Neural Wikipedian: Generating Textual Summaries from Knowledge Base Triples

Pavlos Vougiouklis[a], Hady Elsahar[b], Lucie-Aimée Kaffee[a], Christophe Gravier[b], Frédérique Laforest[b], Jonathon Hare[a], Elena Simperl[a]

[a]*School of Electronics and Computer Science*
University of Southampton
Southampton, United Kingdom
[b]*Laboratoire Hubert Curien, CNRS*
UJM-Saint-Étienne
Université de Lyon,
Lyon, France

Abstract

Most people need textual or visual interfaces in order to make sense of Semantic Web data. In this paper, we investigate the problem of generating natural language summaries for Semantic Web data using neural networks. Our end-to-end trainable architecture encodes the information from a set of triples into a vector of fixed dimensionality and generates a textual summary by conditioning the output on the encoded vector. We explore a set of different approaches that enable our models to verbalise entities from the input set of triples in the generated text. Our systems are trained and evaluated on two corpora of loosely aligned Wikipedia snippets with triples from DBpedia and Wikidata, with promising results.

Keywords: Natural Language Generation, Neural Networks, Semantic Web Triples

1. Introduction

While Semantic Web data, such as triples in Resource Description Framework (RDF), is easily accessible by machines, it is difficult to be understood by people who are unfamiliar with the underlying technology. On the contrary, for humans, reading text is a much more accessible activity. In the context of the Semantic Web, Natural Language Generation (NLG) is concerned with the implementation of textual interfaces that would make the information that is stored in the knowledge bases' triples more accessible to the potential users. Further development of NLG systems could be beneficial in a great range of application domains. A typical application is their integration in Question Answering platforms, whose users' experience could be improved by the ability to automatically generate a textual description of an entity that is returned at a user's query (e.g. the Google Knowledge Graph[1] and the Wikidata Reasonator[2]). Another example is dialogue systems in commercial environments. These systems can be enhanced further by NLG components capable of generating responses that better address the users' questions [1].

So far, research has mostly focused on adapting rule-based approaches to generate text from Semantic Web data. These systems work in domains with small vocabularies and restricted linguistic variability, such as football match summaries [2] and museum exhibit descriptions [3]. However, the difficulty of transferring the involved rules across different domains or languages along with the tedious repetition of their textual patterns prevented them from becoming widely accepted [4].

We address these limitations by proposing a statistical model for NLG using neural networks. Our work explores how an adaptation of the encoder-decoder framework [5, 6] could be used to generate textual summaries for triples. More specifically, given a set of triples about an entity (i.e. the entity appears as the subject or the object of the triples), our task consists in summarising them in the form of comprehensible text. We propose a model that consists of a feed-forward architecture that encodes each triple from an input set of triples in a vector of fixed dimensionality in a continuous semantic space, and an RNN-based decoder that generates the textual summary one word at a time. Our model jointly learns unique vector representations "embeddings" for entities and words that exist in the text, and predicates and entities as they occur in the corresponding triples. In contrast with less flexible, rule-based strategies for NLG, our approach does not constrain the number of potential relations between the triples' predicates and the generated text. Consequently, a learnt predicate embedding, given its position in the semantic space,

Email addresses: pv1e13@ecs.soton.ac.uk (Pavlos Vougiouklis), hady.elsahar@univ-st-etienne.fr (Hady Elsahar), kaffee@soton.ac.uk (Lucie-Aimée Kaffee), christophe.gravier@univ-st-etienne.fr (Christophe Gravier), frederique.laforest@univ-st-etienne.fr (Frédérique Laforest), jsh2@ecs.soton.ac.uk (Jonathon Hare), e.simperl@soton.ac.uk (Elena Simperl)

[1] https://googleblog.blogspot.co.uk

[2] https://tools.wmflabs.org/reasonator

can be expressed in an varied number of different ways in the text.

Our proposed model learns to generate a textual summary as a sequence of words and entities. We experiment with two different approaches, one rule-based, and one statistical, in order to infer the verbalisation of the predicted entities in the generated summary. Conventional systems based on neural networks when employed on text generative tasks, such as Machine Translation [6] or Question Generation [7] are incapable of learning high quality vector representations for the infrequent tokens (i.e. either words or entities) in their training dataset. Inspired by [7, 8], we address this problem by adapting a multi-placeholder method that enables the model to emit special tokens that map a rare entity in the text to its corresponding triple in the input set.

In order to both train and evaluate the performance of our model, we propose an automatic approach for building a large data-to-text corpus of rich linguistic variability. Training data for NLG models is not always readily available; this applies to Semantic Web scenarios as well. The difficulty is that data that is available in knowledge bases needs to be aligned with the corresponding texts. Existing solutions for data-to-text generation either focus mainly on creating a small, domain-specific corpus where data and text are manually aligned by a small group of experts, such as the WeatherGov [9] and RoboCup [10] datasets, or rely heavily on crowdsourcing [11, 12], which makes them costly to apply for large domains. We rely on the alignment of DBpedia and Wikidata with Wikipedia in order to create two corpora of knowledge base triples from DBpedia and Wikidata, and their corresponding textual summaries. For the purpose of this paper, we chose to retrieve articles about people and their biographies [13]. We extracted two different corpora with vocabularies of over 400k words that consist of: (i) 260k Wikipedia summaries aligned with a total of 2.7M DBpedia triples, and (ii) 360k Wikipedia summaries allocated to a total of 4.3M Wikidata triples.

We use perplexity, and the BLEU and ROUGE metrics in order to automatically evaluate our approach's ability of predicting the Wikipedia summary that corresponds to a set of unknown triples showing substantial improvement over our baselines. Furthermore, we evaluate a set of generated summaries against human evaluation criteria. Based on the average rating across our selected criteria, we conclude that our approach is able to generate coherent textual summaries that address most of the information that is encoded in the input triples. Lastly, we demonstrate that our method can infer semantic relationships among entities by computing the nearest neighbours of the learned embeddings of the entities in our datasets.

The structure of the paper is as follows. Section 2 discusses existing NLG approaches for the Semantic Web, and relates them to our model. Section 3 presents the components of our approach. Section 4 describes the construction of our datasets. Section 5 presents experiments and the evaluation of the systems. Section 6 summarises the contributions of this work and outlines future plans.

2. Related Work

NLG systems typically work in three different stages: (i) document planning or content selection, (ii) microplanning, and (iii) surface realisation [4, 14]. During document planning the information that will be communicated in the text is selected and organised (i.e. document structuring). The output of the document planner is used by the microplanner to decide how this information should be linguistically expressed in the generated text. Subsequently, the realiser generates the actual text that satisfies the linguistic requirements that are set by the microplanner, and expresses the information as it was structured by the document planner. While in conventional text generation systems that relied on rules these phases were performed independently, they were associated not only with the domain and the language of the end-application but, in many cases, with the application itself [15, 16, 17]. More recently, data-driven approaches which "learn" to perform content selection and realisation under a single framework have been proposed [18, 19].

Neural network approaches have been applied to a wide variety of NLP tasks ranging from machine translation [5, 6] and automatic response generation [20, 21, 22, 23] to semantic constituency parsing [24], and generation of textual descriptions from visual data [25, 26]. Our approach is inspired by the general encoder-decoder framework [5, 6] with multi-gated Recurrent Neural Network (RNN) variants, such as the Gated Recurrent Unit (GRU) [5] and the Long Short-Term Memory (LSTM) cell [27]. Adaptations of this framework have demonstrated state-of-the-art performance in many generative tasks, such as machine translation [5, 6], and conversation modelling and response generation [20, 28].

Most of the previous work on NLG with Semantic Web data has focused on the verbalisation from domain ontologies using hand-coded rules. Examples include systems that generate text in domains with limited linguistic variability, such as clinical narratives [29], summaries of football matches [2], and, descriptions of museums' exhibits [3]. Further Semantic Web-oriented NLG applications can be found in [4]. Our work naturally lies on the path opened by recent unsupervised [30] and *distant-supervision* [31] based approaches for the extraction of RDF verbalisation templates using parallel data-to-text corpora. However, rather than making a prediction about the template that would be the most appropriate to verbalise a set of input triples, our model jointly performs content selection and surface realisation, without the inclusion of any rules or templates. More recently, adaptation of out-of-the-box Neural Machine Translation models have shown great potential at tackling various aspects of triples-to-text tasks ranging from microplanning [32] to generation of paraphrases [33].

Implementations based on the encoder-decoder framework work by mapping sequences of source tokens to sequences of target tokens. We adapt this Sequence-to-Sequence model to the requirements of Semantic Web data. Since sets of triples that are given to our systems as an input are unordered, and not sequentially correlated, in the next section we propose a model that consists of a feed-forward neural network that encodes each triple from a set into a vector of fixed dimensionality in a continuous semantic space. Within this space, triples that have similar semantic meaning will have similar positions. We couple this novel encoder with an RNN-based decoder that generates the textual summary one token at a time (i.e. a token can be a word or an entity or a surface form of an entity).

Our task is most similar to recent work by Lebret et al. and Chisholm et al., who both employ adaptations of the encoder-decoder framework to generate the first sentence of a Wikipedia biography [13, 34]. Lebret et al. propose a system that generates a summary from a Wikipedia infobox. The model proposed by Chisholm et al. generates a biography given a sequence of slot-value pairs extracted from Wikidata. In both cases, the representation of the input is essentially limited to expressing only one-subject relationships. In our case, the input set of triples that is allocated to each Wikipedia summary is made of more than just the DBpedia or Wikidata triples of the corresponding Wikipedia article. As we discuss in more detail in Section 4.2, the input also includes triples with entities that are related to the main entity of a Wikipedia biography in the respective knowledge base, and their object is the main subject of the Wikipedia summary. An example of an input triples set that we use in our task is displayed in Table 1. The first two triples that share the same subject (i.e. `dbr:Walt_Dinsey`) come from the DBpedia triples that are allocated to the article of *Walt Disney*; the latter one comes from the DBpedia triples of `dbr:Mickey_Mouse`, and is part of the input set since its object is the main entity of the original triples extracted for Walt Disney. Furthermore, we believe that constraining the generative process to only the first sentence significantly simplifies the task in terms of the amount of information (i.e. in our case number of triples) that is lexicalised in the summary. Consequently, we choose to train on longer snippets of text to generate more elaborate summaries.

3. Our Model

An idealised example of our NLG task is presented in Table 1; our system takes as an input a set of triples about the entity *Walt Disney* (i.e. the entity *Walt Disney* is either the subject or object of the triples in the set), and generates a sequence of words in order to summarise them in the form of natural language text. Given a set of E triples, $F = \{f_1, f_2, \ldots, f_E\}$, our goal is to learn a model that is able to generate a sequence of T tokens, $Y = y_1, y_2, \ldots, y_T$. We regard Y as a representation in natural language of the input set of triples, and build a model that computes the probability of generating $y_1, y_2, \ldots, y_T$, given the initial set of triples $f_1, f_2, \ldots, f_E$:

$$p(y_1, \ldots, y_T | f_1, f_2, \ldots, f_E) = \prod_{t=1}^{T} p(y_t | y_1, \ldots y_{t-1}, F) \quad . \tag{1}$$

Table 1: An idealised example of our NLG task. Our system takes as an input a set of triples about *Walt Disney*, whose either subject or object is related to the entity of Walt Disney, and generates a textual summary.

Triples	`dbr:Walt_Disney dbo:birthDate ''1901-12-05''`
	`dbr:Walt_Disney dbo:birthPlace dbr:Chicago`
	`dbr:Mickey_Mouse dbo:creator dbr:Walt_Disney`

Textual Summary	Walt Disney was born in Chicago, and was the creator of Mickey Mouse.

Our model consists of a feed-forward architecture that encodes each triple from the input set into a vector of fixed dimensionality in a continuous semantic space. This is coupled to an RNN-based decoder that generates the textual summary one token at a time (i.e. a token can be a word or an entity or a surface form of an entity). The architecture of our generative model is shown in Figure 1. Note that since *bias* terms can be included in each weight-matrix multiplication, they are not explicitly shown in the equations of this section [35].

