Journal of Artificial Intelligence Research 69 (2020) 1-31 Submitted 09/2019; published 09/2020

Point at the Triple: Generation of Text Summaries from Knowledge Base Triples

Pavlos Vougiouklis PV1E13@ECS.SOTON.AC.UK
School of Electronics and Computer Science
University of Southampton
Southampton, United Kingdom

Eddy Maddalena EDDY.MADDALENA@KCL.AC.UK
King's College London
London, United Kingdom

Jonathon Hare JSH2@ECS.SOTON.AC.UK
School of Electronics and Computer Science
University of Southampton
Southampton, United Kingdom

Elena Simperl ELENA.SIMPERL@KCL.AC.UK
King's College London
London, United Kingdom

Abstract

We investigate the problem of generating natural language summaries from knowledge base triples. Our approach is based on a pointer-generator network, which, in addition to generating regular words from a fixed target vocabulary, is able to verbalise triples in several ways. We undertake an automatic and a human evaluation on single and open-domain summaries generation tasks. Both show that our approach significantly outperforms other data-driven baselines.

1. Introduction

Natural Language Generation (NLG) is the task of generating text that captures the content of structured-data records in a human-readable way (Reiter and Dale, 2000). In the context of knowledge graphs, structured data takes the form of triples. As an example, the DBpedia knowledge graph (Lehmann et al., 2015) contains triples such as: `dbr:Barack_Obama dbo:spouse dbr:Michelle_Obama`, where `dbr:Barack_Obama` is the subject of the triple, `dbr:Michelle_Obama` is the object, and `dbo:spouse` is the predicate (or property[1]) that connects the two to each other.

NLG systems for knowledge graphs take as input a subset of the *graph's triples* and output a *text summary* (Bouayad-Agha et al., 2014). They are deployed in various domains, from search (Li et al., 2017; Kartsaklis et al., 2018) and chatbots (Celikyilmaz et al., 2015; Ma et al., 2015) to digital humanities (Dannélls et al., 2012). Earlier attempts to generate text from triples tend to use rules or template-based techniques, which work well in domains with a regular structure and limited vocabulary (Bouayad-Agha et al., 2012). More recent proposals leverage deep learning, which was successful in similar text-generative tasks, in-

1. In this paper we will use these terms interchangeably.

Triples	Atlas_Shrugged literaryGenre Science_fiction Atlas_Shrugged country United_States John_Galt series Atlas_Shrugged Atlas_Shrugged publicationYear ‘‘1957’’ Atlas_Shrugged author Ayn_Rand
Text Summary	Atlas Shrugged is a science fiction novel by Ayn Rand.

Table 1: A simplified NLG example. Our systems generate a text summary from an input set of un-ordered triple-facts about *Atlas Shrugged*. Predicates (or properties) are highlighted in red. The items before and after each predicate are the subject and object respectively of each triple. Numerical values (e.g. 1957) can only appear in the third, object position.

cluding machine translation (Cho et al., 2014; Sutskever et al., 2014; Bahdanau et al., 2014) and text summarisation (Rush et al., 2015; See et al., 2017). Systems such as those described by Chisholm et al. (2017); Lebret et al. (2016) and Vougiouklis et al. (2018a) are able to generate coherent, relevant text for Wikipedia from structured data, without any input from linguists or domain experts. However, these newer systems are not without their challenges: while they could, in principle, scale to open-domain tasks, they only report their performance on one domain, people's biographies. In addition, they struggle to verbalise rare or previously unseen entities, which are represented as *placeholder* tokens in the output and are meant to be replaced in a post-processing step. This introduces a degree of stochastic behaviour when multiple relations from the input match the predicted placeholders.

In this paper, we present an approach that addresses these concerns. Table 1 presents a canonical NLG task. Our aim is to automatically generate a textual summary describing the graph about *Atlas Shrugged*. The input contains triples in which the given entity, in this case Atlas_Shrugged, is the subject or the object of the triples. Our approach is inspired by *pointer-generator* networks which have been recently introduced in text summarisation (Gu et al., 2016; See et al., 2017). Our systems jointly learn to: (i) verbalise the entities from *pointed* triples in several ways; (ii) copy the label or the number in the case that the pointed triple consists of either infrequent entities or numbers; and (iii) generate words or other human-readable realisations of entities from a fixed target vocabulary.

Following the methodology proposed by Vougiouklis et al. (2018a), we create a dataset encompassing the entirety of Wikipedia rather than just the biographies. We use this dataset to demonstrate our model's ability to generalise on a much more challenging task. We evaluate our approach in two ways: automatically and manually. For the former, we use the BLEU, ROUGE and METEOR metrics in order to evaluate the performance of our approach in both the widely cited task of biographies generation (Chisholm et al., 2017; Lebret et al., 2016; Liu et al., 2018; Vougiouklis et al., 2018a; Yeh et al., 2018), and the generation of open-domain Wikipedia summaries. Furthermore, we run a user study in which we explore the *fluency* and *coverage* of the summaries, as well as the presence of *contradictions*. In all scenarios, our systems outperform a variety of competing baselines of different natures. Our dataset along with the code of our systems is available at: https://github.com/pvougiou/Point-at-the-Triple.

2. Related Work

NLG commonly has three steps: (i) document planning, (ii) micro-planning, and (iii) surface realisation (Reiter and Dale, 2000; Bouayad-Agha et al., 2014). During document planning, the system selects and structures the information that will be captured in the text. The result is used by the micro-planner to decide how the information would be expressed. Finally, the realiser generates text that matches the linguistic requirements set by the micro-planner, and verbalises the information as it was structured in the first step. In traditional, rule-based NLG systems these steps are carried out independently, and are custom-built with an end-application in mind. This means that often they are optimised for a particular domain, type of language, and application capability (Reiter et al., 2005; Green, 2006; Turner et al., 2009), which makes them costly for large domains and difficult to reuse.

More recent, data-driven approaches "learn" to perform content selection and realisation in a single framework (Angeli et al., 2010; Chen and Mooney, 2008; Chen et al., 2010; Kim and Mooney, 2010; Konstas and Lapata, 2012a,b, 2013). For example, Chen and Mooney (2008) and Chen et al. (2010) learn to generate descriptions for robotic football matches (using the RoboCup dataset) by retraining a system that does supervised semantic parsing and syntax-based statistical machine translation, using an iterative algorithm similar to Expectation Maximisation (EM). A more advanced system on the same task has been proposed by Kim and Mooney. They enhanced a generative alignment model (Liang et al., 2009) with additional linguistic information produced by Lu et al. (2008)'s semantic parser. Angeli et al. introduced a system that jointly learns to perform the full NLG pipeline as a sequence of local decisions using a log-linear classifier. The end-system also leverages Liang et al.'s alignment model in order to infer the alignment between words in the text and database records. They use a set of domain-independent features for their log-linear model, which enables them to handle long-range dependencies. The final output is fluent due to several domain-specific features that are considered by their template generation system. Konstas and Lapata propose an approach based on a probabilistic context-free grammar that uses a set of trees to capture how the records from a database are rendered into text (Konstas and Lapata, 2012a,b, 2013). Text generation is then achieved by approximating the best derivation tree in the hyper-graph. In contrast to previous work using templates, in their case fluency is enhanced by intersecting the hyper-graph with an n-gram language model, which is trained separately on the dataset of interest.

There is a large body of literature that uses the encoder-decoder framework from machine translation (Cho et al., 2014; Sutskever et al., 2014; Bahdanau et al., 2014) for NLG (Sleimi and Gardent, 2016; Gardent et al., 2017; Chisholm et al., 2017; Mei et al., 2016; Lebret et al., 2016; Wiseman et al., 2017; Vougiouklis et al., 2018a; Liu et al., 2018; Li and Wan, 2018; Gehrmann et al., 2018; Yeh et al., 2018). The decoder, typically a multi-gated Recurrent Neural Network (RNN), formed of either Long Short-Term Memory cells (Hochreiter and Schmidhuber, 1997) or Gated Recurrent Units (Cho et al., 2014), is conditioned on a set of structured records and acts as a language model. Adaptation of such systems have shown great potential at tackling various aspects of triples-to-text tasks ranging from microplanning by Gardent et al. (2017) to generation of paraphrases by Sleimi and Gardent (2016). Pointer-generator networks have been brought up recently by Gu et al. (2016) and

See et al. (2017) as alterations of the original *pointer* architecture proposed by Vinyals et al. (2015). While the original model was generating an output sequence by copying tokens from the input sequence, the newer architectures are also able to generate words from the fixed vocabulary of the decoder. Overall, the results are promising, especially for domains with limited linguistic variability. For example, Mei et al. introduced a system that generates textual descriptions from datasets about weather and RoboCup football matches. In a similar context, Wiseman et al. and Li and Wan used pointer-generator networks to generate descriptions of basketball games, while Gehrmann et al. did the same for restaurant descriptions. The latter started from See et al.'s pointer-generator network, and proposed a training strategy based on ensemble learning that helped them capture distinct sentence templates.

Several authors proposed end-to-end systems to generate Wikipedia biographies (Lebret et al., 2016; Chisholm et al., 2017; Vougiouklis et al., 2018a; Liu et al., 2018; Yeh et al., 2018). Lebret et al. used a feed-forward language model with slot-value templates to generate the first sentence of a Wikipedia summary from its corresponding infobox. Liu et al.'s system outperformed them by adapting an encoder-decoder architecture equipped with LSTM cells and a novel double-attention mechanism over the input table's fields and their values. Chisholm et al. introduced a system that generates a biography given a sequence of slot-value pairs extracted from the Wikidata knowledge graph. A similar task has been tackled by Yeh et al., who used the pointer-generator network described by Gu et al. (2016) to write single-sentence biographies given slot-value pairs from Freebase, another knowledge graph, now part of Wikidata.

In all the above 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 knowledge graph triples of the corresponding Wikipedia article. As we discuss in more detail in Section 4, the input also includes triples with entities that are related to the main entity of a Wikipedia article, and their object is the main subject of the Wikipedia summary. 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. In a recent work of our own, we sought to extend the length of the generated textual content to two sentences (Vougiouklis et al., 2018a). The system described by Vougiouklis et al. has a feed-forward architecture that encodes knowledge base triples (i.e. from Wikidata and DBpedia) in a vector of fixed dimensionality, and an RNN-based decoder that generates the two-sentence summary one word at a time.

All of the above approaches use a set of *placeholder* tokens to verbalise rare or unseen entities. These tokens are replaced in a post-processing step with the part of the input that matches their requirements (for instance, field type or instance type). If more than one candidate is found, the system chooses randomly. In this paper, we propose a different approach. Our end-to-end architecture learns a model that predicts both the entities that should appear in the summary (e.g. `United_States`) and their surface form (e.g. "American", "United States" etc.). In addition, we evaluate the system on a new dataset that is extracted from the entire Wikipedia, rather than focusing on specific domains (e.g. weather, sports or biographies).

