$ pairs, resulting in a decreased performance. From all these observations, we conclude that the complexity of LS plays a crucial role in the performance of our NMV-LS model. Furthermore, GRU is more sensitive to the complexity of LS than LSTM/BiLSTM. LSTM/BiLSTM can handle very complex LSs and improve performance.

To answer Q_2, we analysed the results in Table 8. First, Limes original LS (i.e, it means that LS generated by Limes) is used to evaluate the second part of NMV-LS. Our findings indicate that our technique is capable of generating verbalization at the human level and outperforms earlier approaches. For instance, BLUE score is 76.27, ChrF++ is 0.87, and METEOR is 0.54. Note that we fine-tuned our model with only ①. In another word, we only used Limes silver data generated by the first stage of NMV-LS (i.e., rule-based verbalizer) to fine-tune our model. This answers Q_2.

To answer Q_3, we deployed the fine-tuned model on ① Limes silver data and evaluated on Silk LS as extremest case because Limes & Silk have different rules and grammars to build their LSs. The goal is to study to which extent our model can be generalized to verbalize LSs in different formats and from different systems (e.g., Silk). The results in Table 8 show that our model achieves BLUE score of 34.26. Another case is that we fine-tuned NMV-LS using ② and tested on Limes manipulated LSs. In this case, NMV-LS scores 45.76 BLUE, 0.68 ChrF++, and 0.37. Another case is fine-tuning our model on training data ③ and evaluating on Silk. From the results, there is no improvement comparing with the result generated by the model fine-tuned on ① and tested on Silk. This can be justified that both ① and ③ do not contain any information about Silk. To investigate this further, we added few samples of Silk LSs to create ④. To this end, we used ④ for fine-tuning NMV-LS and then evaluated on Silk. This improved the performance by 1.65%. We see these results, in all cases, as a big milestone toward generalizing our approach to verbalize LSs produced by other systems.

To answer Q_4, we implemented a couple of cases. In first case, we fine-tuned NMV-LS using ② and evaluated on Limes original LSs. The results in Table 8 indicates that human annotated verbalization of a LS improves the verbalization very slightly. For instance, BLUE score is 77.91 and it is 1.65% higher than BLUE score produced using ①. In the second case, we fine-tuned NMV-LS applying ③ and evaluated on Limes manipulated LSs. This improved the performance by 17.88 in BLUE score. We believe that the improvement is lead by including Limes manipulated LSs in the training data ③. This answer Q_4.

5 Related Work

Explainable artificial intelligence (XAI) is widely acknowledged as a crucial feature for the practical use of AI models. Thus, there is an emerging need for understanding the results achieved by such models. With this comes the need for *the verbalization of semantic data (i.e., translation to natural text)* involved with such approaches (e.g., LD and LS systems which are our focus here). For instance, the authors of [5] have surveyed (XAI) as new field of research and covered many aspects of it. While in the work [8], the authors used a convolutional neural network (CNN) model combined with a BiLSTM model as encoder to extract video features and then feed these features to an LSTM decoder to generate textual descriptions. This work [8] and our work both fall under post-hoc explainability approaches such as text explanations(see [5]). In last years, the neural machine translation has achieved a notable momentum [11] [3][7][22][6]. These papers have proposed the use of neural networks to directly learn this conditional distribution that reads a sentence and outputs a correct translation (from a natural language to another natural language, e.g., English to French or English to German).

Recently,transfer learning, where a model is initially pre-trained on a data-rich task before being fine-tuned on a downstream task, has emerged as a potent method in natural language processing (NLP). Applying few-shot learning by fine-tuning LLM such as T5 on a range of English-based NLP tasks, including as sentiment analysis, question answering, and document summarizing achieves state-of-the-art results [18]. However, few works have addressed the link specification verbalization (i.e., translation to natural languages). Recently, the authors addressed the readability of LS and proposed a generic rule-based approach to produce natural text from LS [2]. Their approach is motivated by the pipeline architecture for natural language generation (NLG) systems performed by systems such as those introduced by Reiter & Dale [19]. While in this work [1], they proposed a multilingual rule-based approach, including English, German, and Spanish, to produce natural text from LS. They also have presented a neural architecture which is a bidirectional RNN-LSTM 2 layers encoder-decoder model with an attention mechanism [13]. We used [2] and [1] to generate our silver dataset.

6 Conclusion & Future Work

In this paper, we present NMV-LS, a language-based LS verbalization system that is able to translate (verbalize) LS into natural language. our approach consists of two independent parts. The first part is based on standard encoder-decoder architectures such as two layers seq2seq with GRU, LSTM and BiLSTM, and transformer. The second part applies the concept of few-shot learning and is based on T5 model. The first part of our approach is multilingual in nature, where we tested it to generate both English and German verbalization. The second part is evaluated on English. In future work, we plan to evaluate the second part of the second stage in NMV-LS on more languages such as German, French, and Spanish. In addition, we will integrate our model into the LS learning algorithms, e.g., WOMBAT and EAGLE for generating on-the-fly multilingual verbalization of the learned LS.

Acknowledgements

We acknowledge the support of the German Federal Ministry for Economic Affairs and Climate Action (BMWK) within project SPEAKER (01MK20011U), the German Federal Ministry of Education and Research (BMBF) within the EuroStars project PORQUE (01QE2056C), the German Research Foundation (DFG) within the project INGRID (NG 105/7-3), the Ministry of Culture and Science of North Rhine-Westphalia (MKW NRW) within the project SAIL (NW21-059D), the European Union's Horizon Europe research and innovation programme within project ENEXA (101070305), and Mora Scholarship from the Ministry of Religious Affairs, Republic of Indonesia.