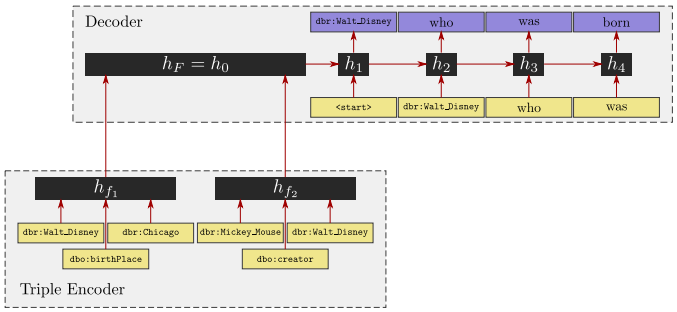


Figure 1: The triple encoder computes a vector representation, h_{f_1} and h_{f_2}, for each one of the two input triples. Subsequently, the decoder uses the concatenation of the two vectors, $[h_{f_1}; h_{f_2}]$ to initialise the decoding process that generates the summary, token by token. Each textual summary starts and ends with the respective start-of-sequence `<start>` and end-of-sequence `<end>` tokens. At each timestep t, the decoder takes as an input the current word and hidden state of the previous timestep $t-1$, and makes a prediction about the next token that should be appended in the summary. For example in the second timestep, the decoder takes as an input the `dbr:Walt_Disney` token and the previous hidden state h_1 and predicts the token (i.e. in this particular scenario *who*) that should come next.

3.1. Triple Encoder

Let $F = \{f_1, \ldots, f_E : f_i = (s_i, p_i, o_i)\}$ be the set of triples $f_1, \ldots, f_E$, where s_i, p_i and o_i are the one-hot[3] vector representations of the respective subject, predicate and object of the i-th triple. The vector representation h_{f_i} of the i-th triple is computed by forward propagating the triples encoder as follows:

$$\widetilde{h_{f_i}} = [\mathbf{W}_{\mathbf{x}\rightarrow\widetilde{\mathbf{h}}} s_i; \mathbf{W}_{\mathbf{x}\rightarrow\widetilde{\mathbf{h}}} p_i; \mathbf{W}_{\mathbf{x}\rightarrow\widetilde{\mathbf{h}}} o_i] \ , \quad (2)$$

$$h_{f_i} = \mathrm{ReLU}(\mathbf{W}_{\widetilde{\mathbf{h}}\rightarrow\mathbf{h}} \widetilde{h_{f_i}}) \ , \quad (3)$$

where ReLU is the rectifier (i.e. non-linear activation function), $[\ldots;\ldots]$ represents vector concatenation, $\mathbf{W}_{\mathbf{x}\rightarrow\mathbf{h}} : \mathbb{R}^{|N|} \rightarrow \mathbb{R}^m$ is a trainable weight matrix that represents an unbiased linear mapping, where $|N|$ is the cardinality of all the potential one-hot input vectors (i.e. size of the dictionary of all the available predicates and entities of the triples dictionary), and $\mathbf{W}_{\widetilde{\mathbf{h}}\rightarrow\mathbf{h}} : \mathbb{R}^{3m} \rightarrow \mathbb{R}^m$ is an unbiased linear mapping.

3.2. Decoder

After the vector representation h_{f_i} for each triple f_i is obtained, we start the decoding process during which the corresponding textual summary is generated. At each timestep t, the decoder makes a prediction about the next token that will be appended to the summary by taking into consideration both the tokens that have already been generated, and the contextual knowledge from the triples that have been provided to the system as input. We experiment with two commonly used multi-gated RNN variants: (i) the LSTM cell and (ii) the GRU, in order to explore which one works best for the requirements of our architecture. The main advantage of using multi-gated units is their ability to process information from much longer sequences compared to the *simple* RNN [36].

We initialise the decoder with a fixed-length vector that we obtain after encoding all the information from the vector representations of the triples. Our approach is inspired by the general Sequence-to-Sequence framework, within which an RNN-based encoder encapsulates the information that exists in a sequence, and an RNN-based decoder generates a new sequence from this encapsulation [5, 6]. However, since the triples that we use in our problem are not sequentially correlated, we propose a concatenation-based formulation in order to capture the information across all the triples that are given as an input to our system into one single vector. More specifically, given a set of triples' vector representations, $h_{f_1}, \ldots, h_{f_E}$, we compute:

$$\widetilde{h_F} = [h_{f_1}; h_{f_2}; \ldots; h_{f_{E-1}}; h_{f_E}] \ , \quad (4)$$

$$h_F = \mathbf{W}_{\mathbf{h_F}\rightarrow\mathbf{h_0^1}} \widetilde{h_F} \ , \quad (5)$$

where $\mathbf{W}_{\mathbf{h_F}\rightarrow\mathbf{h_0^1}} : \mathbb{R}^{Em} \rightarrow \mathbb{R}^m$ is a biased linear mapping. Subsequently, the hidden units of the LSTM- or GRU-based decoder (discussed below) at layer depth $l = 1$ are initialised with $h_0^1 = h_F$.

Let $h_t^l \in \mathbb{R}^m$ be the aggregated output of a hidden unit at timestep $t = 1...T$ and layer depth $l = 1...L$. The vectors at zero layer depth, $h_t^0 = \mathbf{W}_{\mathbf{x}\rightarrow\mathbf{h}} x_t$, represent the words or entities that are given to the network as an input. The parameter matrix $\mathbf{W}_{\mathbf{x}\rightarrow\mathbf{h}}$ has dimensions $[|X|, m]$, where $|X|$ is the cardinality of all the potential one-hot input vectors (i.e. size of the dictionary of all the available words and entities of the textual summaries dictionary). All subsequent matrices have dimension $[m, m]$ unless stated otherwise.

3.2.1. Long Short-Term Memory (LSTM).

We adopt the architecture from [37]:

$$\begin{pmatrix} in_t^l \\ f_t^l \\ out_t^l \\ \widetilde{c_t^l} \end{pmatrix} = \begin{pmatrix} \mathrm{sigm} \\ \mathrm{sigm} \\ \mathrm{sigm} \\ \mathrm{sigm} \end{pmatrix} \mathbf{W^l} \begin{pmatrix} h_t^{l-1} \\ h_{t-1}^l \end{pmatrix} \ , \quad (6)$$

$$c_t^l = f_t^l \odot c_{t-1}^l + in_t^l \odot \widetilde{c_t^l} \ , \quad (7)$$

$$h_t^l = out_t^l \odot \tanh(c_t^l) \ , \quad (8)$$

where $\mathbf{W^l} : \mathbb{R}^{4m} \rightarrow \mathbb{R}^{2m}$ is a biased linear mapping, and in_t^l, f_t^l, out_t^l and c_t^l are the vectors at timestep t and layer depth l that correspond to the *input gate*, the *forget gate*, the *output gate* and the *cell* respectively. The cell state c_t^l represents each LSTM cell's internal memory. The information that will be stored in each c_t^l is regulated by the three multiplicative gates (i.e. input, forget and output gate). The input gate in_t^l is used in order to protect the cell state from any irrelevant inputs. Similarly, the output gate out_t^l is introduced to protect any LSTM cells at further timesteps or higher layers from irrelevant information stored in the current c_t^l. Finally, the forget gate f_t^l determines which part of the previous cell state c_{t-1}^l is ignored in the computation of the current cell state c_t^l.

3.2.2. Gated Recurrent Unit (GRU).

The GRU [5] is a less complex variant of the LSTM cell with comparable performance [36].

$$\begin{pmatrix} r_t^l \\ u_t^l \end{pmatrix} = \begin{pmatrix} \mathrm{sigm} \\ \mathrm{sigm} \end{pmatrix} \mathbf{W^l} \begin{pmatrix} h_t^{l-1} \\ h_{t-1}^l \end{pmatrix} \ , \quad (9)$$

$$\widetilde{h_t^l} = \tanh(\mathbf{W_{in}^l} h_t^{l-1} + \mathbf{W_{h\rightarrow h}^l}(r_t^l \odot h_{t-1}^l)) \ , \quad (10)$$

$$h_t^l = (1 - u_t^l) \odot h_{t-1}^l + u_t^l \odot \widetilde{h_t^l} \ , \quad (11)$$

[3] One-hot is a vector that contains a 1 at the index of this particular s_i, p_i and o_i token in the vocabulary with all the other values set to zero.

where $\mathbf{W^l} : \mathbb{R}^{2m} \rightarrow \mathbb{R}^{2m}$ is a biased linear mapping, and r_t^l, u_t^l and $\widetilde{h}_t^l$ are the vectors at timestep t and layer depth l that represent the values of the *reset gate*, the *update gate* and the *candidate hidden state* respectively. In contrast to the LSTM cell, the GRU is essentially absolved from an explicit internal memory mechanism. At each timestep, the reset gate r_t^l learns to determine how much of the previous hidden state h_{t-1}^l should be ignored for the computation of the candidate hidden state $\widetilde{h}_t^l$. In a similar fashion, the update gate u_t^l learns to decide what part of h_{t-1}^l will be leaked into the computation of the current hidden state h_t^l.

3.3. Model Training

The conditional probability distribution over the each token of the summary at each timestep t is represented with the softmax function over all the entries in the textual summaries dictionary:

$$p(y_t|y_1, \dots y_{t-1}, F) = \text{softmax}(\mathbf{W_y} h_t^L) \ , \qquad (12)$$

where $\mathbf{W_y} : \mathbb{R}^m \rightarrow \mathbb{R}^{|X|}$ is a biased trainable weight matrix. Our model learns to make a prediction about the next token by using the negative cross-entropy[4] criterion. During training and given a set of triples, the model predicts the sequence of tokens of which the generated summary is comprised. The model computes how far the generated sequence of tokens is from the empirical, actual text by utilising the negative logarithmic probability of the generated summary given set of triples:

$$\text{cost} = -\sum_{t=1}^{T} \log \ p(y_t|y_1, \dots y_{t-1}, F) \ . \qquad (13)$$

Consequently, our model tries to minimise the above cost function. This non-convex optimisation problem is solved using Back-Propagation [38] with a dynamic learning rate update provided by the RMSProp[5] algorithm.

3.4. Generating Summaries

During testing, our goal is to find:

$$\mathbf{y}^* = \arg\max_{\mathbf{y}} \sum_{t=1}^{T} \log \ p(y_t|y_1, \dots y_{t-1}, F) \ , \qquad (14)$$

where $\mathbf{y}^*$ is the optimal summary computed by the model. Recall from Eq. 12 that at each timestep, the model predicts a probability distribution over the token that is more likely to come next. In theory, Viterbi decoding could approximate an optimal summary. However, in practice, the fact that the target vocabulary $|X|$ is large enough deems such an approach intractable [40]. A different approach is to approximate the best summary by appending the token with the highest probability at each timestep of the generation process. Even though such greedy decoders have proven to be very fast when employed in machine translation problems, they tend to produce low quality approximations [40].

A compromise between a strictly-greedy decoding algorithm and Viterbi is to adopt a beam-search decoder [6, 40], which provides us with the B-most-probable summaries (or hypotheses) given a set F of input triples. The decoder maintains only a small number of B hypotheses (i.e. partially completed summaries), which it extends at every timestep with every token in the target vocabulary $|X|$.

During testing, we provide our network with an unknown set of triples, and initialise the decoder with a special start-of-summary `<start>` token. The B tokens with the highest probability are used as separate inputs to the decoder at the second timestep. This leads to $B|X|$ partial hypotheses from which we only retain the B-best. After all the second words of our hypotheses are inputted to the decoder, we end up with $B|X|$ partial three-worded hypotheses from which again we only keep the B ones with the highest probability. When the end-of-summary `<end>` token is predicted for a hypothesis, it is appended to the list of complete summaries, and the process carries on with $B = B-1$. A simplified example of a beam-search decoder with a beam B of size 2, when the set F of input triples consists of only a single triple ($|F| = 1$ and $E = 1$) is displayed in Figure 2.

4. Datasets

In order to train our proposed model, we build two datasets of aligned knowledge base triples with texts. For the first dataset (D1), we leverage the intrinsic alignment of DBpedia and Wikipedia in order to create a corpus of loosely aligned triples and textual summaries. For the second (D2), we align Wikipedia summaries with the community-curated triples of Wikidata.