3. Our Model

We assume our model is trained on records consisting of an English language summary and a set of triples. The alignment of the elements of each triple to their realisation in the vocabulary of the summary is either *explicit* or *inferred*. For example, the triple: `Dwayne_Johnson occupation Actor` is explicitly aligned to this sentence: "Dwayne Johnson is **an actor** ...". By comparison, the alignment of `Michael_Jordan birthPlace Brooklyn` to "Michael Jordan is **an American** retired professional basketball player ..." is inferred.

Let $F_z = \{f_1, \ldots, f_E : f_i = (s_i, p_i, o_i)\}$ be the set of triples $f_1, \ldots, f_E$ about the entity z (i.e. z is either the subject or the object of the triples in the set), where s_i, p_i and o_i are the one-hot vector representations of the respective subject, predicate and object of the i-th triple. We build a model that computes the probability of generating a sequence of tokens $\mathbf{y} = y_1, y_2, \ldots, y_T$, given the initial set of triples $f_1, f_2, \ldots, f_E$:

$$p(\mathbf{y}|F) = \prod_{t=1}^{T} p(y_t|y_1, \ldots y_{t-1}, F) \ , \tag{1}$$

where $T > 1$. We regard $\mathbf{y}$ as a textual summary of the input set of triples F_z.

We build upon architectures from the literature (See et al., 2017) and our own previous work (Vougiouklis et al., 2018a). See et al. introduces a pointer-generator network capable of both copying tokens from the input sequence and generating words for the fixed vocabulary of the decoder. While this model handles sequential inputs and outputs (i.e. sentences), in our case the sets of input triples are un-ordered, and not sequentially correlated. For this reason, we use the encoder proposed in the latter work by Vougiouklis et al., and we compute the attention scores on top of this feed-forward architecture.

In many cases, the entities that participate in the properties contained in the input triples cannot be directly copied to the generated text. For example, the entities of `dbr:Actor` and `dbr:United_States` could be expressed, based on the context, as both "actor" or "actress", and "United States" and "American", respectively. To tackle this, we propose a technique that enables our model to learn different realisations for the entities of the pointed triples. The technique also helps with handling rare entities, for which we do not have good vector representations, numerical values that are in the third, object position of the pointed triples, and their labels. We discuss this in more detail in Section 3.3.

3.1 Decoder

We implement the decoder as a multi-gated RNN variant with Gated Recurrent Units (GRUs). Let $h_t^l \in \mathbb{R}^m$ be the aggregated output of a hidden unit at timestep $t = 1 \ldots T$ and layer depth $l = 1 \ldots L$. All matrices that follow have dimension $[m, m]$ unless stated otherwise. The vectors at zero layer depth, $h_t^0 = \mathbf{W}_{\mathbf{x}\rightarrow\mathbf{h}} x_t$, represent the tokens 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 tokens in the Wikipedia summaries dictionary). At each timestep t, h_t^l is computed as follows:

$$h_t^l = \text{GRU}(h_{t-1}^l, h_t^{l-1}) \ . \tag{2}$$

3.2 Triple Encoder

We compute the vector representation h_{f_i} of the i-th triple by forward propagating the triple encoder as follows:

$$\tilde{h}_{f_i} = [\mathbf{W}_{\mathbf{x}\to\tilde{\mathbf{h}}} s_i; \mathbf{W}_{\mathbf{x}\to\tilde{\mathbf{h}}} p_i; \mathbf{W}_{\mathbf{x}\to\tilde{\mathbf{h}}} o_i] \ , \tag{3}$$

$$h_{f_i} = \text{ReLU}(\mathbf{W}_{\tilde{\mathbf{h}}\to\mathbf{h}} \tilde{h}_{f_i}) \ , \tag{4}$$

where ReLU is the rectifier (i.e. non-linear activation function), $[\ldots;\ldots]$ represents vector concatenation, $\mathbf{W}_{\mathbf{x}\to\tilde{\mathbf{h}}} : \mathbb{R}^{|N|} \to \mathbb{R}^m$ 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 entities and predicates of the triples' dictionary), and $\mathbf{W}_{\tilde{\mathbf{h}}\to\mathbf{h}} : \mathbb{R}^{3m} \to \mathbb{R}^m$ is an unbiased linear mapping.

Attending the Triples Rather than asking the model to compress all available information from the triples in a single vector (Vougiouklis et al., 2018a), we implement an attention mechanism over all triples, based on works in semantic parsing (Dong and Lapata, 2016) and machine translation (Luong et al., 2015). The attention scores between the current state of the decoder h_t^L and the representation of each one of the E input triples are computed from the attention weights $\mathbf{W}_\mathbf{a} : \mathbb{R}^m \to \mathbb{R}^m$ as:

$$a_t^{(i)} = \frac{\exp[(h_t^L)^T \mathbf{W}_\mathbf{a} h_{f_i}]}{\sum_{j=1}^{E} \exp[(h_t^L)^T \mathbf{W}_\mathbf{a} h_{f_j}]} \ . \tag{5}$$

Based on the attention scores, the model computes a context vector that aggregates the information from the most important triples for the token that is to be generated at timestep t as a weighted sum over the representation of each triple of the encoder:

$$c_t = \sum_{i=1}^{E} a_t^{(i)} h_{f_i} \ . \tag{6}$$

This vector allows the decoder to selectively decide to which part of the input triples it should pay attention to. The alignment between the context vector and the information that has already been processed in a generated summary are jointly learned through trainable weights $\mathbf{W}_\mathbf{c} : \mathbb{R}^m \to \mathbb{R}^m$ and $\mathbf{W}_\mathbf{h} : \mathbb{R}^m \to \mathbb{R}^m$ as follows:

$$h_t^{L+1} = \tanh\left(\mathbf{W}_\mathbf{c} c_t + \mathbf{W}_\mathbf{h} h_t^L\right) \ . \tag{7}$$

3.3 Dynamically Expanding the Vocabulary

As discussed earlier, the pointer-generator network which has been proposed by See et al. (2017) learns to copy only a single representation per input token. This means that the system uses the same label for each copied entity regardless of the context in the text, which impacts fluency (Vougiouklis et al., 2018a). Our approach addresses this concern by learning multiple realisations for the entity of a pointed triple. We propose an architecture that can: (i) generate words from a fixed target vocabulary; (ii) copy a number of different

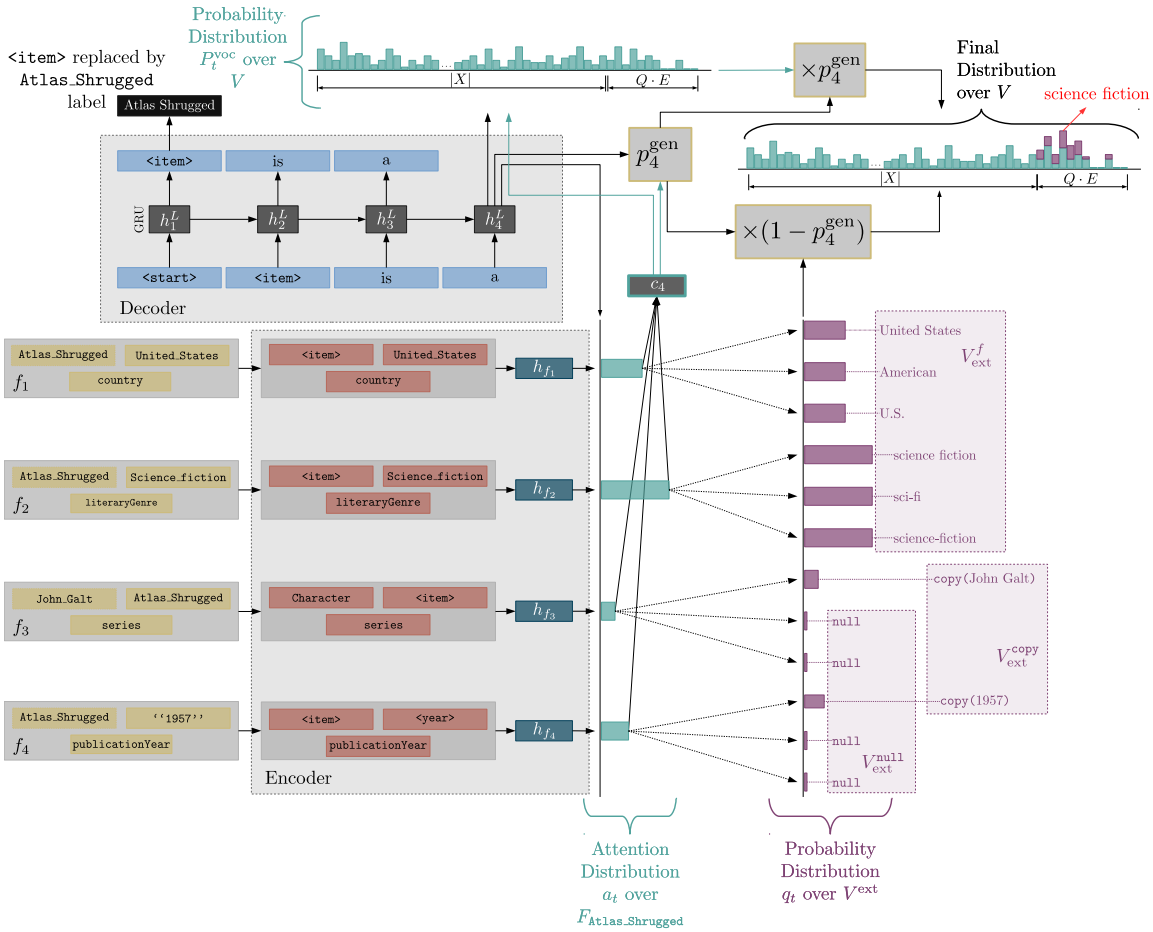


Figure 1: The architecture of our pointer-generator network. At timestep $t = 4$, p_4^{gen} weighs the probability of copying a word from V^{ext} higher than generating a word from the fixed vocabulary $V^{\dagger}$. The decoder learns to interpret the weighted sum of h_4^L and c_4 in order to compute a probability distribution for the most appropriate text realisation given the context of the triples. The attention mechanism highlights f_2 as the most important triple for the generation of the upcoming token. The attention scores are distributed among the entries of V^{ext}, and accumulated into the final distribution over V. As a result, the model copies "science fiction" that is one of the surface forms associated with f_2.

surface forms for the entities in the input triple set; or (iii) copy the number or literal for triples whose objects are numerical values or infrequent entities.

Our approach is partially inspired by how humans would perform on the same task. When provided with a set of triple-facts which they are asked to summarise in text, people would start summarising by using their own known vocabulary. However, they would focus their attention on a particular triple when they would want to realise an entity's name or a number in the text[2].