References

1. Ahmed, A.F., Sherif, M.A., Moussallem, D., Ngonga Ngomo, A.C.: Multilingual verbalization and summarization for explainable link discovery. Data and Knowledge Engineering **133**, 101874 (2021). https://doi.org/10.1016/j.datak.2021.101874
2. Ahmed, A.F., Sherif, M.A., Ngomo, A.C.N.: Lsvs: Link specification verbalization and summarization. In: International Conference on Applications of Natural Language to Information Systems. pp. 66–78. Springer (2019)
3. Bahdanau, D., Cho, K., Bengio, Y.: Neural machine translation by jointly learning to align and translate (2016)
4. Banerjee, S., Lavie, A.: 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 MT and/or Summarization. pp. 65–72. ACL (2005)
5. Barredo Arrieta, A., Díaz-Rodríguez, N., Del Ser, J., Bennetot, A., Tabik, S., Barbado, A., Garcia, S., Gil-Lopez, S., Molina, D., Benjamins, R., Chatila, R., Herrera, F.: Explainable artificial intelligence (xai): Concepts, taxonomies, opportunities and challenges toward responsible ai. Information Fusion **58**, 82–115 (2020). https://doi.org/10.1016/j.inffus.2019.12.012
6. Cho, K., van Merrienboer, B., Bahdanau, D., Bengio, Y.: On the properties of neural machine translation: Encoder-decoder approaches (2014)
7. Cho, K., van Merrienboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., Bengio, Y.: Learning phrase representations using rnn encoder-decoder for statistical machine translation (2014)
8. Dong, Y., Su, H., Zhu, J., Zhang, B.: Improving interpretability of deep neural networks with semantic information (2017)
9. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. **9**(8), 1735–1780 (Nov 1997). https://doi.org/10.1162/neco.1997.9.8.1735

10. Isele, R., Jentzsch, A., Bizer, C.: Efficient Multidimensional Blocking for Link Discovery without losing Recall. In: WebDB (2011)
11. Kalchbrenner, N., Blunsom, P.: Recurrent continuous translation models. In: Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing. pp. 1700–1709. Association for Computational Linguistics, Seattle, Washington, USA (2013), `https://www.aclweb.org/anthology/D13-1176`
12. Moussallem, D., Kaur, P., Ferreira, T., van der Lee, C., Shimorina, A., Conrads, F., Roder, M., Speck, R., Gardent, C., Mille, S., Ilinykh, N., Ngonga Ngomo, A.C.: A general benchmarking framework for text generation. pp. 27–33 (2020), international Workshop on Natural Language Generation from the Semantic Web 2020, WebNLG+ ; Conference date: 18-12-2020 Through 18-12-2020
13. Moussallem, D., Ngonga Ngomo, A.C., Buitelaar, P., Arcan, M.: Utilizing knowledge graphs for neural machine translation augmentation. In: Proceedings of the 10th International Conference on Knowledge Capture. pp. 139–146 (2019)
14. Moussallem, D., Wauer, M., Ngomo, A.C.N.: Machine translation using semantic web technologies: A survey. Journal of Web Semantics **51**, 1–19 (2018). https://doi.org/10.1016/j.websem.2018.07.001
15. Ngonga Ngomo, A.C., Sherif, M.A., Georgala, K., Hassan, M., Dreßler, K., Lyko, K., Obraczka, D., Soru, T.: LIMES - A Framework for Link Discovery on the Semantic Web. KI-Künstliche Intelligenz, German Journal of Artificial Intelligence - Organ des Fachbereichs "Künstliche Intelligenz" der Gesellschaft für Informatik e.V. (2021), `https://papers.dice-research.org/2021/KI_LIMES/public.pdf`
16. Papineni, K., Roukos, S., Ward, T., Zhu, W.J.: BLEU: a method for automatic evaluation of machine translation. In: Proceedings of the 40th annual meeting on Association for Computational Linguistics (2002)
17. Popović, M.: chrF++: words helping character n-grams. In: Proceedings of the Second Conference on Machine Translation. pp. 612–618 (2017)
18. Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., Liu, P.J.: Exploring the limits of transfer learning with a unified text-to-text transformer. J. Mach. Learn. Res. **21**, 140:1–140:67 (2020), `http://jmlr.org/papers/v21/20-074.html`
19. Reiter, E., Dale, R.: Building natural language generation systems. Cambridge University Press, New York, NY, USA (2000)
20. Sherif, M., Ngonga Ngomo, A.C., Lehmann, J.: WOMBAT - A Generalization Approach for Automatic Link Discovery. Springer (2017)
21. Snover, M., Dorr, B., Schwartz, R., Micciulla, L.: A study of translation edit rate with targeted human annotation (2006)
22. Sutskever, I., Vinyals, O., Le, Q.V.: Sequence to sequence learning with neural networks (2014)
23. Sutskever, I., Vinyals, O., Le, Q.V.: Sequence to sequence learning with neural networks. In: Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N.D., Weinberger, K.Q. (eds.) Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada. pp. 3104–3112 (2014), `https://proceedings.neurips.cc/paper/2014/hash/a14ac55a4f27472c5d894ec1c3c743d2-Abstract.html`
24. Usbeck, R., Röder, M., Ngonga Ngomo, A.C., Baron, C., Both, A., Brümmer, M., Ceccarelli, D., Cornolti, M., Cherix, D., Eickmann, B., Ferragina, P., Lemke, C., Moro, A., Navigli, R., Piccinno, F., Rizzo, G., Sack, H., Speck, R., Troncy, R., Waitelonis, J., Wesemann, L.: Gerbil: General entity annotator benchmarking framework. In: Proceedings of the 24th International Conference

on World Wide Web. p. 1133–1143. WWW '15, International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, CHE (2015). https://doi.org/10.1145/2736277.2741626
25. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I.: Attention is all you need (2017)