Inspired by Lebret et al. and Chisholm et al., we chose a corpus about biographies. Biographies represent one of the largest single domains in Wikipedia, providing us with a substantial amount of training data. While arguably Wikipedia biographies tend to adopt a limited number of structural paradigms, they are still a dataset of rich linguistic variability with challenging vocabulary sizes of greater than 400k words (cf. Table 4). Their linguistic variability is also supported by the fact that our D1 and D2 datasets contain summaries (95th percentile) whose main entities are of 45 and 44 different instance types respectively. Consequently, we believe that biographies offer a good trade-off between diversity of instance types

[4] In information theory, the entropy H is a measure of the uncertainty. The concept of cross-entropy is associated with the similarity between two distributions, an empirical one q and a predicted one p given a random variable X and set of parameters θ. It is defined as: $H(X) = -\sum q(y^{(i)}) \log p(y^{(i)}|x^{(i)}, \theta)$.

[5] RMSProp stands for Root Mean Square Propagation, and is a form of stochastic gradient descent where the gradient for each weight is divided by a running average of its recent gradients norm [39].

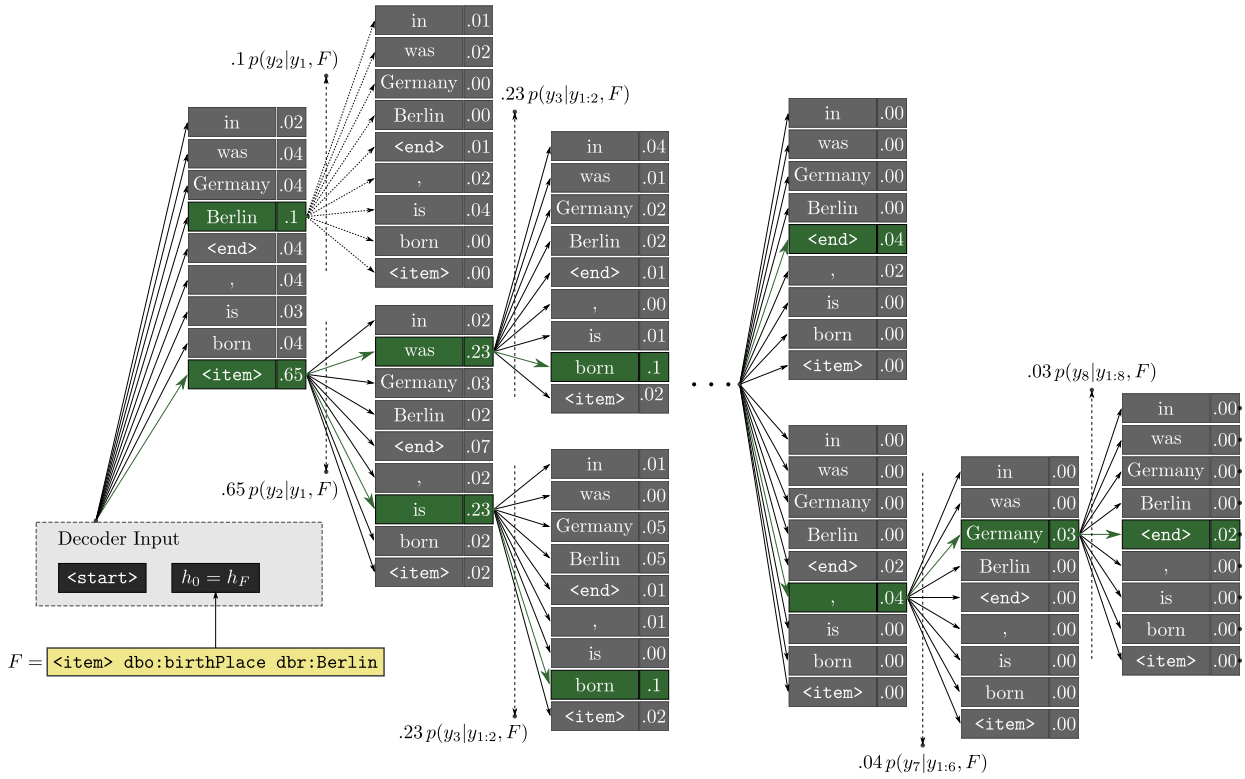


Figure 2: A simplified example of a beam-search decoder with a beam B of size 2 and target vocabulary size $|X|$ equal to 9. After the vector representation h_F for the whole triples' set is computed, it is provided as input to the decoder along with the `<start>` token. The scores at the right-hand side of the words in the vocabulary is the probability of the summary when it is extended by that particular word. At the first timestep, the decoder retains the two most-probable words with which a summary given this particular input set could begin. Subsequently, *Berlin* and `<item>` (i.e. `<item>` is a special token that is described in more detail in Sections 4.3.1 and 4.3.2) are inputted to the decoder separately at the second timestep. This results in 18 partial hypotheses. We keep only the two most probable summaries, and since both of them start with the `<item>` token, no summaries with *Berlin* as their first word are retained. At the third timestep, *is* and *was* are provided as separate inputs to the decoder. The process continues in a similar fashion until the seventh timestep, when the special `<end>` token is predicted for the one of the partial hypothesis. This hypothesis is then appended to the list of complete hypotheses and the beam search continues with beam of size 1, until the `<end>` token is predicted for the other partial hypothesis. At the end the list of complete hypotheses is formed as follows: (a) "`<start> <item>` was born in Berlin `<end>`" with a probability of 0.04 and (b) "`<start> <item>` is born in Berlin, Germany `<end>`" with a probability of 0.02.

and regular structure allowing us to explore the strengths and limitations of our purely data-driven approach. Table 2 presents the distribution of the 10 most frequent instance types of the main discussed entities of our corpora. Please note that in the case of D2, the Wikidata entities are mapped to their respective DBpedia ones using the sameas-all-wikis[6] dataset.

We used PetScan[7] to collect a list of 1,479,170 Wikipedia articles that have been curated by the WikiProject Biography[8]. We then extracted the Wikipedia summaries and their corresponding DBpedia triples from the Mapping-based Objects[9] (DB2) and Literals[9] DBpedia dataset (DB3), retaining only articles for which an infobox exists. For the second dataset, we used the Wikidata truthy dumps[10] (WD1) and we kept only items for which Wikidata triples exist. All relevant Wikipedia summaries were extracted using the Long Abstracts[9] DBpedia dataset (DB1).

In addition to the above datasets, we also leverage two DBpedia datasets: (i) the Instance Types[9] (DB4) and (ii) the Genders[9] (DB5) datasets. The first one is used to map the entities that occur infrequently in our aligned datasets to special tokens, and the second in order for us to append a gender-related triple to the DBpedia triples that have been already allocated to an article. Since co-reference resolution is not performed as a data pre-processing stage, our hypothesis is that the additional knowledge from the inclusion of gender-related triples will increase the model's awareness towards the gender of the main discussed entity of an article. Please note that the Genders dataset is used for the DBpedia version of the aligned dataset, in which gender-related triples are extremely sparse. In summary, the datasets that we built

[6] `http://wikidata.dbpedia.org/downloads/20160111`

[7] PetScan (i.e. `petscan.wmflabs.org`) is a tool that identifies Wikipedia articles, images and categories based on the category or subcategory to which they belong.

[8] `en.wikipedia.org/wiki/Wikipedia:WikiProject_Biography`

[9] `http://wiki.dbpedia.org/downloads-2016-10`

[10] `https://dumps.wikimedia.org/wikidatawiki/entities`

Table 2: Distribution of the 10 most frequent DBpedia instance types of the main discussed entities of our two corpora, D1 (i.e. based on DBpedia triples) and D2 (i.e. based on Wikidata triples).

D1		**D2**	
Instance Type	%	Instance Type	%
dbo:Person	26.48	owl#Thing	19.87
dbo:MusicalArtist	9.65	dbo:Person	17.48
dbo:OfficeHolder	6.75	dbo:MusicalArtist	6.57
dbo:Band	4.86	dbo:OfficeHolder	4.31
dbo:Writer	3.78	dbo:Band	3.18
dbo:SoccerPlayer	3.64	dbo:Writer	2.96
dbo:MilitaryPerson	2.98	dbo:SoccerPlayer	2.37
dbo:Scientist	2.87	dbo:MilitaryPerson	2.03
dbo:Artist	2.06	dbo:Scientist	1.90
dbo:Royalty	1.76	dbo:Artist	1.47

along with their intermediate components are enumerated below:

D1 DBpedia triples aligned with Wikipedia biographies; its intermediate components are listed below:

- Long Abstracts[9] DBpedia dataset (DB1)
- Mapping-based Object[9] DBpedia dataset (DB2)
- Mapping-based Literals[9] DBpedia dataset (DB3)
- Instance Types[9] DBpedia dataset (DB4)
- Genders[9] DBpedia dataset (DB5)

D2 Wikidata triples aligned with Wikipedia biographies that has been formed from:

- Long Abstracts[9] DBpedia dataset (DB1)
- Wikidata truthy dumps[10] (WD1)
- Instance Types[9] DBpedia dataset (DB4)

4.1. Wikipedia Summaries

One of the main challenges that is associated with the alignment of triples from a structured knowledge base with text is the identification of how the entities of the knowledge base are mentioned in the text. For instance in the Wikipedia sentence: "Barack Hussein Obama II is an American politician who served as the 44th President of the United States from 2009 to 2017."[11], we need to be able to identify that the surface forms of "Barack Hussein Obama II" and "United States" refer to the respective DBpedia resources of `dbr:Barack_Obama` and `dbr:United_States`. In order to sidestep this problem, we use DBpedia Spotlight [41], an automatic system for annotation of DBpedia entities in text. *Confidence* and *support* are the two main variables that parameterise the annotation results that are returned by DBpedia Spotlight. Confidence refers to the lowest threshold of certainty which the system must have in order to return an annotation. Support is the lowest bound of the un-normalised total number of links to the returned entities.

We run each Wikipedia summary through DBpedia Spotlight. Our goal was to find the combination of *confidence* and *support* that provides the greatest number of relevant annotations, in order to (i) enhance the set of triples allocated to each Wikipedia page, and (ii) allow the model to learn directly how entities in the triples on the encoder side manifest themselves in the text on the decoder side. We empirically found that by setting the *confidence* and *support* parameters to 0.35 and −1 respectively, we increased the recall of the identified entities while maintaining precision at acceptable levels. We retained a list of all the possible surface forms to which each entity was mapped. Furthermore, we excluded any Wikipedia summaries whose main discussed entity was not identified in the text.

Each Wikipedia summary is tokenised and split into sentences using the Natural Language Toolkit (NLTK) [42].

Table 3: An example of how a triple whose object is identified as a date is encoded into two different triples. The first one represents the month that has been identified in the original triple, and the second the year.

Original Triple	dbr:Andre_Agassi dbo:birthDate ''1970-04-29''
Resultant Triples	dbr:Andre_Agassi dbo:birthDateMonth 4 dbr:Andre_Agassi dbo:birthDateYear <year>

4.2. Knowledge Base Triples

Our text generation task consists of learning how entities, along with their relationships, are mentioned in the text. Given a set of triples, our approach learns to generate text one token at a time, without constraining the generative procedure to pre-defined templates that would include a given textual string *as-it-is* in the generated summary. Consequently, we excluded from our corpora any triples with a textual string as their object, except those that referred to numbers, dates or years. All instances of number-objects are replaced with the special token `0`, except for year-objects that are mapped to the special `<year>` token [13]. In both Wikidata and DBpedia, date-related objects are expressed as a string followed by its corresponding XML Schema URI (e.g. `XMLSchema/-#dateTime` or `XMLSchema/#date`). In order to enable our model to process date-related triples and learn how their information is lexicalised in the text, we decompose them into two different triples. The first one is used to represent the month as it has been identified in the original triple, and the second one to represent the year. The object of the latter is subsequently mapped to the special `<year>` token. Table 3 presents an example of our date encoding approach.

[11] `https://en.wikipedia.org/wiki/Barack_Obama`

For each entity that has been identified in a Wikipedia summary using DBpedia Spotlight, we extracted its corresponding triples from the Mapping-based Objects dataset in the DBpedia's case, and the Wikidata truthy dump in Wikidata's case. We assume that the subjects or objects of a set of triples are consistent with the main subject of the corresponding Wikipedia summary. Consequently, from this additional set of triples, we only retain those whose object matches the main discussed entity in each summary, and we append them to the initial set. This results in 450 and 609k unique predicates and entities in DBpedia's case and in 378 and 278k unique predicates and entities respectively in Wikidata's case.