2. We base this on work by Jing (2002) that looked at how human abstractors summarise articles. The abstractors were found to reuse portions of the input text in their summaries. Similarly to our task, this

Let $K = \{k_1, k_2, \ldots, k_D\}$ be the set of all the entities that have been expressed in the textual summaries of a training dataset. In addition, let $(g_j^{k_d}, z_j^{k_d})$ be the tuple of the j-th surface form $g_j^{k_d}$ of the entity $k_d \in K$, along with the number of occurrences of the realisation $g_j^{k_d}$ in the dataset, $z_j^{k_d} \in \mathbb{N}$. Additionally, let $l^{k_d} = \{(g_1^{k_d}, z_1^{k_d}), (g_2^{k_d}, z_2^{k_d}), \ldots, (g_R^{k_d}, z_R^{k_d})\}$ be the partially ordered set of the triples that are associated with the entity k_d, s.t. $z_j^{k_d} \geq z_{j+1}^{k_d} \forall j \in [1, R-1]$, where R is the total number of realisations of k_d. We compute the 95th percentile of the number of all the possible textual realisations of k_d, q^{k_d}. We define $G^{k_d} = \{g_1^{k_d}, g_2^{k_d}, \ldots, g_Q^{k_d}\}$ s.t. $Q \leq R$ as the set of all possible verbalisations through which our model learns to express k_d in the generated summary. Q is a dataset-specific hyper-parameter for our model, and is calculated by averaging the number of possible realisations $q^{k_d}\ \forall k_d \in K$.

Let $H^{(f)}$ and $H^{(i)}$ be the sets of all the frequent and infrequent entities that participate in the triples (see first paragraph of Section 5.1). In addition, let $\mathcal{E} = \{e_1, e_2, \ldots, e_E\}$ s.t. $e_j \in (s_j, o_j)$ and $e_j \neq z\ \forall j \in [1, E]$ be the set of all the items (numerical values or entities) other than entity z that participate in the corresponding relationships in F. We assume a fixed target vocabulary $V^\dagger = \{v_1^\dagger, v_2^\dagger, \ldots, v_{|X|}^\dagger\}$. In comparison to similar pointer-generator networks that expand the decoder's fixed vocabulary by the length of their input E, we expand it by $Q \cdot E$, and we define the dynamic vocabulary extension (where the values are based on the input triples), $V^{\text{ext}} = \{v_1^{\text{ext}}, v_2^{\text{ext}}, \ldots, v_{Q \cdot E}^{\text{ext}}\}$ along with its subsets V_{ext}^f, $V_{\text{ext}}^{\texttt{copy}}$ and $V_{\text{ext}}^{\texttt{null}}$, s.t.:

$$v_j^{\text{ext}} = \begin{cases} g_{j\%Q}^{e_{\lceil j/Q \rceil}} \in V_{\text{ext}}^f & e_{\lceil j/Q \rceil} \in H^{(f)} \\ g_1^{e_{\lceil j/Q \rceil}} \in V_{\text{ext}}^{\texttt{copy}} & e_{\lceil j/Q \rceil} \in H^{(i)} \text{ and } j\%Q = 1 \\ g_1^{e_{\lceil j/Q \rceil}} \in V_{\text{ext}}^{\texttt{copy}} & e_{\lceil j/Q \rceil} \in \mathbb{R} \text{ and } j\%Q = 1 \\ \texttt{null} \in V_{\text{ext}}^{\texttt{null}} & \text{otherwise} \end{cases} \tag{8}$$

$\forall j \in [1, Q \cdot E]$, where $\lceil \ldots \rceil$ represents the ceiling function.

During both training and testing, for each set of input triples we form the values of the extended vocabulary V^{ext}. Each triple is provided with Q slots in V^{ext}. For example in Figure 1 where $Q = 3$, the frequent entity `United_States` $\in H^{(f)}$ results in the inclusion of "United States", "American" and "U.S." in the vocabulary extension V^{ext}. In case a rare entity is either the subject or the object of a triple in the triple set, it is replaced by its corresponding instance type token before it is provided to our model (e.g. `John_Galt` is replaced by the `Character` token when it is inputted in the triple encoder in Figure 1). In such scenario, all values of V^{ext} that correspond to this particular triple are filled with `null`, except the first one which refers to the copy of the label of this rare entity. A similar methodology is used for numbers[3]. For instance, in Figure 1, ``1957'' is replaced by the

was notable in the case of copy-pasting named entities from the original article into the resulting text. However, they would still perform syntactical transformations and lexical paraphrasing using their own vocabulary in order to produce more compact and less "noisy" summaries.

3. In the current implementation, we focus on years only, but the approach would work all the same for any other type of numerical data types.

<year> token, and the first slot of the positions in V^{ext} that correspond to this triple is filled with the copy of this token.

3.4 Summarising By Pointing and Generating

The probability distribution q_t for each entry in the vocabulary extension V^{ext} after distributing the attention scores over the realisations of the relevant triples is computed as follows:

$$\widetilde{q}_t^{(i)} = \begin{cases} \exp[a_t^{(\lceil i/Q \rceil)}] & v_i^{\text{ext}} \in V_{\text{ext}}^{f} \cup V_{\text{ext}}^{\texttt{copy}} \\ 0 & v_i^{\text{ext}} \in V_{\text{ext}}^{\texttt{null}} \end{cases} \tag{9}$$

$$q_t^{(i)} = \frac{\widetilde{q}_t^{(i)}}{\sum_{j=1}^{Q \cdot E} \widetilde{q}_t^{(j)}} \tag{10}$$

$\forall i \in [1, Q \cdot E]$. We adopt the notion of *generation probability* $p_t^{\text{gen}} \in [0, 1]$, to simulate a soft switch at each timestep t between generating a token from the fixed vocabulary or copying either the surface form or the label of an entity from the highlighted triple (See et al., 2017).

$$p_t^{\text{gen}} = \text{sigm}(\mathbf{W}_{\hat{\mathbf{c}}} c_t + \mathbf{W}_{\hat{\mathbf{h}}} h_t^L) \ , \tag{11}$$

where $\mathbf{W}_{\hat{\mathbf{c}}} :: \mathbb{R}^m \rightarrow \mathbb{R}^1$ and $\mathbf{W}_{\hat{\mathbf{h}}} :: \mathbb{R}^m \rightarrow \mathbb{R}^1$ are biased linear mappings.

Our model computes the following probability distribution for each entry w in the extended vocabulary $V = V^{\dagger} \cup V^{\text{ext}}$ as follows:

$$P_t(w) = \begin{cases} p_t^{\text{gen}} P_t^{\text{voc}}(w) + (1 - p_t^{\text{gen}}) q_t^{(w)} & w \in V_{\text{ext}}^{f} \cup V_{\text{ext}}^{\texttt{copy}} \\ p_t^{\text{gen}} P_t^{\text{voc}}(w) & w \in V^{\dagger} \\ 0 & w \in V_{\text{ext}}^{\texttt{null}} \ , \end{cases} \tag{12}$$

where $P_t^{\text{voc}} = \text{softmax}(\mathbf{W}_{\mathbf{y}} h_t^{L+1})$, and $\mathbf{W}_{\mathbf{y}} : \mathbb{R}^m \rightarrow \mathbb{R}^{|X|+Q \cdot E}$ is a trainable weight matrix.

The decoder learns to interpret h_t^{L+1} to make a decision about which realisation is the most appropriate over the V given the context from both the input and the text that it has generated so far. The attention scores point at the triple that should be verbalised in the summary. The model makes the final prediction about the token that will be outputted only after these scores are accumulated in the final distribution over V.

While our formulation for Q favours batch-level operation, its functionality in our system goes beyond that. Since Q remains unaltered for all input triples, our model is able to easily learn the association of each triple position with its corresponding Q slots in V^{ext} easier, throughout the training process. Furthermore, while the position of the input entities can vary, its verbalisations and their order in V^{ext} are consistent across all training and testing. During the initial training stages, the model learns that realisations of the $j(+Q)$-th entry from V^{ext} is usually attributed to their co-existence with certain entities in the $\lceil j/Q \rceil$ position of $\mathcal{E}$. Since the possible realisations of the entities remain in the same order, at subsequent training stages, the decoder learns to interpret h_t^{L+1} in order to choose the most appropriate entry from the Q realisations, that correspond to the single triple to which the attention mechanism pays most attention.

In contrast to the architecture proposed by See et al. (2017), our model does not aggregate the probabilities of potentially common entries in the extended and the fixed target vocabulary, V^{ext} and $V^{\dagger}$ respectively. This enables us to achieve better separation of the particular action required at each timestep, in terms of either selecting a regular word or explicitly realising a triple using the corresponding extended vocabulary. Furthermore, in comparison to See et al. who extend the fixed target vocabulary by only the single-token entries of the source sequence, our extended vocabulary consists of both single- and multi-token labels, such as "sci-fi" and "science fiction", in Figure 1, respectively. Since all entries of $V^{\dagger}$ consist of single tokens, aggregating probabilities only for potentially common single-token labels would substantially skew the resulting distribution over V against their multi-token counterparts. Our model is also agnostic with respect to the number of tokens of which each entry of V^{ext} consists. Consequently, both single- and multi-token labels are directly appended to a summary should their corresponding entry of V^{ext} is selected during the decoding stage.

4. Datasets

We train and evaluate our system on two corpora. The first is the D1 Biographies corpus provided by Vougiouklis et al. (2018a), which consists of triples from DBpedia aligned with Wikipedia biographies. The second corpus, which we refer to as the Full corpus, has been built for the purpose of this paper. It uses the same methodology as the one described by Vougiouklis et al., applied, however, to the entire Wikipedia. Table 2 provides statistics for both.

Parameter	Biographies	Full
# Articles	256850	864862
# Entities	609k	1173k
# Predicates	450	1124
Vocab. Size	400k	1114k
Avg. # Triples / Article	10.68 (7.87)	8.59 (6.33)
Avg. # Tokens / Summary	41.30 (17.83)	39.95 (21.05)

Table 2: Statistics of the two corpora. Average parameters are shown with standard deviations in brackets.

We leverage the intrinsic alignment of DBpedia and Wikipedia in order to create a corpus of loosely aligned triples and text summaries. We extracted DBpedia triples from the Mapping-based Objects[4] and Literals[4] DBpedia datasets. All relevant Wikipedia summaries were extracted using the Long Abstracts[4] DBpedia dataset retaining only articles with at least a single triple in the Mapping-based Objects and Literals corpora. We retained only the first two sentences of each Wikipedia summary.

Entities in the summaries are identified using DBpedia Spotlight (Daiber et al., 2013). We excluded any Wikipedia summaries whose main discussed entity (e.g. *Atlas Shrugged*

4. http://wiki.dbpedia.org/downloads-2016-10

in the example of Table 1) was not identified in the text. 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. 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. Table 3 presents the distribution of the 10 most common predicates, and entities in the resulting Full dataset. We urge interested readers to refer to our previous work (Vougiouklis et al., 2018a) for further details about the dataset building process.

Predicates In Triples		**Entities In Triples**		**Entities In Summaries**	
Predicate	%	Entity	%	Entity	%
birthDate	4.52	United_States	0.56	United_States	2.23
birthPlace	3.92	United_Kingdom	0.25	Mile	0.72
country	3.47	India	0.15	Actor	0.64
isPartOf	2.83	France	0.14	Town	0.60
genre	2.79	Canada	0.12	Village	0.59
location	2.12	England	0.12	England	0.54
careerStation	2.00	Animal	0.12	Association_football	0.49
type	1.97	Italy	0.11	City	0.48
starring	1.92	Germany	0.10	Germany	0.43
occupation	1.89	Australia	0.10	France	0.43

Table 3: Distribution of the 10 most frequent predicates and entities in the Full dataset. The left and middle columns display the distribution of the 10 most common predicates and entities in the triples that have been allocated to the Full dataset. The right column depicts the distribution of the 10 most common entities in the Wikipedia summaries as they have been identified using DBpedia Spotlight.

After an entity is annotated in the text, its realisation is replaced by a *surface form tuple* that consists from this realisation and the name of the entity. These annotations are used for the construction of the G^{k_d} sets for each entity k^d (cf. Section 3.3) that the approach determines in the summaries of each corpus. When the realisation of an annotated entity or a year in the text is identified in the V^{ext} of an input set of triples, it is replaced by the token of the position of the surface form in V^{ext}. Table 4 shows an example of the alignment of the datasets, after some pre-processing. We note that the output of DBpedia Spotlight might not always be ideal. For example, during data preparation, only the first part of "Bosnia and Herzegovina" in the original summary has been identified as realisation of `dbr:Bosnia_and_Herzegovina`. However, our model during generation should be able to first select the first realisation from V^{ext} for `dbr:Bosnia_and_Herzegovina`, and then sample the remaining tokens for "and" and "Herzegovina" from the fixed target vocabulary, $V^{\dagger}$, at the subsequent decoding timesteps.

For each input triple set, we determine the values of the extended vocabulary V^{ext}. Each triple has Q slots in V^{ext}. Years which have not been identified in the input set of triples are mapped to a pre-defined `<year>` token. Regular numbers are replaced by the special `0` token and are not considered further in the evaluation. Every out-of-vocabulary

token in summaries is represented by the special <rare> token in the case of words, and their corresponding instance-type token (e.g. dbo:SoccerPlayer) in the case of surface form tuples. The special tokens of an entity's instance type are retrieved from the Instance Types DBPedia[5] dataset. We compute a Q value of 2 (for the D1 Biographies dataset), and 3 (for the Full corpus), which results in $\sim 98\%$ coverage of the total number of textual realisations of the triples' entities for both corpora (see Section 3.3).

		V^{ext}
<item>	dbr:Čizma	
Triples	<item> dbo:country dbr:Bosnia_and_Herzegovina [dbo:Country]	1: Bosnia 2: Bosian 3: Bosnia-Herzegovina
	<item> dbo:isPartOf dbr:Kiseljak [dbo:Settlement]	4: Kiseljak 5: null 6: null
	<item> dbo:timeZone dbr:Central_European_Time [unknown_type]	7: CET 8: cet 9: UTC+2
	<item> dbo:type dbr:Village [owl#Thing]	10: village 11: rural community 12: selo
	<item> dbo:utcOffset 0 [unknown_type]	13: 0 14: null 15: null
Original Summary	Čizma is a village in the municipality of Kiseljak, Bosnia and Herzegovina.	
Annotated Summary	<start> <item> is a (dbr:Village, village) in the (dbr:Municipalities_of_Bosnia_and_Herzegovina, municipality) of (dbr:Kiseljak, Kiseljak) , (dbr:Bosnia_and_Herzegovina, Bosnia) and Herzegovina . <end>	
Summary w/ Surf. Form Tuples	<start> <item> is a [ext_10] in the (dbr:Municipalities_of_Bosnia_and_Herzegovina, municipality) of [ext_4] , [ext_1] and Herzegovina . <end>	
Summary w/o Surf. Form Tuples	<start> <item> is a [ext_10] in the municipality of [ext_4] , [ext_1] and Herzegovina . <end>	

Table 4: An example alignment in our dataset. Each triple has $Q = 3$ slots in V^{ext}. The main entity in both the input triples and the summaries is replaced with the <item> token. Each triple is stored along with the instance type of the other, besides the main, entities it contains (e.g. [dbo:Settlement] for dbr:Kiseljak). The word "village" in the summary is recognised as the DBpedia entity dbr:Village, which is mentioned in the triples of the input set. Since "village" is also one of the realisations of dbr:Village in the 10th position of V^{ext}, it is replaced by the [ext_10] token in the text. Each summary is augmented with start-of-summary <start> and end-of-summary <end> tokens.

5. Available at http://wiki.dbpedia.org/downloads-2016-10.

In order to constrain the space complexity of the task, we limit the number of triples E that are allocated to each summary to:

$$\lfloor E_{\min} + 0.25\sigma_E \rfloor \leq E \leq \lfloor \overline{E} + 2\sigma_E \rfloor \quad . \tag{13}$$

With this limitation, the model size of the pointer-generator systems is around 10 GB, which is comparable with the available GPU memory of the Titan X (Pascal) GPU used for the evaluation. When the number of triples of an input set exceeds this threshold, we filter out any potential redundant triples from the oversized sets (Vougiouklis et al., 2018a), and prioritise triples whose objects or subjects have been mentioned in the text. We found that triple sets whose size exceeds the above threshold account for 4% and 4.7% in the case of the Full and Biographies dataset respectively. We perform this during training in order to maximise the effect of the proposed pointer mechanism and the property-type placeholders of the competing Triples2GRU and Triples2LSTM (see Section 5.2) models. During testing, we first attempt to exclude potential duplicates; in case their number still exceeds the limit, we retain only the first ones until the $E_{\max}$ threshold is reached (Vougiouklis et al., 2018a).

5. Experiments

We trained and evaluated the performance of our approach on the two corpora from the previous section. Both datasets were split into training, validation and test, corresponding to 85%, 10%, and 5% of the data. We implemented our neural network models using Torch[6]. We make the Full corpus along with the code of our system available on GitHub[7].

Furthermore, we investigated whether the inclusion of the frequent surface form tuples (Vougiouklis et al., 2018a), whose entities had not been associated with any triple from the input set, enhanced the performance of our models. We see surface form tuples as a mechanism of implicitly realising entities in the text—for instance, generating a summary about "... an American *author*" when the only relevant input triple is *author* `birthPlace` `Brooklyn`. In our prior work, we found that the inclusion of surface form tuples can generally improve the performance since the model learns easier to correlate particular realisation of entities (that can consist of more than a single token) with particular predicate-entities patterns from the input triple set (Vougiouklis et al., 2018a). However, the Triples2GRU and Triples2LSTM systems had the ability to directly verbalise entries from the input (i.e. by copying the corresponding labels using the property-type placeholder mechanism) only in the case of infrequent input entities. Consequently, we sought to explore the effect of the surface form tuples on our new architecture which due to its pointer mechanism has the ability to directly realise all input entities.

For each dataset, we ran two sets of experiments: one in which the surface form tuples were part of the fixed vocabulary of the decoder (w/ Surf. Form Tuples), and one in which they were treated as regular words (w/o Surf. Form Tuples). At test time, our systems are provided with the G^{k_d} sets of the up to Q verbalisations with which each k_d entity in our knowledge graph can be realised in the texts. Given an input set of triples, a textual summary is generated after V^{ext} is formed from the G^{k_d} sets of the frequent entities and the single labels of both the infrequent ones and the numerical objects that participate in

6. http://torch.ch
7. https://github.com/pvougiou/Point-at-the-Triple

the input. The expected output of our model, in the case of the former and latter setup, is displayed in the "Summary w/ Surf. Form Tuples" and "Summary w/o Surf. Form Tuples" rows of Table 4 respectively. At a post-processing step, the special `<item>` token is replaced by the label of the main entity, and any occurrences of `[ext_1−Q·E]` tokens with their respective entries in V^{ext}.

5.1 Training Details

On the encoder side, we included all entities and properties that occurred at least 30 times in each dataset. Triples with rare properties were excluded, while infrequent entities were replaced with instance-type tokens. This resulted in a source vocabulary size of $|N| = 5785$ for the Biographies dataset, and $|N| = 17146$ tokens for the Full dataset.

On the decoder side, we used a single layer of 500 GRUs, and included the $|X| = 15$k and $|X| = 17$k more frequent tokens (i.e. only words in the case of w/o Surf. Form Tuples systems, and words and surface form tuples in the case of w/ Surf. Form Tuples systems) from the two datasets. In all experiments, we set the dimensionality of the hidden states to $m = 500$. We initialised all parameters with a random uniform distribution between -0.1 and 0.1, and used batch normalisation before each non-linear activation function and after each fully-connected layer (Ioffe and Szegedy, 2015) on the encoder side.

Our training objective was to minimise the sum of the negative log-likelihoods of a mini-batch of 80 predicted summaries. Optimisation was performed using Adam (Kingma and Ba, 2014) with a learning rate of $5 \cdot 10^{-5}$. An l_2 regularisation term of 0.05 over the parameters was also included in the cost function.

For Biographies, the networks converged after the 13th epoch[8]. For the Full dataset, convergence was achieved after 95 epochs in the w/o Surf. Form Tuples system, and after 80 in the case of w/ Surf. Form Tuples. All systems were trained on a single Titan X (Pascal) GPU. The pointer-generator networks completed an epoch of training in around 36 minutes when trained on biographies, and in around 2 hours for the Full dataset.

During testing and evaluation, we did beam-search (Sutskever et al., 2014; Rush et al., 2015; Vougiouklis et al., 2018a) with a beam size of 8 and retained only the the summary with the highest probability.

5.2 Baselines

We demonstrate the effectiveness of our approach by comparing it against a set of competitive baselines.

Random We computed the expected lower bounds for performance across all metrics by using a random Wikipedia summary baseline. For each set of triples in the validation and test set, the system retrieves a response by randomly selecting a summary from the training set.

KN We used the KenLM toolkit (Heafield et al., 2013) to train a 5-gram Kneser-Ney (KN) language model.

8. The epoch at which the model converged to the lowest possible validation error. After this epoch, the error on the validation set either did not improve further or it increased, causing the models to overfit.

IR We implemented an Information Retrieval (IR) baseline similar to the one used for abstractive sentence summarisation (Rush et al., 2015). It encodes the sets of input triples in the training set using TF-IDF followed by LSA (Halko et al., 2011). For each set of triples in the validation and test set, we performed K-Nearest Neighbours to retrieve the closest vector from the training set, and output the related summary.

Triples2GRU and Triples2LSTM We compared against the encoder-decoder architectures from our prior work (Vougiouklis et al., 2018a). Both systems are equipped with the surface form tuples mechanism. We set the dimensionality of their hidden state to 500.

Pointer-Generator We trained and tested an adaptation of the original pointer-generator system proposed by See et al. (2017). Similarly to our approach, "Pointer-Generator" uses a feed-forward encoder that processes the input triples (see Section 3.2) and an RNN decoder based on GRUs that attends the input. The decoder can generate words from a fixed target vocabulary or by copying them from the input without, however, being able to learn different entity realisations. Parameter Q, discussed in Section 3.3, was set to 1.