4.3. Aligned Dataset

We built two aligned datasets that consist of: (i) 256850 instances of Wikipedia summaries aligned with 2.74M DBpedia triples, and (ii) 358908 instances of Wikipedia summaries aligned with the total of 4.34M Wikidata triples. The size difference is explained as follows. First, there are Wikipedia biographies without an infobox (i.e. and, thus, without any available triples in the Mapping-based Objects and Literals DBpedia datasets). Second, even when they do have an infobox, the retrieved triples that are made available in the DBpedia dumps might not meet the requirements of our task (i.e. Section 4.2). For example, we exclude a Wikipedia summary from an aligned dataset, in case the objects of all the triples that are allocated to it are strings other than dates or numbers.

We describe next all the pre-processing steps that we followed in order to make our aligned datasets fit for the training of our neural network architectures.

4.3.1. Modelling the Generated Summaries

We retained only the first two sentences of each summary in order to reduce the computational cost of our task; summaries that consist of only one sentence were included unaltered. Since it would be impossible to learn a unique vector representation for the entity of interest of each Wikipedia summary due to the lack of occurrences of the majority of those entities in the datasets, we replaced them with the special `<item>` token. We used a fixed vocabulary of 30000 and 32000 of the most frequent tokens (i.e. either words or entities) of the summaries that are aligned with the DBpedia and Wikidata triples. Similarly to the input triples (i.e. Section 4.2), all occurrences of numbers in the text are replaced with the special token `0`, except for years that are mapped to the special `<year>` token [13]. Every out-of-vocabulary word is represented by the special `<rare>` token.

Due to the nature of our task, we are required to handle both words and entities that occur infrequently in the textual summaries. For every out-of-vocabulary word, we use the special `<rare>` token. However, using a single special token for all the rare entities that have not been included in the fixed target vocabulary would substantially limit the model, causing unnecessary repetition of this particular token in the generated summaries. Inspired by the Multi-Placeholder model [7], we attempt to match a rare entity that has been annotated in the text, in the subjects or the objects of the allocated triples. In case the rare entity exists in the triples, then it is replaced by a placeholder token, which consists of the predicate of the triple, a descriptor of the component of the triple that was matched (i.e. `__obj__` or `__subj__`), and the instance type of the entity. The instance type of an entity is obtained from the Instance Types dataset. For example, if the subject of the triple: (`dbr:Atlas_Shrugged dbo:author dbr:Ayn_Rand`) is annotated as a rare entity in the corresponding summary, it is replaced with the subject-related placeholder: `dbo:author__sub__dbo:Book`. If a rare entity is matched to the object of the triple: (`Michael_Jordan dbo:birthPlace dbr:Brooklyn`), it is replaced with the appropriate object-related placeholder: `dbo:birthPlace__obj__dbo:City`. We refer to those placeholders as *property-type placeholders*. In case the entity does not have a type in the Instance Types dataset, the instance type part of the placeholder is filled by the `<unk>` token (e.g. `dbo:birthPlace__obj__<unk>`). If the rare entity is not matched to any subject or object of the set of corresponding triples, then it is replaced by the special token of its instance type. In case the rare entity does not exist in the instance types dataset, it is replaced by the `<unk>` token. The property-type placeholders, enable our systems to verbalise rare entities in the text by replacing any predicted placeholder with the label of the subject or object of the relevant triple at a post-processing step. In case the same property-type placeholder occurs more than once in a generated sentence, then we assign each one of them to the relevant triples randomly, while making sure that each placeholder will be mapped to a different triple.

Note that each summary is augmented with the respective start-of-summary `<start>` and end-of-summary `<end>` tokens.

4.3.2. Modelling the Input Triples

Similar to the Wikipedia summaries, we represent the occurrence of the main entity of the corresponding summary as either subject or object of a triple with the special `<item>` token. A shared, fixed dictionary was used for all subjects, predicates and objects. First, we included all the predicates and entities that occur at least 20 times. Triples with rare predicates were discarded. Every out-of-vocabulary entity was replaced by the special token of its instance type, which is retrieved from the Instance Types dataset. For example, the rare entity of `dbr:Mamma_Mia!` was replaced by the `dbo:Musical` token. In case an infrequent entity is not found in the Instance Types dataset, it is replaced with the special `<unk>` token. We appended to the source vocabulary only the instance type tokens that occur at least 20 times. We used the `<resource>` token for the rare entities with also infrequent instance types.

In order both to increase the homogeneity of the dataset in terms of the number of triples that are aligned with each

Table 4: Statistics regarding the initial and the training versions of our two corpora, D1 (i.e. based on DBpedia triples) and D2 (i.e. based on Wikidata triples).

Parameter	D1		D2	
	Initial Dataset	Training Dataset	Initial Dataset	Training Dataset
Total # of Articles	256850	239806	358908	354321
Total # of Entities	609k	8702	278k	10684
Total # of Predicates	450	256	378	395
Avg. # of Triples (incl. Encoded Dates) per Article	10.68	10.68	12.09	11.96
Max. # of Alloc. Triples (incl. Encoded Dates) per Article	175	22	255	21
Total # of Words In the Summaries	400k	14297	500k	16728
Total # of Annotated Entities In the Summaries	194k	15703	222k	16272

Wikipedia summary and to contain the space complexity of our task to a single Graphics Processing Unit (GPU), we limit the number of allocated triples per summary E to:

$$\lfloor E_{\min} + 0.25\sigma_E \rfloor \leq E \leq \lfloor \overline{E} + 1.5\sigma_E \rfloor \quad . \qquad (15)$$

If a biography is aligned with fewer triples then it is excluded from the respective dataset. If a summary is aligned with more triples, we first attempt to exclude potential duplicates (e.g. `Fiorenzo_Magni dbp:proyears 1945` and `Fiorenzo_Magni dbp:proyears 1944` would result in the same triple: `<item> dbp:proyears <year>` after the year-replacement process that is described in Sections 4.2). In case their number still exceeds the limit, we retain only the first ones until the threshold is reached.

Table 4 shows statistics on the initial and the training-ready versions of each corpus. An example of the structure of the datasets is displayed in Table 5. More details about the two different types of summaries ("Summary w/ URIs" and "Summary w/ Surface Form Tuples") with which we trained our models are provided in Section 5.

5. Experiments

We use our datasets, D1 and D2, to train and evaluate the performance of our neural network architectures. Both datasets are split into training, validation and test, with respective portions of 85%, 10%, and 5%. We implemented our neural network models using the Torch[12] package. Any cleaning or restructuring procedure that has been performed on the datasets has been conducted with Python scripts. The aligned corpora along with the code of our systems and the competing baseline are available at `github.com/pvougiou/Neural-Wikipedian`.

[12]Torch is a scientific computing package for Lua. It is based on the LuaJIT package.

Our proposed neural network architectures learn to generate a textual summary as a sequence of words and entities. In order to infer the verbalisation of the predicted entities in a generated summary, we experiment with two different approaches which are described in detail below.

5.1. Generating Words Along With URIs

In this setup, all entities that have been annotated in the text with DBpedia Spotlight are replaced with their URIs. The summaries vocabulary is comprised of words and the entities' URIs. The model thus learns to generate words along with entity URIs. In order to improve the generated text further, as a post-processing step we replace: (i) the `<item>` token with its corresponding surface form, and (ii) the tokens of DBpedia or Wikidata entities in the text, with their most frequently matched surface form, as these are recorded during data pre-processing (see Section 4.1).

5.2. Generating Words Along With Surface Form Tuples

Instead of replacing entity URIs with their most frequent surface forms, we propose a setup that enables our system to make a prediction about the best verbalisation of a predicted entity in the text. Each entity that has been identified in the text of the Wikipedia summaries using DBpedia Spotlight is stored as a tuple of the annotated surface form and its URI. Let $K = \{k_1, k_2, \ldots, k_D\}$ be the set of all D entities that are annotated in the text. We define the r-th *surface form tuple* of the d-th entity k_d as: $u_r^{k_d} = (k_d, g_r) : k_d \in K$, where g_r is the r-th surface form that is associated with the entity k_d. Similarly to Section 5.1, those tuples are stored as tokens in the target dictionary. This setup enables the models to verbalise each entity with more than one way by adapting the surface forms to the context of both the generated tokens and input triples. For example, the entity of `dbr:Actor` is associated with the surface form tuples of `(dbr:Actor, actor)` and `(dbr:Actor, actress)`, and, thus, it can be

Table 5: Example of the alignment of our datasets. One Wikipedia summary is coupled with a set of triples from either DBpedia or Wikidata. Any reference to the main discussed entity of the summary (i.e. `dbr:Papa_Roach` or `wikidata:Q254371` respectively) is replaced by the `<item>` token both in the text and the triples. Each other entity in the triples is stored along with its instance type. Each infrequent entity in the triples is replaced by the special token of its instance type, before it is provided to our neural network architectures as part of the input (e.g. `dbr:Infest_(album)` is replaced by `dbo:Album`). When a rare entity in the text is matched to an entity of the corresponding triples' set, then it is replaced by a unique token, which consists from the predicate of the relevant triple, a descriptor of the component (i.e. subject or object) of the triple that was matched, and the instance type of the entity (e.g. the music album "Infest (2000)" is replaced by the property-type placeholder `[dbo:artist__sub__dbo:Album]`). In case an infrequent entity in the text is not matched to any of the entities in the triples, it is replaced by the special token of its instance type (e.g. the entity of "Vacaville, California" does not appear in the Wikidata triples, and as a result is replaced in their corresponding text by the `dbo:City` token).

<item>	dbr:Papa_Roach and wikidata:Q254371
Original Wikipedia Summary	Papa Roach is an American rock band from Vacaville, California. Formed in 1993, their first major-label release was the triple-platinum album Infest (2000).
DBpedia Triples	<item> dbo:bandMember dbr:Jacoby_Shaddix [dbo:MusicalArtist] <item> dbo:bandMember dbr:Jerry_Horton [dbo:MusicalArtist] <item> dbo:genre dbr:Hard_rock [dbo:MusicGenre] ⋮ <item> dbo:hometown dbr:United_States [dbo:Country] <item> dbo:hometown dbr:Vacaville,_California [dbo:City] [dbo:Album] dbr:Infest_(album) dbo:artist <item> [dbo:Album] dbr:Metamorphosis_(Papa_Roach_album) dbo:artist <item>
Summary w/ URIs	<start> <item> is an dbr:United_States dbr:Rock_music band from [dbo:hometown__obj__dbo:City] . Formed in <year> , their first major-label release was the dbr:RIAA_certification album [dbo:artist__sub__dbo:Album] (<year>) . <end>
Summary w/ Surface Form Tuples	<start> <item> is an (dbr:United_States, American) (dbr:Rock_music, rock) band from [dbo:hometown__obj__dbo:City] . Formed in <year> , their first major-label release was the (dbr:RIAA_certification, triple-platinum) album [dbo:artist__sub__dbo:Album] (<year>) . <end>
Wikidata Triples	<item> wikidata:P136 wikidata:Q11399 [dbo:MusicGenre] <item> wikidata:P495 wikidata:Q30 [dbo:Country] <item> wikidata:P571Month 1 [<unk>] <item> wikidata:P571Year <year> [<unk>] <item> wikidata:P31 wikidata:Q215380 [<unk>] <item> wikidata:P264 wikidata:Q212699 [dbo:RecordLabel] ⋮ [dbo:Album] wikidata:Q902353 wikidata:P175 <item>
Summary w/ URIs	<start> <item> is an wikidata:Q30 wikidata:Q11399 band from dbo:City . Formed in <year> , their first <rare> release was the wikidata:Q2503026 album [wikidata:P175__sub__dbo:Album] (<year>) . <end>
Summary w/ Surface Form Tuples	<start> <item> is an (wikidata:Q30, American) (wikidata:Q11399, rock) band from dbo:City . Formed in <year> , their first <rare> release was the <unk> album [wikidata:P175__sub__dbo:Album] (<year>) . <end>

verbalised accordingly (as *actor* or *actress*) based on the gender of the main entity of interest.