All baselines included the special `<item>` tokens, and except Pointer-Generator, all of them were also equipped with the property-type placeholders (Vougiouklis et al., 2018a). After a summary was generated, the first were replaced by the label of the main discussed entity, and the second by the label of the entity of the triple from the input set that satisfied the requirements of the placeholder. In the case of the Pointer-Generator system, entities from the triples were realised in the generated summaries using the corresponding input's extended vocabulary V^{ext} with $Q = 1$. During testing, in the case of KN, Triples2GRU, Triples2LSTM and Pointer-Generator, we perform beam-search with a beam size of 8 in order to sample the most likely summaries for each set of triples.

5.3 Automatic Evaluation

We report the results in terms of the following three metrics on the validation and test sets: (i) BLEU (Bilingual Evaluation Understudy) (Papineni et al., 2002), (ii) ROUGE (Recall-Oriented Understudy for Gisting Evaluation) (Lin, 2004), and (iii) METEOR (Lavie and Agarwal, 2007). BLEU is a precision-oriented metric for measuring the quality of generated text by comparing it to the actual, empirical text. BLEU-n calculates similarity scores based on the co-occurrence of up to n-grams in the generated and the empirical text. ROUGE is a metric that computes the recall of n-grams in the generated text with respect to the n-grams of the actual text. ROUGE_L is a variant of ROUGE based on the longest common sequence in the two texts. METEOR computes a weighted average of the precision and recall of uni-grams in the generated and the empirical text by considering stemming, synonyms and paraphrases.

We adapted the code from the evaluation package that was released by Peter Anderson[9], originally used to score the quality of image captions.

5.3.1 Results

We report the results for BLEU 2, BLEU 3, BLEU 4, ROUGE_L and METEOR in Table 5. In almost all scenarios, we outperformed the baselines. On the Full corpus, both our systems

9. http://github.com/peteanderson80/coco-caption

	Model	BLEU 2		BLEU 3		BLEU 4		ROUGE_L		METEOR	
		Valid.	Test	Valid.	Test	Valid.	Test	Valid.	Test	Valid.	Test
Biographies	Random	15.78 (±.01)	15.56 (±.02)	10.25 (±.00)	10.09 (±.01)	6.71 (±.00)	6.59 (±.01)	27.68 (±.01)	27.51 (±.02)	14.15 (±.01)	14.07 (±.01)
	KN	15.16	14.97	11.00	10.85	8.04	7.94	36.74	36.47	31.46	31.32
	IR	24.60	24.44	18.16	17.96	13.66	13.45	36.20	36.08	18.12	18.02
	Triples2GRU	25.54	25.18	20.46	20.14	16.35	16.07	48.20	47.87	33.38	33.19
	Triples2LSTM	25.27	24.99	20.33	20.08	16.28	16.08	48.48	48.23	33.96	33.84
	Pointer-Generator	28.78	28.77	24.40	24.40	21.13	21.14	49.63	49.77	34.57	34.68
	Ours w/o Surf. Form Tuples	**29.40**	**29.27**	**24.79**	**24.67**	**21.39**	21.26	49.11	49.12	33.92	33.98
	Ours w/ Surf. Form Tuples	28.68	28.81	24.37	24.54	21.15	**21.34**	**49.65**	**49.90**	**34.96**	**35.30**
Full	Random	13.13 (±.01)	13.17 (±.01)	7.92 (±.01)	7.95 (±.01)	4.92 (±.01)	4.94 (±.01)	23.57 (±.01)	23.59 (±.01)	11.34 (±.01)	11.34 (±.00)
	KN	11.98	11.99	8.07	8.06	5.55	5.53	28.49	28.50	16.50	16.50
	IR	27.24	27.09	20.88	20.68	16.66	16.45	38.62	38.46	17.77	17.59
	Triples2GRU	27.21	27.13	21.84	21.73	17.87	17.73	**47.33**	**47.36**	27.21	27.23
	Triples2LSTM	25.89	25.85	20.56	20.48	16.68	16.58	46.20	46.23	27.08	27.03
	Pointer-Generator	27.45	27.27	22.24	22.08	18.45	18.30	46.86	46.73	**30.02**	**29.88**
	Ours w/o Surf. Form Tuples	28.05	27.97	**22.84**	22.73	19.07	18.95	**47.21**	47.14	29.50	29.47
	Ours w/ Surf. Form Tuples	**28.07**	**28.11**	22.81	**22.86**	**19.04**	**19.09**	47.02	**47.20**	28.20	28.31

Table 5: Automatic evaluation of our architectures against baselines using BLEU $2-4$, ROUGE_L and METEOR on the validation and test sets of the Biographies and Full corpora. The average performance of the random baseline along with its standard deviation is reported after sampling 10 times.

achieved an improvement that ranges from 0.59 to 0.79 and 0.16 to 0.47 BLEU 4 and ROUGE points, respectively, in comparison to Pointer-Generator, which is our strongest competitor. On the Biographies corpus, we can show small improvements of at least 0.02 in terms of both BLEU and ROUGE points. We believe that the lower performance on the smaller corpus is mainly a function of the limited linguistic variability of biographies—in 92.67% of the cases, entities from the triples in that corpus were realised in the text with their most frequent surface form. Our approach, which is more ambitious in terms of entity verbalisations, paid a higher cost than Pointer-Generator, which relies on a single label per entity ($Q = 1$). The advantages of our approach become more clear on the Full corpus, which is linguistically more challenging and includes entities with varied realisations—in that corpus, in 14% of the cases, entities were realised with their 2nd or 3rd label. The high METEOR results of the Pointer-Generator also indicate that it often generated text that differed from the empirical summaries due to morphological or synonymic variations. In Section 5.4, we explore the extent to which these variations influence people's perceptions of the fluency of the generated summaries.

The two corpora overlap. Table 6 shows how the models trained on the Full dataset performed on validation and test triples that are also in the Biographies corpus. We note

that when we trained our systems on the more challenging task, with the same hyper-parameters for the dimensionality of the hidden states and number of layers, we were still able to achieve reasonably good results compared to the systems trained specifically on the narrower task. Our pointer-generator network w/o Surf. Form Tuples could generate biographies regardless of the complexity of the dataset (Full or Biographies) on which it was originally trained.

Model	**BLEU 2**		**BLEU 3**		**BLEU 4**		**ROUGE$_L$**		**METEOR**	
	Valid.	Test	Valid.	Test	Valid.	Test	Valid.	Test	Valid.	Test
KN	10.13	10.16	6.47	6.46	3.64	3.61	25.31	25.33	15.84	15.81
IR	24.23	24.37	17.75	17.82	13.24	13.27	35.81	35.91	17.58	17.48
Triples2LSTM	24.41	24.64	19.32	19.47	15.37	15.46	47.06	47.41	30.73	30.71
Triples2GRU	24.95	25.06	19.92	19.99	15.88	15.90	47.71	47.99	32.56	32.52
Pointer-Generator	27.54	27.62	23.21	23.39	19.95	20.17	48.94	49.08	35.09	35.12
Ours w/o Surf. Form Tuples	**28.77**	**28.92**	**24.34**	**24.48**	**21.05**	**21.18**	49.37	49.47	33.14	33.20
Ours w/ Surf. Form Tuples	27.76	27.64	23.49	23.37	20.30	20.17	**49.47**	**49.51**	**34.19**	**34.42**

Table 6: Automatic evaluation of our systems against all other baselines using BLEU $2-4$, ROUGE$_L$ and METEOR, trained on the Full corpus, on the triples from the validation and test set that are also in the Biographies dataset.

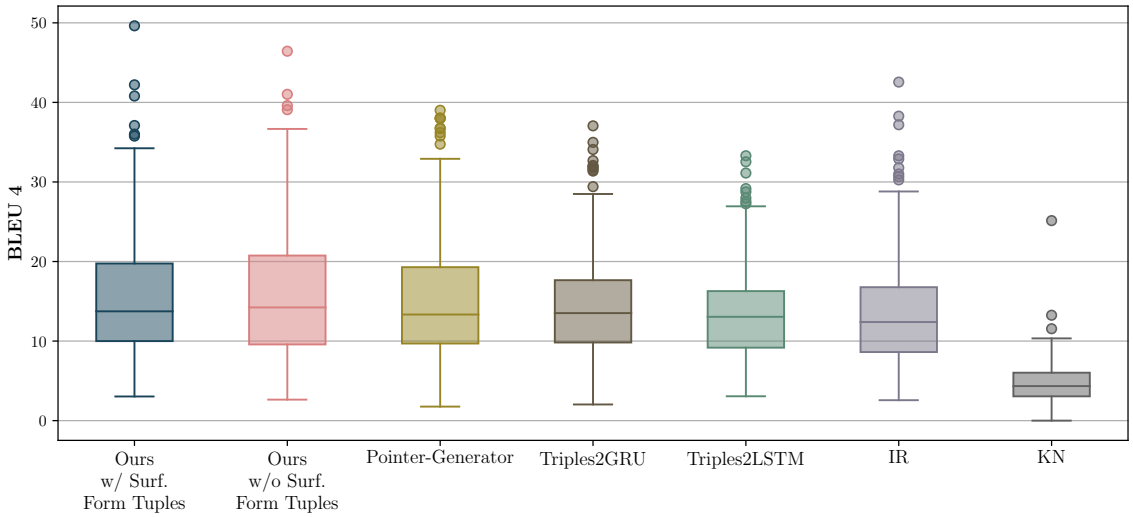


Figure 2: Performance across 225 domains

To understand how well our systems generalise across different domains, we grouped the Wikipedia summaries from the Full dataset according to the instance type of their main entity (e.g. `dbo:Village` and `dbo:SoccerPlayer`). Figure 2 shows performance results against the baselines across the 99th percentile of the included instance types (i.e. ~ 225 different instance types). Furthermore, Table 7 shows the BLEU 4 performance of our two

Instance Type	Triple Coverage	BLEU 4			
		Ours w/ Surf. Form Tuples	Ours w/o Surf. Form Tuples	Pointer-Generator	Triples2GRU
dbo:Village	11.36 (±5.58)	37.10	**39.09**	36.20	34.98
dbo:Mollusca	11.09 (±5.82)	32.99	32.59	**35.77**	34.08
dbo:Congressman	10.85 (±4.37)	**42.20**	41.02	39.01	25.57
dbo:IceHockeyPlayer	10.58 (±4.54)	36.01	**39.60**	38.02	32.68
dbo:City	10.53 (±6.10)	30.08	31.31	26.26	**31.56**
⋮	⋮	⋮	⋮	⋮	⋮
dbo:RugbyPlayer	2.39 (±3.42)	**21.91**	19.90	19.28	19.40
dbo:Country	2.24 (±2.74)	3.20	**5.42**	4.58	4.65
dbo:ComicsCreator	1.99 (±2.75)	13.94	**15.09**	13.47	13.37
dbo:ComicsCharacter	1.53 (±2.60)	19.02	**21.67**	17.93	20.31
owl#Thing	0.95 (±1.89)	5.36	5.75	5.90	**6.01**

Table 7: BLEU 4 performance of our systems against the top performing baselines across different domains (i.e. instance types). The domains with the greatest coverage with respect to the triples whose content is directly copied in the text appear on the top part of the table; domains with the least coverage are at the bottom. Only instance types with more than 20 generated summaries in the test sets were considered.

systems against the two stronger baselines in the domains (i.e. instance types) with the most and least coverage with respect to the triples whose content is directly copied in the text. The lowest performance bounds of our systems are similar to the ones of Triples2GRU and Pointer-Generator and higher than the ones of the other baselines. However, in domains with greater coverage with respect to the triples that are verbalised in the text, both our systems either outperformed or were on par with the competition.

For example, `dbo:Village` and `dbo:IceHockeyPlayer` were one of the highest scored instance types[10] for both Triples2GRU and Pointer-Generator. The two achieve a BLEU 4 scores of 34.98 and 32.68, respectively using Triples2GRU and 36.20 and 38.02 using the Pointer-Generator system. By comparison, they are scored at the same level by the w/ Surf. Form Tuples system (i.e. with respective BLEU 4 scores of 37.10 and 36.01) and are outperformed by the one w/o Surf. Form Tuples with BLEU 4 performance of 39.09 and 39.60 respectively. Moreover, in summaries about `dbo:Congressman` and `dbo:MixedMartialArtsEvent`, which are among the domains in which our systems achieve their best performance, the competing baseline are significantly outperformed (cf. Table 7).

5.4 Human Evaluation

The three metrics we used for the automatic evaluation do not capture performance well in tasks where the input and the output are loosely correlated (Reiter, 2010). Furthermore, while our pointer-generator systems (with or without surface form tuples) outperformed the competition, the difference to some of the stronger baselines were not substantial enough to allow one to choose among them. To understand how well our approach would do in

10. Only instance types with more than 20 generated summaries in the test sets were considered.

practice, we undertook two user studies with participants recruited from the Figure Eight crowdsourcing platform[11]. In the first one, we looked at the performance of our networks against the two most competitive baselines, Triples2GRU and Pointer-Generator, on the open-domain task. In the second one, we explored whether training our systems on the Full dataset with the same hyper-parameters (except the size of the input/output vocabularies) would yield results comparable to systems that were trained on the biographies task.

5.4.1 Inputs and Outputs

We included only input sets of at least six triples—our experiments have shown that all systems from the automatic evaluation did not do well on smaller inputs since they tend to lack adequate information to form a two-sentence summary. Both studies used sets of triples that occurred in the test sets of all four systems of interest (ours two plus two stronger baselines). From those, for the first study we sampled 32 sets of triples according to the instance-type distribution of the main discussed entities in the Full dataset. For the second study, we used a random sample of the same size. In both cases, we took the summaries generated by the four systems and asked 10 participants to assess them against three criteria: (i) *fluency*; (ii) *coverage*[12], which is concerned with the triples in the input whose content is mentioned either implicitly or explicitly in the text; and (iii) *contradiction*, which refers to information that is conveyed by the sentence, but conflicts with one or more of triples from the input set.

In addition to the results from our crowdsourcing experiment, we report the expected upper bounds for coverage and contradiction by manually annotating the empirical Wikipedia summaries that correspond to the selected triples sets.

5.4.2 Selecting Participants

For each of the 32 sets of triples used in each study, we also compiled a gold standard of 8 additional sets of triples with manually written summaries. The goal was to form summaries that are straightforward to assess against the three criteria to allow us to filter out crowdworkers who would perform poorly on the tasks, as explained below.

5.4.3 Tasks Design

We designed three crowdsourcing tasks, one for fluency, one for coverage, and one for contradiction. In previous work of ours (Vougiouklis et al., 2018b), we piloted alternative designs that covered two or even all three aspects, but they proved to be challenging for the participants. Both the analysis of the answers and qualitative feedback from the participants suggested that mixing the three activities increased the cognitive effort of the tasks and impacted on the accuracy of the results. In particular, some participants seemed to believe contradictions were complimentary to coverage and scored the former higher when the text was simply missing triples from the input.

11. https://www.figure-eight.com
12. Based on the notion of coverage described by Ell and Harth (2014). They use coverage to measure the number of sub-graphs that are included in the text.

Fluency Participants had to rank the coherence and grammatical correctness of four summaries, one per competing system, for a given set of triples. A task consisted of five such ranking questions, of which one was a gold standard example created by us, for a fee of 15¢. A single question on the task interface that we used for the evaluation of fluency is presented in Figure 3. Participants were considered for payment only when they answered the gold standard question correctly. To distinguish between participants who genuinely found it challenging to complete the work and those who were spamming, we provided feedback on the failed gold standard question and allowed people to review their answers two more times. This is in line with current discussions into the ethics of microtask crowdsourcing for scientific work (Le et al., 2010; Daniel et al., 2018). Per study, we collected ten judgements for each of the 32 graphs from the sample. The average rank of each system for a given set of triples is computed after we compute its mean relative position across the ten collected judgements. We subsequently average these values across all the 32 samples in order to obtain the mean fluency of each competing system.

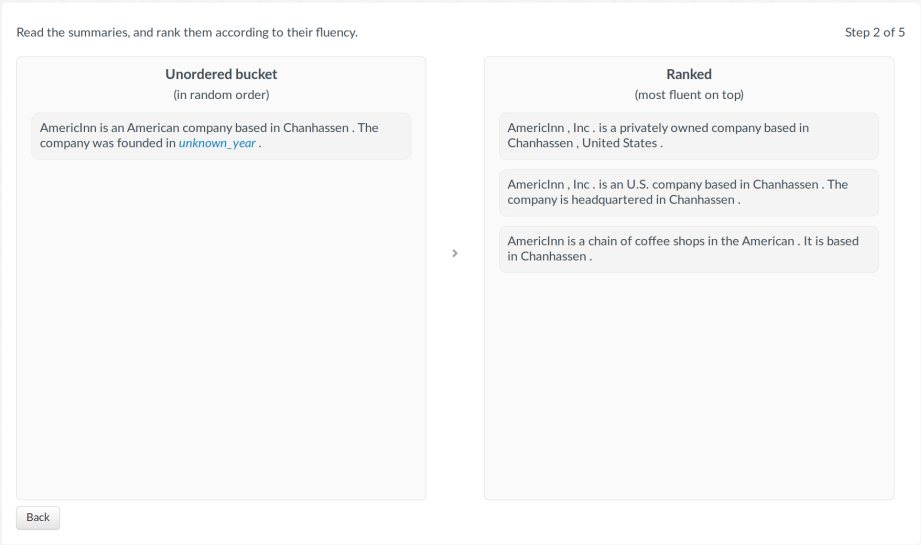
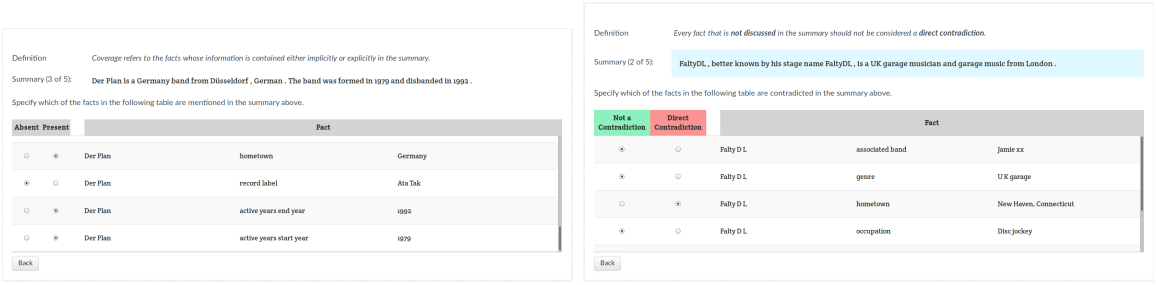


Figure 3: An example of question on the task interface that was designed for the evaluation of the fluency of the competing systems.

Coverage Participants saw a summary generated by one of the competing system alongside its corresponding input triples, and had to determine whether each triple in the set was either "Absent" or "Present" in the summary. Figure 4a displays a question on the task interface that we designed for the evaluation of coverage. Similarly to the fluency task, we presented each participant with five questions. We paid 15¢ per task. One of the questions was from the gold standard; participants who missed more than a triple in the gold standard question were able to review and re-submit two more times. If they failed after three attempts, they were excluded from the study. We allocated each pair of triples and text summaries to ten participants and used majority voting to decide whether a triple was covered by a summary or not. We then computed the total number of triples present and normalised by the total number of input triples to obtain the final score.



(a) Coverage (b) Contradiction

Figure 4: Examples of questions on the task interfaces that were designed for the evaluation of the coverage (a) and the contradiction (b) of the competing systems.

Contradiction The design of the contradiction task is very similar to the coverage one (see Figure 4b). Each participant had to answer five questions. For each question, the participants had to assess whether a triple from the input set would be a "Direct Contradiction" of the information from the accompanying text summary or not.

5.4.4 Results of the First Study

The results of the two studies are summarised in Table 8. The numbers are average scores for fluency, coverage and contradiction over the 32 summaries sampled for each study. The Wikipedia scores refer to the coverage and contradiction of the corresponding empirical summaries as they have been annotated by the authors. The results of the two studies are in alignment with the results of the automatic evaluation.

Fluency In the first study, which focused on the performance on the Full dataset, the summaries generated by both our systems were ranked significantly higher than those of the the two baselines (one-way ANOVA test, $p < .05$). Among our two systems, the one with surface form tuples was also found to generate significantly more fluent summaries than the one without. It should also be noted that summaries generated by the Pointer-Generator system were consistently ranked lower than the competition. This confirms our original hypothesis that realising entities from the triples with a single label does not lead to fluent text. It also emphasises the contribution of our realisation mechanism, which enables our pointer mechanism to "point" at the entity labels that are the most suitable for the generated text.

Coverage Both our systems realised more information from the input triples in the generated summaries; the coverage of both systems is also significantly better than Triples2GRU, our former work. Figure 5a shows the extent to which different number of predicates from the input triples are covered in the text. Notably our architectures are able to realise not only more predicates than the competition but also to address these predicates with higher consistency (i.e. greater coverage) in the corresponding summaries.

Contradiction All four systems scored low with respect to the amount of information in the text that is contradicted by the triples, and no statistically significant differences were

noted. In the vast majority of the cases the generated summaries were considered to be in alignment with the input set of triples. When any contradictions occurred, they were usually isolated to single triples. Out of the total of 12 summaries across all systems for which contradiction $> 0\%$ after the aggregation of annotations, only two of them had a second triple whose information was contradicted in the text; the other 10 had only a single triple marked as a potential contradiction. In a prior study, we have established a tendency of crowdworkers to overestimate contradictions (Vougiouklis et al., 2018b), which we believe is due to participants confusing missing and contradictory information. While the introduction of gold standard examples and splitting coverage and contradictions tasks helped, upon manual inspection, we found that there were still triples marked as contradicting to the summary simply because the corresponding information had not been captured in the text.