5.3. Training Details

We train two different models. The first one is the triple encoder coupled with the GRU-based decoder to which we refer as Triples2GRU; the other is the same triple encoder coupled with the LSTM-based decoder (Triples2-LSTM). For each dataset, D1 and D2, we train each model on our task of generating a summary once as a combination of words with URIs (w/ URIs, 5.1) and once as a mixture of words and surface form tuples (w/ Surf. Form Tuples, 5.2).

For the recurrent component of our networks, we use 1 layer of (i) 650 LSTM cells and (ii) 750 GRUs, resulting in 3.385M and 3.380M recurrent connections, respectively. We found that increasing the number of layers does not improve the performance of our architectures, whereas the dimensionality of the hidden states plays a crucial role in achieving the best possible results. Table 6 summarises the hyper-parameters that have been used in training.

The feed-forward triples encoder consist of a sequence of fully-connected layers with the following [input, output] sizes: (i) one-hot input to vector representation of subject or predicate or object: $[|N| \cdot m, m]$, (ii) concatenated vector representation of each triple's subject-predicate-object to hidden representation of triple: $[3 \cdot m, m]$. At the topmost layer of the encoder, we have a fully-connected layer that maps the concatenated hidden representations of all the aligned to a summary triples to one single vector: $[E_{\max} \cdot$

Table 6: Training Hyperparameters of the Systems

Parameter	Triples2LSTM w/ URIs		Triples2GRU w/ URIs		Triples2LSTM w/ Surf. Form Tuples		Triples2GRU w/ Surf. Form Tuples	
	D1	D2	D1	D2	D1	D2	D1	D2
Batch Size	85	85	85	85	85	85	85	85
Max. Timestep T	66	69	66	69	66	69	66	68
Embedding Size m	650	650	750	750	650	650	750	750
Target Vocabulary Size $\|X\|$	30692	33644	30692	33644	30761	33715	30761	33715
Source Vocabulary Size $\|N\|$	9168	11088	9168	11088	9168	11088	9168	11088
Max. # of Alloc. Triples per Article $E_{\max}$	22	21	22	21	22	21	22	21
# Training Epochs[13]	12	16	12	16	12	15	13	22

$m, m]$, where $E_{\max}$ is the maximum number of triples per article. Sets of triples with fewer than $E_{\max}$ triples are *padded* with zero vectors when necessary.

We optimised our architectures using an alteration of stochastic gradient descent with adaptive learning rate. We found that a fixed learning rate was resulting in the explosion of the gradients that were propagated to the encoder side. We believe that this behaviour is explained by the fact that our models learn to project data of dissimilar nature (i.e. structured data from the triples and unstructured text from the summaries) in a shared continuous semantic space. In case their parameters are not initialised properly, our neural architectures propagate vectors of different orders of magnitude leading to the explosion of the gradients phenomenon. However, finding the appropriate values to initialise the models' parameters is not trivial [43]. In order to avoid this problem, we use Batch Normalisation before each non-linear activation function and after each fully-connected layer both on the encoder and the decoder side, and initialise all parameters with a random uniform distribution between -0.001 and 0.001 [43]. The networks are trained with mini-batch RMSProp with an initial learning rate value of 0.002. Each update is computed using a mini-batch of 85 dataset instances. An l_2 regularisation term over each network's parameters (weights) is also included in the cost function. After the 2nd epoch, the learning rate was decayed by 0.8 every epoch. During testing, our systems generate a summary for each unknown set of triples, using a beam B of size 5. We retain only the summary with the highest probability.

We trained all of our systems on a single Titan X (Pascal) GPU. The LSTM-based models complete an epoch of training: (i) in around 25 minutes when trained on the D2 dataset, and (ii) 17 minutes when trained on D1; the GRU-based architectures require (i) around 22 minutes when trained on D2, and (ii) 15 minutes when trained on D1.

5.4. Automatic Evaluation

We use perplexity[14], BLEU[15], and ROUGE[16] on the validation and test set. Perplexity indicates how well the model learns its training objective (c.f. Section 3.3); BLEU and ROUGE measure how close the generated text is to the actual Wikipedia summary. Essentially, BLEU and ROUGE are complimentary to each other. The first computes a modified version of n-gram precision[17], whereas the latter computes the n-gram recall, of the automatically generated sentences with respect to the actual Wikipedia summaries.

We adapt the code from the evaluation package that was released by Peter Anderson[18], which was originally implemented to evaluate the quality of textual descriptions from images. Perplexity, BLEU 1, BLEU 2, BLEU 3, BLEU 4, and ROUGE_L (an alteration of the original ROUGE that is automatically measured on the longest common sub-sequence) results are reported in Table 7.

To demonstrate the effectiveness of our system, we compare it to two baselines. First, we compute the expected lower bounds for BLEU scores by using a random Wikipedia summary generation baseline. We consider this a particularly strong baseline due to the fact that Wikipedia biographies tend to follow the same text structure. For each triples set on the validation and test set, the random system generates a response by randomly selecting a Wikipedia summary from our training set. Secondly,

[14]Perplexity measures the cross-entropy between the predicted sequence of words and the actual, empirical, sequence of words.

[15]BLEU (Bilingual Evaluation Understudy) [44] is precision-oriented metric for measuring the quality of generated text by comparing it to the actual, empirical text. BLEU-n calculates a similarity scores based on the co-occurrence of up to n-grams (i.e. 1-grams, ..., n-grams) in the generated and the actual text.

[16]ROUGE (Recall-Oriented Understudy for Gisting Evaluation) is a metric that computes the recall of n-grams in the generated text with respect to the n-grams of the actual text [45].

[17]The count of True Positives of an n-gram in the generated summary has an upper bound which is defined by the number of occurrences of this particular n-gram in the actual summary.

[18]`http://github.com/peteanderson80/coco-caption`

Table 7: Automatic evaluation with perplexity (i.e. lower is better), and the BLEU and ROUGE_L metrics (i.e. higher is better) on the validation and test set. The average performance of the two baselines along with its standard deviation is reported after sampling 10 times.

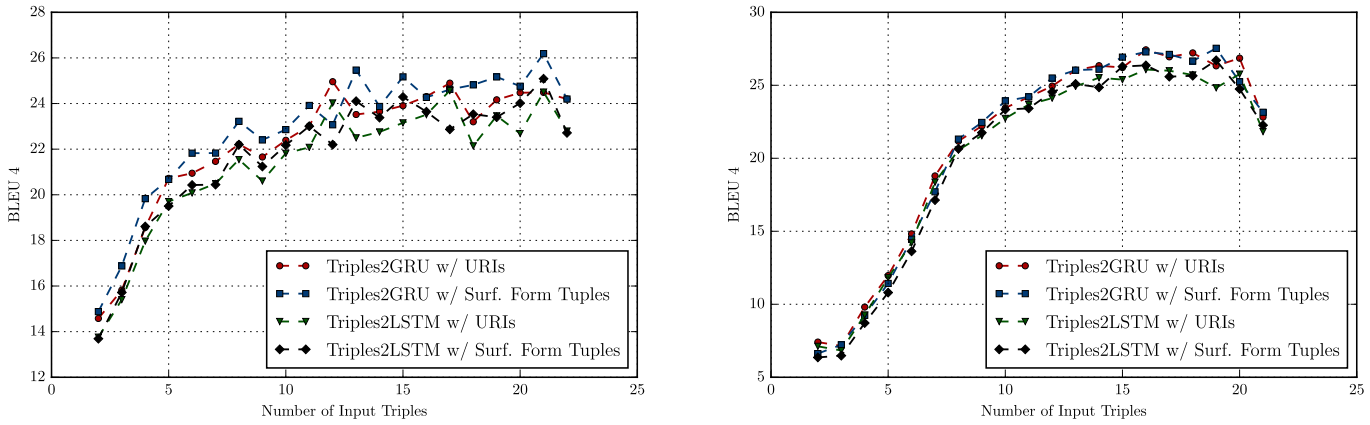
Model	**Perplexity**		**BLEU** 1		**BLEU** 2		**BLEU** 3		**BLEU** 4		**ROUGE$_\text{L}$**	
	Valid.	Test	Valid.	Test	Valid.	Test	Valid.	Test	Valid.	Test	Valid.	Test
Random Baseline on D1	–	–	29.523 (±.04)	29.650 (±.06)	17.270 (±.04)	17.390 (±.05)	11.415 (±.04)	11.528 (±.04)	7.561 (±.03)	7.658 (±.03)	27.578 (±.05)	27.715 (±.06)
KN on D1	–	–	22.587 (±.00)	22.685 (±.01)	16.601 (±.01)	16.722 (±.01)	12.626 (±.01)	12.750 (±.01)	9.412 (±.01)	9.518 (±.01)	38.202 (±.01)	38.418 (±.01)
Triples2LSTM on D1 w/ URIs	19.447	19.769	40.134	39.902	30.610	30.430	25.188	25.025	21.285	21.121	45.981	45.937
Triples2GRU on D1 w/ URIs	20.530	20.929	41.003	40.954	31.557	31.479	26.088	25.984	22.116	22.001	**47.092**	47.100
Triples2LSTM on D1 w/ Surf. Form Tuples	19.171	19.086	40.679	40.763	30.809	30.904	25.234	25.344	21.287	21.393	44.973	45.143
Triples2GRU on D1 w/ Surf. Form Tuples	20.164	20.007	**41.350**	**41.457**	**31.877**	**31.991**	**26.387**	**26.510**	**22.419**	**22.531**	47.027	**47.235**
Random Baseline on D2	–	–	29.636 (±.03)	29.650 (±.03)	17.587 (±.03)	17.581 (±.03)	11.818 (±.02)	11.800 (±.03)	7.910 (±.02)	7.892 (±.03)	28.083 (±.04)	28.109 (±.04)
KN on D2	–	–	22.716 (±.00)	22.713 (±.00)	16.680 (±.00)	16.675 (±.00)	12.692 (±.00)	12.685 (±.00)	9.448 (±.01)	9.432 (±.01)	37.937 (±.00)	37.957 (±.01)
Triples2LSTM on D2 w/ URIs	20.995	21.045	40.967	41.134	31.190	31.312	25.729	25.812	21.780	21.845	46.522	46.675
Triples2GRU on D2 w/ URIs	21.770	21.823	41.470	**41.618**	31.920	32.072	26.475	26.604	22.497	22.619	47.577	47.761
Triples2LSTM on D2 w/ Surf. Form Tuples	20.779	20.403	40.660	40.604	31.158	31.144	25.776	25.783	21.874	21.898	47.031	47.140
Triples2GRU on D2 w/ Surf. From Tuples	21.493	21.200	**41.527**	41.566	**32.072**	**32.097**	**26.645**	**26.673**	**22.679**	**22.708**	**47.979**	**48.100**

we use the KenLM toolkit [46] in order to build a 5-gram Kneser-Ney (KN) language model. During testing, similarly to the case of our neural network approaches, we use beam-search with a beam of size 10, to generate the 10 most probable summaries for each triple set in the validation and test set. We equip both baselines with surface form tuples, and the `<item>` and property-type placeholders. After a summary is selected, its `<item>` placeholders along with any potential property-type placeholders are replaced according to the original triples. In the case where a property-type placeholder is not matched to the content of the triples, it is replaced by its corresponding instance type token (see more in Section 4.3.1). The results are illustrated in Table 7.

The GRU-based systems outperform the LSTM-based ones according to all the automatic evaluation metrics. The Triples2GRU model equipped with surface form tuples achieves a total improvement of 13 BLEU (using BLEU 4) and 9 ROUGE points over the KN baseline on both datasets. Furthermore, for the same architecture (GRU- or LSTM-based), we noted a correlation between perplexity, which is our training objective, and BLEU. We found that an average reduction of 0.5 in perplexity gives us an improvement of around 0.2 BLEU-4 points in the case of the LSTM-based systems, and 0.4 BLEU-4 points in the case of the GRU-based ones.

In addition to the above experiments, we group Wikipedia summaries that are allocated to same number of input triples and compute a BLEU score per group. Figure 3 displays the performance of our models with the BLEU 4 metric across different numbers of input triples. The low performance of the models when they are initialised with a low number of triples is explained by the fact that the systems are lacking information required to form a two-sentence summary. In general, all the systems achieve a stable performance when they are inputted with 10 or more triples.

BLEU and ROUGE are well-established automatic text similarity metrics that are extensively used in machine translation tasks where there is a tight alignment between the source and the generated language [5, 6]. Our generative task is more challenging since it consists of learning



(a) D1 (b) D2

Figure 3: Performance of our models with the BLEU 4 metric across the different number of input triples on D1 (a) and D2 (b).

to generate text from a corpus of triples loosely associated with text. While this explains why our scores are lower than those usually reported for machine translation tasks, it also suggests that these metrics have limitations when applied to NLG tasks in which the "correct" output is neither purely deterministic (i.e. there are multiple ways to correctly summarise a set of knowledge base triples in text) or necessarily based on the empirical data (i.e. the actual Wikipedia summary that is allocated to a set of triples might discuss irrelevant facts than those that exist in the allocated triple set). Consequently, we conduct a pilot evaluation in order to determine with greater certainty which one of our explored architectures serves the needs of our task the best.

5.5. Human Evaluation

We evaluated the performance of our approach in a pilot case study with seven researchers from the University of Southampton and Jean Monnet University (UJM-Saint-Étienne). All the participants are experts in the field of Semantic Web and have full professional proficiency in English. For each corpus, we compiled a list of 15 randomly selected sets of triples along with the textual summaries that have been generated from each one of our proposed models (i.e. (i) GRU with URIs and surface form tuples, and (ii) LSTM with URIs and surface form tuples). The sets of triples are sampled from the test sets. We conducted two separate studies, one for each corpus.

Our experiments showed that in our dataset, triples sets with fewer triples usually lack enough information for our systems to generate a summary (c.f. Section 5.4). Hence, we included only input sets of triples that consist of at least 6 triples. The specific model which generated the summary (i.e. LSTM or GRU with URIs or surface form tuples) was not disclosed. Beyond evaluating the fluency of the generated summaries, our goal is to explore to what extent the text addresses the information in the triples, without contradicting the input facts. Consequently, the evaluators were asked to rate each summary against four different criteria: (i) fluency[19], (ii) number of contradicting facts (i.e. information that exists in the sentence but it conflicts one or more of triples from the input set), (iii) number of summarised triples[20] (i.e. triples whose information is mentioned either implicitly or explicitly in the text), and (iv) number of triples to which potential additional information in the text can be interpreted[21].

The scores for criteria (ii)-(iv) are dependent on the number of triples of each input set. Consequently, we normalise them based on the total number of triples of their respective input set. Fluency was marked on a scale from 1 to 6, with 1 indicating an incomprehensible summary, 2 a barely understandable summary with significant grammatical errors, 3 an understandable summary with grammatical flaws, 4 a comprehensible summary with minor grammatical errors, 5 a comprehensible and grammatically correct summary that reads a bit artificial, and 6 a coherent and grammatically correct summary [47].

The results of the human evaluators are in agreement with the results of the automatic evaluation with both the BLEU and ROUGE metrics (i.e. Section 5.4). For the same training setup (i.e. w/ URIs or surface form tuples) and with only exception the fluency performance of the Triples2LSTM compared to the Triples2GRU on D1 w/ URIs, the GRU-based architectures outperform the LSTM-based ones in all criteria. Furthermore, they score consistently better in terms of the inclusion of additional or

[19] Adapted from [47] where they use an identical metric to evaluate the comprehensibility and readability of a generated question in natural language.

[20] Similar to *coverage* in [31] which measures the number of included sub-graphs in the text.

[21] For example, in case there is information in the text about a person's birthplace, without any birthplace-related triple in the input; since the place of birth can be described in a single triple (with the predicates of `dbo:birthPlace` or `dbo:hometown`), we increment the number of additional triples by one.

Table 8: Examples of textual summaries that are generated by our proposed systems given unknown sets of triples as input. For each model, we report its immediate output along with its corresponding (Final) version after the replacement of the generated placeholder tokens. Each other than the main discussed entity (<item>) in the triples is recorded and displayed along with its instance type. Rare entities in the input triples are replaced with their respective instance type tokens. The greyed-out tokens of either entities or instance type tokens refer to tokens that are not used during the training of the neural network models.

<item>	dbr:Barbara_Flynn
Triples	<item> dbo:birthPlace dbr:England [owl#Thing] <item> dbo:birthPlace dbr:St_Leonards-on-Sea [dbo:Settlement] <item> dbo:birthPlace dbr:Sussex [owl#Thing] <item> dbo:occupation dbr:Actress [<unk>] <item> dbo:birthDateMonth 8 [<unk>] <item> dbo:birthDateYear <year> [<unk>] [dbo:TelevisionShow] dbr:Open_All_Hours dbo:starring <item> [dbo:TelevisionShow] dbr:A_Very_Peculiar_Practice dbo:starring <item> [dbo:TelevisionShow] dbr:The_Beiderbecke_Trilogy dbo:starring <item> . . . [dbo:TelevisionShow] dbr:Cracker_(UK_TV_series) dbo:starring <item>
Triples2GRU w/ URIs	<start> <item> (born 0 August <year>) is an dbr:English_people dbr:Actor and dbr:Actor. She is best known for her roles in the dbr:Television_program [dbo:starring__sub__dbo:TelevisionShow] , [dbo:starring__sub__dbo:TelevisionShow] and [dbo:starring__sub__dbo:TelevisionShow] . <end>
Triples2GRU w/ URIs (Final)	<start> Barbara Flynn (born 0 August <year>) is an English actor and actor . She is best known for her roles in the television series A Very Peculiar Practice , Beiderbecke Trilogy and Open All Hours . <end>
Triples2GRU w/ Surf. Form Tuples	<start> <item> (born 0 August <year>) is an (dbr:English_people, English) (dbr:Actor, actress) . She is best known for her roles in [dbo:starring__sub__dbo:TelevisionShow] and [dbo:starring__sub__dbo:TelevisionShow] . <end>
Triples2GRU w/ Surf. Form Tuples (Final)	<start> Barbara Flynn (born 0 August <year>) is an English actress . She is best known for her roles in Beiderbecke Trilogy and Open All Hours . <end>
Triples2LSTM w/ URIs	<start> <item> (born 0 August <year>) is an dbr:English_people dbr:Actor . She is best known for her role as dbo:SoapCharacter in the BBC soap opera EastEnders . <end>
Triples2LSTM w/ URIs (Final)	<start> Barbara Flynn (born 0 August <year>) is an English actor . She is best known for her role as dbo:SoapCharacter in the BBC soap opera EastEnders . <end>
Triples2LSTM w/ Surf. Form Tuples	<start> <item> (born 0 August <year>) is an (dbr:English_people, English) (dbr:Actor, actress) . She is best known for her role as dbo:SoapCharacter in the (dbr:BBC, BBC) soap opera (dbr:EastEnders, EastEnders) . <end>
Triples2LSTM w/ Surf. Form Tuples (Final)	<start> Barbara Flynn (born 0 August <year>) is an English actress . She is best known for her role as dbo:SoapCharacter in the BBC soap opera EastEnders . <end>

<item>	dbr:Lee_Jeong-beom
Triples	<item> dbo:birthPlace dbr:South_Korea [dbo:Country] <item> dbo:education dbr:Korea_National_University_of_Arts [dbo:University] <item> dbo:occupation dbr:Film_director [owl#Thing] <item> dbo:occupation dbr:Screenwriter [owl#Thing] <item> dbo:birthDateMonth 9 [<unk>] <item> dbo:birthDateYear <year> [<unk>] . . . [dbo:Film] dbr:Cruel_Winter_Blues dbo:director <item> [dbo:Film] dbr:Cruel_Winter_Blues dbo:writer <item>
Triples2GRU w/ URIs	<start> <item> (born September 0 , <year>) is a dbr:South_Korea dbr:Film_director and dbr:Screenwriter . He is best known for his films [dbo:director__sub__dbo:Film] (<year>) and [dbo:director__sub__dbo:Film] (<year>) . <end>
Triples2GRU w/ URIs (Final)	<start> Lee Jeong-beom (born September 0 , <year>) is a South Korean film director and screenwriter . He is best known for his films Cruel Winter Blues (<year>) and dbo:Film (<year>) . <end>
Triples2GRU w/ Surf. Form Tuples	<start> <item> ([dbr:Hangul, Hangul] : <rare> ; born September 0 , <year>) is a (dbr:South_Korea, South Korean) (dbr:Film_director, film director) and (dbr:Screenwriter, screenwriter) . He is best known for directing the <year> (dbr:Film_director, film) [dbo:director__sub__dbo:Film] . <end>
Triples2GRU w/ Surf. Form Tuples (Final)	<start> Lee Jeong-beom (Hangul : <rare> ; born September 0 , <year>) is a South Korean film director and screenwriter . He is best known for directing the <year> film Cruel Winter Blues . <end>
Triples2LSTM w/ URIs	<start> <item> (born September 0 , <year>) is a dbr:South_Korea dbr:Film_director and dbr:Screenwriter . He has directed more than 0 films since year . <end>
Triples2LSTM w/ URIs (Final)	<start> Lee Jeong-beom (born September 0 , <year>) is a South Korean film director and screenwriter . He has directed more than 0 films since <year> . <end>
Triples2LSTM w/ Surf. Form Tuples	<start> <item> (born September 0 , <year>) is a (dbr:South_Korea, South Korean) (dbr:Film_director, film director) and (dbr:Screenwriter, screenwriter) . He has directed 0 films since <year> . <end>
Triples2LSTM w/ Surf. Form Tuples (Final)	<start> Lee Jeong-beom (born September 0 , <year>) is a South Korean film director and screenwriter . He has directed 0 films since <year> . <end>

contradicting information. Since they are more reluctant to introduce out-of-context information in the text, their generated textual content is better aligned with the input triples.

Table 9: The average rating of our systems against the human evaluation criteria. For fluency and summarised triples the higher the score the better; for contradicting triples and additional information, the lower the score the better. The results are reported in the "mean (± standard deviation)" format.

Model	Fluency	Summarised Triples	Contradicting Triples	Additional Information
Triples2LSTM on D1 w/ URIs	5.124 (±0.963)	0.4 (±0.169)	0.045 (±0.069)	0.143 (±0.151)
Triples2LSTM on D1 w/ Surf. Form Tuples	5.287 (±0.791)	0.457 (±0.236)	0.05 (±0.068)	0.145 (±0.169)
Triples2GRU on D1 w/ URIs	4.9 (±1.006)	0.423 (±0.221)	0.023 (±0.06)	**0.112** (±0.141)
Triples2GRU on D1 w/ Surf. Form Tuples	**5.511** (±0.640)	**0.497** (±0.247)	**0.017** (±0.056)	0.134 (±0.177)
Triples2LSTM on D2 w/ URIs	5.036 (±1.017)	0.582 (±0.185)	0.018 (±0.037)	0.103 (±0.109)
Triples2LSTM on D2 w/ Surf. Form Tuples	5.470 (±0.687)	0.582 (±0.185)	0.018 (±0.037)	0.103 (±0.109)
Triples2GRU on D2 w/ URIs	5.349 (±0.833)	0.596 (±0.200)	**0.006** (±0.023)	0.085 (±0.107)
Triples2GRU on D2 w/ Surf. Form Tuples	**5.663** (±0.668)	**0.597** (±0.194)	0.009 (±0.028)	**0.073** (±0.101)

Table 10: Nearest neighbours of the vector representations of some of the most frequently occurring entities as these are learned by the encoder.