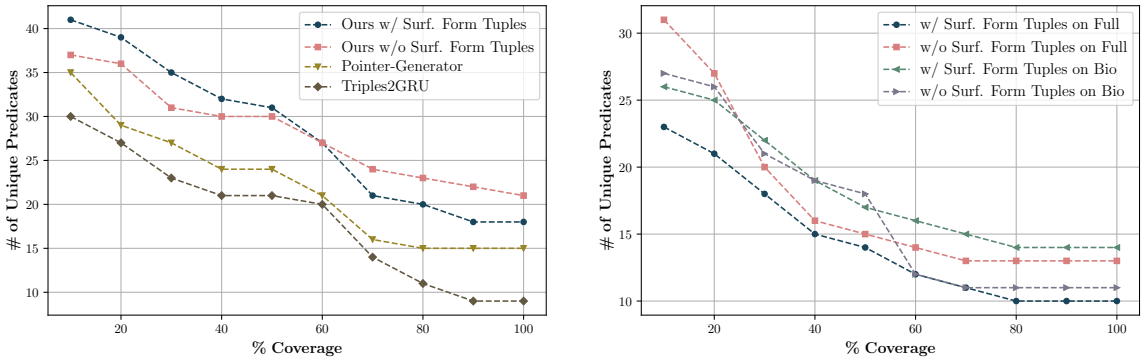
	Model	Fluency	Coverage	Contradiction
Full	Wikipedia	–	53.10	0.01
	Triples2GRU	$2.67^{(*)}$	$27.50^{(*)}$	0.35
	Pointer-Generator	$2.83^{(\dagger)}$	32.60	1.04
	Ours w/o Surf. Form Tuples	$2.38^{(*,\dagger,\ddagger)}$	$36.07^{(*)}$	1.14
	Ours w/ Surf. Form Tuples	$\mathbf{2.13}^{(*,\dagger,\ddagger)}$	$\mathbf{37.11}^{(*)}$	1.40
Biographies	Wikipedia	–	56.15	0.00
	w/o Surf. Form Tuples on Bio	2.47	40.51	1.09
	w/ Surf. Form Tuples on Bio	2.51	40.68	0.74
	w/o Surf. Form Tuples on Full	$2.71^{(\ddagger)}$	40.78	0.00
	w/ Surf. Form Tuples on Full	$\mathbf{2.31}^{(\ddagger)}$	37.53	0.95

Table 8: Average scores of the four investigated systems for the three criteria. The Wikipedia scores refer to the coverage and contradiction of the corresponding empirical summaries, annotated by the authors. Fluency shows the average relative position (out of four) in which summaries from a particular system are ranked. Coverage and contradiction are percentages; for the former, the higher the score the better whereas for the latter, the lower the score the better. $*$, † and ‡ denote statistical significance with $p < .05$ using pairwise one-way ANOVA of a system against Triples2GRU, Pointer-Generator and Ours w/o Surf. Form Tuples respectively. **Top:** Scores of the systems on the Full dataset. **Bottom:** Scores of the systems evaluated on the Biographies dataset. The "... on Bio" systems have been trained only on biographies.

5.4.5 Results of the Second Study

Training our systems on a much more challenging corpus results in minimal performance differences in comparison to the performance of the same systems when trained solely on

Biographies. While the average coverage (cf. Table 8 and Figure 5b) of the systems that have been trained on the Full corpus appears to be slightly lower than their respective fine-tuned versions (i.e. "... on Bio" systems), no statistically significant outcomes were observed. Nonetheless, when trained on the Full corpus and tested on biographies, the system w/ Surf. Form Tuples is significantly ($p < .05$) more fluent than w/o Surf. Form Tuples (cf. Table 8).



(a) Full

(b) Only on Biographies

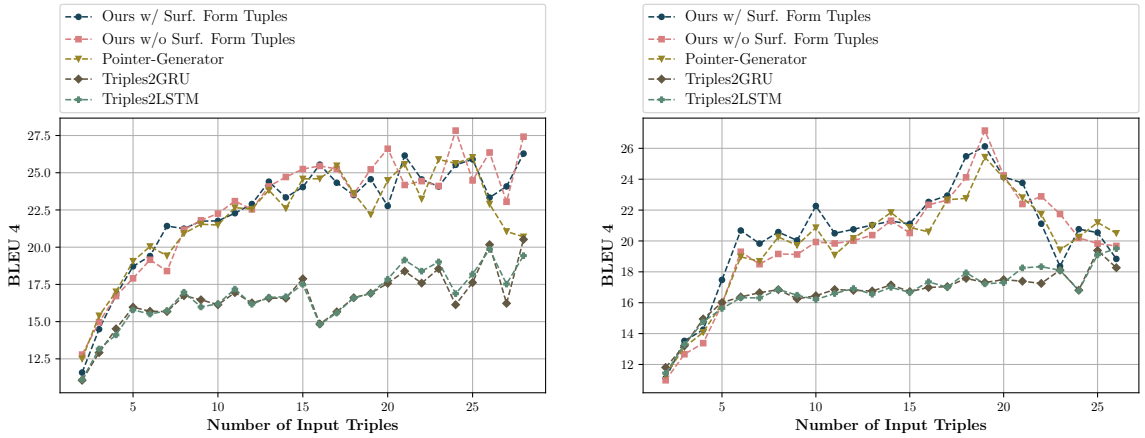
Figure 5: The percentage with which the unique predicates from the input triples are implicitly or explicitly covered in the generated summaries.

6. Discussion

Following the two evaluations, we note that while our pointer-generator network without the surface form tuples performed slightly better than the competition when trained and tested on a single domain, it is difficult for it to reproduce the same quality of summaries when trained on the Full dataset. Nonetheless, the inclusion of the surface form tuples makes the model more flexible at addressing larger vocabulary sizes without paying a high cost on single domains.

In the training portion of the Full dataset, the entities are realised with their first label 86% of the times, with their second label 11.5% of the times, and with their third label 2.5% of the times. During testing, our system realises entities from the triples using the first, second, and third realisation with respective percentages of 85, 11, and 4. We believe that this, along with the higher average fluency achieved by our summaries confirms the added value of our verbalisation approach.

In addition to the above experiments, we grouped Wikipedia summaries that are allocated to the same number of input triples and computed a BLEU score per group. Figure 6 shows the performance of our models with the BLEU 4 metric on the 97th percentile of the Biographies and Full test sets, across different numbers of input triples. We note that the sets of triples that consist of more than 26 triples for Biographies, and 21 triples for the Full corpus are inputted to the systems after they are stripped of their additional triples, according to the methodology described in Section 4. We noticed that when the systems are initialised with a low number of triples, they lack the information required to form a



(a) Biographies (b) Full

Figure 6: Performance of our models with the BLEU 4 metric across the different sizes of triple sets from the test set of the Biographies (a) and the Full (b) dataset. Please note that sets of triples that consist of more than 26 and 21 triples in the case of the Biographies and the Full corpus respectively are inputted to the systems after they are stripped of their additional triples according to the methodology described in Section 4.

two-sentence summary. On the biographies task, the performance of the pointer-generator networks (i.e. Pointer-Generator, Ours w/ Surf. Form Tuples and Ours w/o Surf. Form Tuples) progressively increases as more triples are fed into the system. This shows the ability of the model to successfully "select" the relevant triples and address them in the generated summary. The Full dataset is much more challenging due to the size of its source and target dictionaries (cf. Table 2). In addition, only a low number of triples' entities can be on average identified in the non-biographical articles ($\sim$ 1.8 entities per summary). The latter is of great importance since the ability to directly copy information from the triples to the text is essentially what separates the architectures that leverage the pointer mechanism from the other neural architectures (i.e. Triples2GRU and Triples2LSTM). In a scenario in which no information in the text is directly taken from the triples, we would expect all systems to score almost identical performance[13]. However, even in this scenario, the systems based on the pointer mechanism consistently outperform the competition.

We also observe a drop in performance when our systems were provided with oversized triple sets. This is more noticeable on the Full corpus, and is mainly because of the upper bounds with respect to the number of allocated triples that we set per summary (Eq. 13). Based on these upper bounds, we applied a simple approach to eliminate redundant triples (Vougiouklis et al., 2018a). When their number still exceeded the threshold, during training, we used an $E_{\max}$ number of them by prioritising triples whose subjects or objects have been mentioned in the text. However, since the number of tokens in the summaries that were recognised as realisations of entities or years was relatively low, it

13. In theory, the architectures that are based on pointer-generator networks should still have some advantage due their attention mechanism.

is likely the $E_{\max}$ triples that we retained from a very large set would not be reflected in the text summary. The result of this misalignment of triples and summaries might not be noticeable in the case of biographies due to their regular structure, but its effect is amplified in the context of an open-domain corpus.

We see the generation of multi-sentence summaries (i.e. very long sequences) given very large input sets of triples as a natural extension of this work. Based on our above findings, we identify the following challenges with respect to this direction: (i) our encoder should be provided with all the information it needs in order for the end-system to meet the challenging generative expectations, and (ii) the decoder should be able to retain the information from the input at very distant timesteps.

A valid approach for overcoming the former challenge would be to allow the model to "select" the most appropriate triples from an oversized set. Selection of triples is already actively performed by the proposed systems, but only on the basis of the $E_{\max}$ parameter. Our pointer-generator networks generate a summary by attending the most relevant parts of the input at each decoding timestep. Consequently, triples which are more relevant to the task are rewarded with higher probabilities. An iterative process that is worth investigating is to identify the properties of the triples that are attended the most by a trained system, and retrain the system by prioritising triples whose predicates have been attended the most during testing. The process would eventually stop when no further improvement in the automatic evaluation metrics could be observed.

The second challenge gives ground to repetition, which is an additional problem associated with the generation of much longer snippets of text using attentive adaptations of the general encoder-decoder framework. While such behaviour was not commonly observed in our experiments, it might prove to be one of the challenges in a multi-sentence generation scenario. This problem had been recently addressed with the implementation of a coverage architecture on top of the attention mechanism (Tu et al., 2016; Mi et al., 2016). *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 subsequent timesteps. The existing attention mechanism with which our pointer-generator network is equipped allows us to explore to what extent monitoring the coverage of the generated text is required in a triples-to-multi-sentence-summaries scenario.

7. Conclusion

We presented a data-driven approach to generate open-domain text summaries from knowledge base triples. We proposed a pointer-generator network that jointly learns to verbalise in a different number of ways the content from the triples, while retaining the ability to generate regular words from a fixed target vocabulary. We trained and evaluated two system variants on two different datasets of aligned DBpedia triples with Wikipedia summaries with respective vocabulary sizes of 400k and 1114k words.

We evaluated our approach using well-established automatic text similarity metrics and conducted two user studies to determine how fluent the summaries are, how well they match the input, both in terms of coverage of the information conveyed and unintended contradictions. Both the automatic and the user evaluation show promising results. Our approach outperforms the state of the art; in particular, compared to other encoder-decoder

architectures, our summaries are significantly more fluent and convey a greater share of the content of the input triples.

Acknowledgements

We thank the reviewers for their thorough and insightful feedback. This research is partially supported by the Data Stories and ACTION projects. The former is funded by EPSRC research grant No. EP/P025676/1, and the latter by the European Union's H2020 research and innovation programme under grant agreement number 824603.

References

Angeli, G., Liang, P., and Klein, D. (2010). A simple domain-independent probabilistic approach to generation. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, EMNLP '10, pages 502–512, Stroudsburg, PA, USA. Association for Computational Linguistics.