DBpedia Entity	Nearest Neighbours
dbr:France	dbr:Paris, dbr:France, dbr:Marseille, dbr:Lyon, dbr:Kingdom_of_France, and dbr:Olympique_de_Marseille
dbr:Japan	dbr:Empire_of_Japan, dbr:Chiba_Prefecture, dbr:Yokohama, dbr:Osaka, and dbr:Kyoto
dbr:Singer	dbr:Singing, dbr:Vocalist, dbr:Vocals, dbr:Playback_singer, and dbr:Americana_(music)
dbr:Heavy_metal_music	dbr:Glam_metal, dbr:Doom_metal, dbr:Hard_rock, dbr:Nu_metal, and dbr:Alternative_metal
dbr:FC_Barcelona	dbr:RCD_Mallorca, dbr:Athletic_Bilbao, dbr:Spain_national_under-18_football_team, dbr:Valencia_CF, and dbr:Battle_of_the_Atlantic

Wikidata Entity	Nearest Neighbours
wikidata:Q64 (Berlin)	wikidata:Q1022 (Stuttgart), wikidata:Q365 (Taiwan), wikidata:Q152087 (Humboldt University of Berlin), wikidata:Q1731 (Dresden), and wikidata:Q43287 (German Empire)
wikidata:Q148 (China)	wikidata:Q17427 (Communist Party of China), wikidata:Q865 (Taiwan), wikidata:Q7850 (Chinese language), wikidata:Q8686 (Shanghai), and wikidata:Q1348 (Kolkata)
wikidata:Q20 (Norway)	wikidata:Q35 (Denmark), wikidata:Q486156 (University of Oslo), wikidata:Q9043 (Norwegian language), wikidata:Q11739 (Lahore), and wikidata:Q585 (Oslo)
wikidata:Q15981151 (jazz musician)	wikidata:Q12800682 (saxophonist), wikidata:Q248970 (Berklee College of Music), wikidata:Q806349 (bandleader), wikidata:Q12804204 (percussionist), and wikidata:Q8341 (jazz)
wikidata:Q158852 (conductor)	wikidata:Q1415090 (film score composer), wikidata:Q9734 (symphony), wikidata:Q3455803 (director), wikidata:Q1198887 (music director), and wikidata:Q2994538 (Conservatoire national supérieur de musique et de danse)

Interestingly, all the models that were trained on D2 scored consistently better in terms of the number of summarised triples. We believe that this is due to the fact that the information from the Wikidata triples, with which D2 was formed, is better aligned with the content of the first two sentences of the Wikipedia biographies than the Mapping-based DBpedia datasets that we used for D1. "Noisy" triples, such as: `dbr:Sequoyah dbo:occupation dbr:Sequoyah__1` and `dbr:Acie_Law dbo:termPeriod Acie_Law__[1-10]`, are very common in the DBpedia triples allocated to the Wikipedia biographies. Since, their information is not verbalised in the text, our systems learn to disregard them, explaining the lower scores that these model achieve with respect to the number of summarised triples.

In general the evaluators scored all of our systems with high fluency ratings, thus, emphasising the ability of our approach to generate grammatically and syntactically correct text. We should note, however, that verbalising the occurrence of entities in the text with the mechanism of surface form tuples (systems w/ Surf. Form Tuples) results consistently in higher fluency scores regardless of the architecture of the decoder (LSTM- or GRU-based).

5.6. Discussion

Two examples of textual summaries that are generated by our models are shown in Table 8. We selected two representative sets of triples from the test set. The examples show that our approach can generate sentences that couple information from several triples from an input set. In the first example, all the models are able to capture the main entity's gender from the input triple set. However, only in the case of the models equipped with surface form tuples, we are able to verbalise the entity of `dbr:Actor` correctly as "actress" in the text. This is due to the fact that in the biographies dataset, the most frequent surface form, with which the entity of `dbr:Actor` has been associated, is "actor". Consequently, actor is used as the replacement of all the occurrences of the `dbr:Actor` entity in the summaries that are generated by our w/ URIs systems. This limitation of the w/ URIs systems in terms of their ability to learn different verbalisation for the various entities in the text explains their lower fluency scores compared to the systems w/ Surf. Form Tuples in our case study (see Section 5.5).

The learned embeddings in the decoder capture information that is both coupled with the embeddings in the encoder (e.g. the embedding of the pronouns "She" and "her" are coupled implicitly with the existence of the triple: `<item> dbo:occupation dbr:Actress`), and their own probability of occurring in the context of the sequentially generated text (e.g. word with its first letter capitalised, when it is following a full stop). Consequently, items that have similar meanings find themselves close together in the continuous semantic space. Table 10 shows the nearest neighbours of some of the most frequently occurring entities in our datasets which have been learned by our models. This shows that our models can successfully infer semantic relationships among entities.

The main drawback of training our models on a dataset of loosely associated triples with text is that the information that exists in the triples does not necessarily appear in the corresponding text, and vice versa. As a result, the models are not penalised when they generate textual information that does not exist in the input. For instance, in the first example the LSTM-based models assume that Barbara Flynn has appeared in the "in the BBC soap opera EastEnders". While a textual mention to the *EastEnders* series is relatively common in the Wikipedia summaries of both D1 and D2, those summaries are rarely aligned with a triple set that would include a reference to this series. In particular, out of the 197 total occurrences of *EastEnders* in the Wikipedia summaries of D1, there are only 5 instances where a reference to the series exists both in the biography and the corresponding triple set. In such cases, our systems is not able to sufficiently correlate the output with the respective input, and as a result, they learn a general pattern in terms of predicting a mention of *EastEnders* in the text (i.e. usually appears in the case of actors who have starred in television series).

A similar symptom is the occasional generation of special instance-type tokens, such as `dbo:SoapCharacter`. This is also based on the loose association of the triples with the summaries, and on our design choice not to use a single special token for infrequent entities that have been identified in the text but do not appear in the triples set (see Section 4.3.1). These are essentially learned when the models meet many training examples in which such a token is part of the text associated with a similar pattern of input triples. While their existence is not ideal, we believe that their inclusion is the best possible alternative since the generated summaries: (i) are not overwhelmed by a single special `<rare>` token, and (ii) are more readable because a human can understand what type of information the model wants to communicate in the text. We would also argue that summaries with such tokens could be used as a starting point for improving the coverage of the triples with respect to the automatically generated text, by hinting the missing knowledge base triples based on the type of the not-verbalised entities in the text.

It would have been very challenging to learn high quality vector representations for numerical values due to their infrequent occurrence in the dataset. Our choice of using one special token for numbers and one for years in the textual summaries does not allow us to effectively examine our systems' ability at generating plural forms. One way of addressing this limitation would be the introduction of additional placeholder tokens that replace the occurrences of numbers in the text with the property of the triple that contains them (in case such a triple exists in the input set). This could still prove problematic in the case of multiple triples that share in the input that share the same property. An alternate method would be the adapt the architecture of a pointer-generator network, similar to [48], that would theoretically be able to directly copy from the source a numerical value in the generated text.

The performance of our model is dependent on the number of input triples (see Section 5.4). We observed a stable performance when the size of the input set is ≥ 10. However, we also noted a slight drop in performance when our systems are provided with sets of triples that consist of more than 20 triples (c.f. Figure 3). This is explained by the fact that the decoder progressively "forgets" parts of the information from the encoder [49]. This is slightly more noticeable in our task, since we are generating long snippets of text (i.e. two-sentence summaries) based on the compressed information from a single vector [49]. Based on this finding, we identify the following challenges with respect to the employment of our proposed systems for generation of multi-sentence summaries (i.e. very long sequences) given very large input sets of triples: (i) our encoder would be forced to compress all the relevant information that should be summarised in the text in only one single vector; (ii) the decoder would have to retain the information from the input at very distant timesteps.

We believe that the first and partially the second challenge can be addressed with the introduction of an at-

tention mechanism over the vector representations of each triple in the input set. The idea of an attention mechanism was first introduced in Neural Machine Translation [49, 50]. The inclusion of attention substantially increased the investigated models' performance on longer sequences. In our case, the attention mechanism would make our systems more robust when dealing with over-sized input triples sets, since it would learn which parts of the input are the most important throughout the text generation procedure.

The second challenge gives ground to repetition, which is an additional problem that is associated with the generation of much longer snippets of text using attentive adaptations of the general encoder-decoder framework [51, 52]. While such behaviour was not observed in our experiments, it might prove to be one of the challenges in a multi-sentence generation scenario. Essentially, across distant decoding timestep, the decoder forgets which parts of the input have already been expressed in the generated output, and re-addresses them. This problem had been recently addressed with the implementation of a coverage architecture on top of the attention mechanism [51, 52]. *Coverage* is a vector that records the part of the input that the encoder had paid attention to during previous timesteps in order to avoid attending them, and thus, mentioning them again in the text, at future timesteps. The implementation of an attention mechanism over the input set of triples will allow us to further explore whether an architecture that monitors the coverage of the generated text is required in a triples-to-multi-sentence-summaries scenario.

6. Conclusion

To the best of our knowledge this work constitutes the first attempt to use neural networks for Natural Language Generation on top of Semantic Web triples. We propose an end-to-end trainable system that can generate a textual summary from triples. The generated summary verbalises various aspects of the information encoded within the input triple set.

Our approach does not require any manually defined templates and can be applied to a great variety of domains. We also propose a method of building a loosely aligned dataset of DBpedia and Wikidata triples with Wikipedia summaries in order to satisfy the training requirements of our system. Using these datasets, we have demonstrated that our technique is capable of scaling to domains with vocabularies of over 400k words. We address the problem of learning high quality vector representations for rare entities by adapting a multi-placeholder approach. Our systems learn to emit placeholder tokens that are replaced by the surface forms of the corresponding entities in the triples at a post-processing step.

We use a series of well-established automatic text similarity metrics in order to automatically evaluate our approach's ability of predicting the Wikipedia summary that corresponds to a set of unknown triples showing substantial improvement over our baselines. Furthermore, our statistical approach for inferring the verbalisation of the entities in the text with the surface form tuples mechanism (systems w/ Surf. Form Tuples), further enhances the, reported by our human evaluators, fluency of the generated summaries compared to a purely deterministic replacement of the generated entities' URIs. However, in our pilot study we did not explicitly investigate the performance of our approach under specific realisation scenarios, such as conjugated verbs or plural forms. This could be investigated further under a controlled study in the future. We also believe that our pilot study could serve as a starting point to a future study that would examine the actual usability of the automatically generated summaries in Wikipedia whose coverage, especially in the less popular languages or topics, is still limited [34].

Narrowing down our text generation domain to biographies provided us with the opportunity to better understand the strengths and limitations of our approach. Our imminent goal is to expand our approach on the whole Wikipedia corpus. Furthermore, while our triples are not restricted to a single entity, our model adopts the notion of the "main entity of interest", since our goal is to generate a summary about that particular entity. A natural extension of this work is to explore what alterations of our current approach, in terms both of system architecture and dataset building, are required to produce a narrative about multiple entities.

Acknowledgements

This research is partially supported by the Answering Questions using Web Data (WDAqua) project, a Marie Skłodowska-Curie Innovative Training Network under grant agreement No 642795, part of the Horizon 2020 programme.

References

[1] Janzen S, Maass W. Ontology-based natural language processing for in-store shopping situations. In: 2009 IEEE International Conference on Semantic Computing. 2009, p. 361–6. doi:10.1109/ICSC.2009.44.

[2] Bouayad-Agha N, Casamayor G, Mille S, Wanner L. Perspective-oriented generation of football match summaries: Old tasks, new challenges. ACM Trans Speech Lang Process 2012;9(2):3:1–3:31. URL: http://doi.acm.org/10.1145/2287710.2287711. doi:10.1145/2287710.2287711.

[3] Dannélls D, Damova M, Enache R, Chechev M. Multilingual online generation from semantic web ontologies. In: Proceedings of the 21st International Conference on World Wide Web. WWW '12 Companion; New York, NY, USA: ACM. ISBN 978-1-4503-1230-1; 2012, p. 239–42. URL: http://doi.acm.org/10.1145/2187980.2188018. doi:10.1145/2187980.2188018.

[4] Bouayad-Agha N, Casamayor G, Wanner L. Natural language generation in the context of the semantic web. Semantic Web 2014;5(6):493–513. URL: http://dx.doi.org/10.3233/SW-130125. doi:10.3233/SW-130125.

[5] Cho K, van Merrienboer B, Gulcehre C, Bahdanau D, Bougares F, Schwenk H, et al. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In:

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP). Doha, Qatar: Association for Computational Linguistics; 2014, p. 1724–34. URL: http://www.aclweb.org/anthology/D14-1179.

[6] Sutskever I, Vinyals O, Le QV. Sequence to sequence learning with neural networks. In: Ghahramani Z, Welling M, Cortes C, Lawrence ND, Weinberger KQ, editors. Advances in Neural Information Processing Systems 27. Curran Associates, Inc.; 2014, p. 3104–12.

[7] Serban IV, García-Durán A, Gulcehre C, Ahn S, Chandar S, Courville A, et al. Generating factoid questions with recurrent neural networks: The 30m factoid question-answer corpus. In: Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). Berlin, Germany: Association for Computational Linguistics; 2016, p. 588–98. URL: http://www.aclweb.org/anthology/P16-1056.

[8] Luong T, Sutskever I, Le Q, Vinyals O, Zaremba W. Addressing the rare word problem in neural machine translation. In: Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers). Beijing, China: Association for Computational Linguistics; 2015, p. 11–9. URL: http://www.aclweb.org/anthology/P15-1002.

[9] Liang P, Jordan MI, Klein D. Learning semantic correspondences with less supervision. In: Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1 - Volume 1. ACL '09; Stroudsburg, PA, USA: Association for Computational Linguistics. ISBN 978-1-932432-45-9; 2009, p. 91–9. URL: http://dl.acm.org/citation.cfm?id=1687878.1687893.

[10] Chen DL, Mooney RJ. Learning to sportscast: A test of grounded language acquisition. In: Proceedings of the 25th International Conference on Machine Learning. ICML '08; New York, NY, USA: ACM. ISBN 978-1-60558-205-4; 2008, p. 128–35. URL: http://doi.acm.org/10.1145/1390156.1390173. doi:10.1145/1390156.1390173.

[11] Mrabet Y, Vougiouklis P, Kilicoglu H, Gardent C, Demner-Fushman D, Hare J, et al. Aligning texts and knowledge bases with semantic sentence simplification. In: Proceedings of the 2nd International Workshop on Natural Language Generation and the Semantic Web (WebNLG 2016). Association for Computational Linguistics; 2016, p. 29–36. URL: http://www.aclweb.org/anthology/W16-3506.

[12] Gardent C, Shimorina A, Narayan S, Perez-Beltrachini L. Creating training corpora for NLG micro-planners. In: Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). Vancouver, Canada: Association for Computational Linguistics; 2017, p. 179–88. URL: http://aclweb.org/anthology/P17-1017.

[13] Lebret R, Grangier D, Auli M. Generating text from structured data with application to the biography domain. CoRR 2016;abs/1603.07771. URL: http://arxiv.org/abs/1603.07771.

[14] Reiter E, Dale R. Building Natural Language Generation Systems. New York, NY, USA: Cambridge University Press; 2000. ISBN 0-521-62036-8.

[15] Reiter E, Sripada S, Hunter J, Yu J, Davy I. Choosing words in computer-generated weather forecasts. Artif Intell 2005;167(1-2):137–69. URL: http://dx.doi.org/10.1016/j.artint.2005.06.006. doi:10.1016/j.artint.2005.06.006.

[16] Green N. Generation of biomedical arguments for lay readers. In: Proceedings of the Fourth International Natural Language Generation Conference. INLG '06; Stroudsburg, PA, USA: Association for Computational Linguistics. ISBN 1-932432-72-8; 2006, p. 114–21. URL: http://dl.acm.org/citation.cfm?id=1706269.1706292.

[17] Turner R, Sripada Y, Reiter E. Generating approximate geographic descriptions. In: Proceedings of the 12th European Workshop on Natural Language Generation. ENLG '09; Stroudsburg, PA, USA: Association for Computational Linguistics; 2009, p. 42–9. URL: http://dl.acm.org/citation.cfm?id=1610195.1610202.

[18] Angeli G, Liang P, Klein D. A simple domain-independent probabilistic approach to generation. In: Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing. EMNLP '10; Stroudsburg, PA, USA: Association for Computational Linguistics; 2010, p. 502–12. URL: http://dl.acm.org/citation.cfm?id=1870658.1870707.

[19] Kim J, Mooney RJ. Generative alignment and semantic parsing for learning from ambiguous supervision. In: Proceedings of the 23rd International Conference on Computational Linguistics: Posters. COLING '10; Stroudsburg, PA, USA: Association for Computational Linguistics; 2010, p. 543–51. URL: http://dl.acm.org/citation.cfm?id=1944566.1944628.

[20] Vinyals O, Toshev A, Bengio S, Erhan D. Show and tell: A neural image caption generator. In: 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2015, p. 3156–64. doi:10.1109/CVPR.2015.7298935.

[21] Wen TH, Gasic M, Mrkšić N, Su PH, Vandyke D, Young S. Semantically conditioned LSTM-based natural language generation for spoken dialogue systems. In: Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing. Lisbon, Portugal: Association for Computational Linguistics; 2015, p. 1711–21. URL: http://aclweb.org/anthology/D15-1199.

[22] Wen TH, Gašić M, Mrkšić N, Rojas-Barahona LM, Su PH, Vandyke D, et al. Multi-domain neural network language generation for spoken dialogue systems. In: Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. San Diego, California: Association for Computational Linguistics; 2016, p. 120–9. URL: http://www.aclweb.org/anthology/N16-1015.

[23] Vougiouklis P, Hare J, Simperl E. A neural network approach for knowledge-driven response generation. In: Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers. Osaka, Japan: The COLING 2016 Organizing Committee; 2016, p. 3370–80. URL: http://aclweb.org/anthology/C16-1318.

[24] Vinyals O, Kaiser Lu, Koo T, Petrov S, Sutskever I, Hinton G. Grammar as a foreign language. In: Cortes C, Lawrence ND, Lee DD, Sugiyama M, Garnett R, editors. Advances in Neural Information Processing Systems 28. Curran Associates, Inc.; 2015, p. 2773–81. URL: http://papers.nips.cc/paper/5635-grammar-as-a-foreign-language.pdf.

[25] Karpathy A, Fei-Fei L. Deep visual-semantic alignments for generating image descriptions. IEEE Transactions on Pattern Analysis and Machine Intelligence 2017;39(4):664–76. doi:10.1109/TPAMI.2016.2598339.

[26] Vinyals O, Le QV. A neural conversational model. CoRR 2015;abs/1506.05869.

[27] Hochreiter S, Schmidhuber J. Long short-term memory. Neural Comput 1997;9(8):1735–80. doi:10.1162/neco.1997.9.8.1735.

[28] Shang L, Lu Z, Li H. Neural responding machine for short-text conversation. In: Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers). Beijing, China: Association for Computational Linguistics; 2015, p. 1577–86.

[29] Arguello M, Des J, Fernandez-Prieto MJ, Perez R, Lekkas S. An ontology-based approach to natural language generation from coded data in electronic health records. In: 2011 UKSim 5th European Symposium on Computer Modeling and Simulation. 2011, p. 366–71. doi:10.1109/EMS.2011.47.

[30] Duma D, Klein E. Generating natural language from linked data: Unsupervised template extraction. In: Proceedings of the 10th International Conference on Computational Semantics (IWCS 2013) – Long Papers. Potsdam, Germany: Association for Computational Linguistics; 2013, p. 83–94. URL: http://www.aclweb.org/anthology/W13-0108.

[31] Ell B, Harth A. A language-independent method for the extraction of RDF verbalization templates. In: Proceedings of the 8th International Natural Language Generation Conference (INLG). Philadelphia, Pennsylvania, U.S.A.: Association for Computational Linguistics; 2014, p. 26–34. URL: http://www.aclweb.org/anthology/W14-4405.

[32] Gardent C, Shimorina A, Narayan S, Perez-Beltrachini L. The WebNLG challenge: Generating text from RDF data. In: Proceedings of the 10th International Conference on Natural Language Generation. Association for Computational Linguistics; 2017, p. 124–33. URL: http://aclweb.org/anthology/W17-3518.

[33] Sleimi A, Gardent C. Generating paraphrases from DBpedia using deep learning. In: Proceedings of the 2nd International Workshop on Natural Language Generation and the Semantic Web (WebNLG 2016). Association for Computational Linguistics; 2016, p. 54–7. URL: http://www.aclweb.org/anthology/W16-3511.

[34] Chisholm A, Radford W, Hachey B. Learning to generate one-sentence biographies from Wikidata. In: Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers. Valencia, Spain: Association for Computational Linguistics; 2017, p. 633–42. URL: http://www.aclweb.org/anthology/E17-1060.

[35] Bishop CM. Neural Networks for Pattern Recognition. New York, NY, USA: Oxford University Press, Inc.; 1995. ISBN 0198538642.

[36] Chung J, Gulcehre C, Cho K, Bengio Y. Empirical evaluation of gated recurrent neural networks on sequence modeling. CoRR 2014;abs/1412.3555.

[37] Zaremba W, Sutskever I. Learning to execute. CoRR 2014;abs/1410.4615.

[38] Rumelhart DE, Hinton GE, Williams RJ. Learning representations by back-propagating errors. Nature 1986;323:533 EP –. URL: http://dx.doi.org/10.1038/323533a0.

[39] Tieleman T, Hinton GE. RMSProp: Divide the gradient by a running average of its recent magnitude. 2012. URL: http://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf.

[40] Rush AM, Chopra S, Weston J. A neural attention model for abstractive sentence summarization. In: Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing. Lisbon, Portugal: Association for Computational Linguistics; 2015, p. 379–89. URL: http://aclweb.org/anthology/D15-1044.

[41] Daiber J, Jakob M, Hokamp C, Mendes PN. Improving efficiency and accuracy in multilingual entity extraction. In: Proceedings of the 9th International Conference on Semantic Systems. I-SEMANTICS '13; New York, NY, USA: ACM. ISBN 978-1-4503-1972-0; 2013, p. 121–4. URL: http://doi.acm.org/10.1145/2506182.2506198. doi:10.1145/2506182.2506198.

[42] Bird S, Klein E, Loper E. Natural Language Processing with Python. 1st ed.; O'Reilly Media, Inc.; 2009. ISBN 0596516495, 9780596516499.

[43] Ioffe S, Szegedy C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: Bach F, Blei D, editors. Proceedings of the 32nd International Conference on Machine Learning; vol. 37 of Proceedings of Machine Learning Research. Lille, France: PMLR; 2015, p. 448–56. URL: http://proceedings.mlr.press/v37/ioffe15.html.

[44] Papineni K, Roukos S, Ward T, Zhu WJ. BLEU: A method for automatic evaluation of machine translation. In: Proceedings of the 40th Annual Meeting on Association for Computational Linguistics. ACL '02; Stroudsburg, PA, USA: Association for Computational Linguistics; 2002, p. 311–8. doi:10.3115/1073083.1073135.

[45] Lin CY. ROUGE: A package for automatic evaluation of summaries. In: Marie-Francine Moens SS, editor. Text Summarization Branches Out: Proceedings of the ACL-04 Workshop. Barcelona, Spain: Association for Computational Linguistics; 2004, p. 74–81.

[46] Heafield K, Pouzyrevsky I, Clark JH, Koehn P. Scalable modified Kneser-Ney language model estimation. In: Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers). Sofia, Bulgaria: Association for Computational Linguistics; 2013, p. 690–6. URL: http://www.aclweb.org/anthology/P13-2121.

[47] Ngonga Ngomo AC, Bühmann L, Unger C, Lehmann J, Gerber D. Sorry, i don't speak SPARQL: Translating SPARQL queries into natural language. In: Proceedings of the 22Nd International Conference on World Wide Web. WWW '13; New York, NY, USA: ACM. ISBN 978-1-4503-2035-1; 2013, p. 977–88. URL: http://doi.acm.org/10.1145/2488388.2488473. doi:10.1145/2488388.2488473.

[48] See A, Liu PJ, Manning CD. Get to the point: Summarization with pointer-generator networks. In: Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). Association for Computational Linguistics; 2017, p. 1073–83. URL: http://www.aclweb.org/anthology/P17-1099. doi:10.18653/v1/P17-1099.

[49] Bahdanau D, Cho K, Bengio Y. Neural machine translation by jointly learning to align and translate. CoRR 2014;abs/1409.0473. URL: http://arxiv.org/abs/1409.0473.

[50] Luong T, Pham H, Manning CD. Effective approaches to attention-based neural machine translation. In: Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing. Lisbon, Portugal: Association for Computational Linguistics; 2015, p. 1412–21. URL: http://aclweb.org/anthology/D15-1166.

[51] Tu Z, Lu Z, Liu Y, Liu X, Li H. Modeling coverage for neural machine translation. In: Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). Berlin, Germany: Association for Computational Linguistics; 2016, p. 76–85. URL: http://www.aclweb.org/anthology/P16-1008.

[52] Mi H, Sankaran B, Wang Z, Ittycheriah A. Coverage embedding models for neural machine translation. In: Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing. Austin, Texas: Association for Computational Linguistics; 2016, p. 955–60. URL: https://aclweb.org/anthology/D16-1096.