Bahdanau, D., Cho, K., and Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473.

Bouayad-Agha, N., Casamayor, G., Mille, S., and Wanner, L. (2012). Perspective-oriented generation of football match summaries: Old tasks, new challenges. *ACM Trans. Speech Lang. Process.*, 9(2):3:1–3:31.

Bouayad-Agha, N., Casamayor, G., and Wanner, L. (2014). Natural language generation in the context of the semantic web. *Semantic Web*, 5(6):493–513.

Celikyilmaz, A., Hakkani-Tur, D., Pasupat, P., and Sarikaya, R. (2015). Enriching word embeddings using knowledge graph for semantic tagging in conversational dialog systems.

Chen, D. L., Kim, J., and Mooney, R. J. (2010). Training a multilingual sportscaster: Using perceptual context to learn language. *J. Artif. Int. Res.*, 37:397–435.

Chen, D. L. and Mooney, R. J. (2008). Learning to sportscast: A test of grounded language acquisition. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, pages 128–135, New York, NY, USA. ACM.

Chisholm, A., Radford, W., and Hachey, B. (2017). 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*, pages 633–642, Valencia, Spain. Association for Computational Linguistics.

Cho, K., van Merrienboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014). 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)*, pages 1724–1734, Doha, Qatar. Association for Computational Linguistics.

Daiber, J., Jakob, M., Hokamp, C., and Mendes, P. N. (2013). Improving efficiency and accuracy in multilingual entity extraction. In *Proceedings of the 9th International Conference on Semantic Systems*, I-SEMANTICS '13, pages 121–124, New York, NY, USA. ACM.

Daniel, F., Kucherbaev, P., Cappiello, C., Benatallah, B., and Allahbakhsh, M. (2018). Quality control in crowdsourcing: A survey of quality attributes, assessment techniques, and assurance actions. *ACM Comput. Surv.*, 51(1):7:1–7:40.

Dannélls, D., Damova, M., Enache, R., and Chechev, M. (2012). Multilingual online generation from semantic web ontologies. In *Proceedings of the 21st International Conference on World Wide Web*, WWW '12 Companion, pages 239–242, New York, NY, USA. ACM.

Dong, L. and Lapata, M. (2016). Language to logical form with neural attention. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 33–43, Berlin, Germany. Association for Computational Linguistics.

Ell, B. and Harth, A. (2014). A language-independent method for the extraction of RDF verbalization templates. In *Proceedings of the 8th International Natural Language Generation Conference (INLG)*, pages 26–34, Philadelphia, Pennsylvania, U.S.A. Association for Computational Linguistics.

Gardent, C., Shimorina, A., Narayan, S., and Perez-Beltrachini, L. (2017). The WebNLG challenge: Generating text from RDF data. In *Proceedings of the 10th International Conference on Natural Language Generation*, pages 124–133. Association for Computational Linguistics.

Gehrmann, S., Dai, F., Elder, H., and Rush, A. (2018). End-to-end content and plan selection for data-to-text generation. In *Proceedings of the 11th International Conference on Natural Language Generation*, pages 46–56, Tilburg University, The Netherlands. Association for Computational Linguistics.

Green, N. (2006). Generation of biomedical arguments for lay readers. In *Proceedings of the Fourth International Natural Language Generation Conference*, INLG '06, pages 114–121, Stroudsburg, PA, USA. Association for Computational Linguistics.

Gu, J., Lu, Z., Li, H., and Li, V. O. (2016). Incorporating copying mechanism in sequence-to-sequence learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1631–1640, Berlin, Germany. Association for Computational Linguistics.

Halko, N., Martinsson, P.-G., and Tropp, J. A. (2011). Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM Rev.*, 53(2):217–288.

Heafield, K., Pouzyrevsky, I., Clark, J. H., and Koehn, P. (2013). Scalable modified Kneser-Ney language model estimation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 690–696, Sofia, Bulgaria. Association for Computational Linguistics.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural Comput.*, 9(8):1735–1780.

Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In Bach, F. and Blei, D., editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 448–456, Lille, France. PMLR.

Jing, H. (2002). Using hidden markov modeling to decompose human-written summaries. *Computational Linguistics*, 28(4):527–543.

Kartsaklis, D., Pilehvar, M. T., and Collier, N. (2018). Mapping text to knowledge graph entities using multi-sense LSTMs. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1959–1970, Brussels, Belgium. Association for Computational Linguistics.

Kim, J. and Mooney, R. J. (2010). Generative alignment and semantic parsing for learning from ambiguous supervision. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*, COLING '10, pages 543–551, Stroudsburg, PA, USA. Association for Computational Linguistics.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Konstas, I. and Lapata, M. (2012a). Concept-to-text generation via discriminative reranking. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers - Volume 1*, ACL '12, pages 369–378, Stroudsburg, PA, USA. Association for Computational Linguistics.

Konstas, I. and Lapata, M. (2012b). Unsupervised concept-to-text generation with hypergraphs. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 752–761, Montréal, Canada. Association for Computational Linguistics.

Konstas, I. and Lapata, M. (2013). A global model for concept-to-text generation. *J. Artif. Int. Res.*, 48(1):305–346.

Lavie, A. and Agarwal, A. (2007). METEOR: An automatic metric for mt evaluation with high levels of correlation with human judgments. In *Proceedings of the Second Workshop on Statistical Machine Translation*, StatMT '07, pages 228–231, Stroudsburg, PA, USA. Association for Computational Linguistics.

Le, J., Edmonds, A., Hester, V., and Biewald, L. (2010). Ensuring quality in crowdsourced search relevanceevaluation: The effects of training question distribution. In *ACM SIGIR 2010 Workshop on Crowdsourcing for Search Evaluation (CSE 2010)*, pages 17–20.

Lebret, R., Grangier, D., and Auli, M. (2016). Neural text generation from structured data with application to the biography domain. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1203–1213. Association for Computational Linguistics.

Lehmann, J., Isele, R., Jakob, M., Jentzsch, A., Kontokostas, D., Mendes, P. N., Hellmann, S., Morsey, M., van Kleef, P., Auer, S., and Bizer, C. (2015). DBpedia - A large-scale, multilingual knowledge base extracted from wikipedia. *Semantic Web*, 6(2):167–195.

Li, H., Xiong, C., and Callan, J. (2017). Natural language supported relation matching for question answering with knowledge graphs. In *Proceedings of the First Workshop on Knowledge Graphs and Semantics for Text Retrieval and Analysis (KG4IR 2017)*.

Li, L. and Wan, X. (2018). Point precisely: Towards ensuring the precision of data in generated texts using delayed copy mechanism. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1044–1055, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Liang, P., Jordan, M. I., and Klein, D. (2009). 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, pages 91–99, Stroudsburg, PA, USA. Association for Computational Linguistics.

Lin, C.-Y. (2004). ROUGE: A package for automatic evaluation of summaries. In Marie-Francine Moens, S. S., editor, *Text Summarization Branches Out: Proceedings of the ACL-04 Workshop*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Liu, T., Wang, K., Sha, L., Chang, B., and Sui, Z. (2018). Table-to-text generation by structure-aware seq2seq learning.

Lu, W., Ng, H. T., Lee, W. S., and Zettlemoyer, L. S. (2008). A generative model for parsing natural language to meaning representations. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 783–792, Honolulu, Hawaii. Association for Computational Linguistics.

Luong, T., Sutskever, I., Le, Q., Vinyals, O., and Zaremba, W. (2015). 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)*, pages 11–19, Beijing, China. Association for Computational Linguistics.

Ma, Y., Crook, P. A., Sarikaya, R., and Fosler-Lussier, E. (2015). Knowledge graph inference for spoken dialog systems. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5346–5350.

Mei, H., Bansal, M., and Walter, M. R. (2016). What to talk about and how? selective generation using LSTMs with coarse-to-fine alignment. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 720–730, San Diego, California. Association for Computational Linguistics.

Mi, H., Sankaran, B., Wang, Z., and Ittycheriah, A. (2016). Coverage embedding models for neural machine translation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 955–960, Austin, Texas. Association for Computational Linguistics.

Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. (2002). BLEU: A method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, pages 311–318, Stroudsburg, PA, USA. Association for Computational Linguistics.

Reiter, E. (2010). *Natural Language Generation*, chapter 20, pages 574–598. Wiley-Blackwell.

Reiter, E. and Dale, R. (2000). *Building Natural Language Generation Systems*. Cambridge University Press, New York, NY, USA.

Reiter, E., Sripada, S., Hunter, J., Yu, J., and Davy, I. (2005). Choosing words in computer-generated weather forecasts. *Artif. Intell.*, 167(1-2):137–169.

Rush, A. M., Chopra, S., and Weston, J. (2015). A neural attention model for abstractive sentence summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 379–389, Lisbon, Portugal. Association for Computational Linguistics.

See, A., Liu, P. J., and Manning, C. D. (2017). 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)*, pages 1073–1083. Association for Computational Linguistics.

Sleimi, A. and Gardent, C. (2016). Generating paraphrases from DBpedia using deep learning. In *Proceedings of the 2nd International Workshop on Natural Language Generation and the Semantic Web (WebNLG 2016)*, pages 54–57. Association for Computational Linguistics.

Sutskever, I., Vinyals, O., and Le, Q. V. (2014). Sequence to sequence learning with neural networks. In Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N. D., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems 27*, pages 3104–3112. Curran Associates, Inc.

Tu, Z., Lu, Z., Liu, Y., Liu, X., and Li, H. (2016). Modeling coverage for neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 76–85, Berlin, Germany. Association for Computational Linguistics.

Turner, R., Sripada, Y., and Reiter, E. (2009). Generating approximate geographic descriptions. In *Proceedings of the 12th European Workshop on Natural Language Generation*, ENLG '09, pages 42–49, Stroudsburg, PA, USA. Association for Computational Linguistics.

Vinyals, O., Fortunato, M., and Jaitly, N. (2015). Pointer networks. In Cortes, C., Lawrence, N. D., Lee, D. D., Sugiyama, M., and Garnett, R., editors, *Advances in Neural Information Processing Systems 28*, pages 2692–2700. Curran Associates, Inc.

Vougiouklis, P., Elsahar, H., Kaffee, L.-A., Gravier, C., Laforest, F., Hare, J., and Simperl, E. (2018a). Neural wikipedian: Generating textual summaries from knowledge base triples. *Journal of Web Semantics*.

Vougiouklis, P., Maddalena, E., Hare, J., and Simperl, E. (2018b). How biased is your nlg evaluation? In *Proceedings of the 1st International Workshop on CrowdBias (CrowdBias 2018)*.

Wiseman, S., Shieber, S., and Rush, A. (2017). Challenges in data-to-document generation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2253–2263, Copenhagen, Denmark. Association for Computational Linguistics.

Yeh, S., Huang, H., and Chen, H. (2018). Precise description generation for knowledge base entities with local pointer network. In *2018 IEEE/WIC/ACM International Conference on Web Intelligence (WI)*, pages 214–221.