UNIVERSITY OF SOUTHAMPTON

Neural Generation of Textual Summaries from Knowledge Base Triples

by

Pavlos Vougiouklis

A thesis submitted in partial fulfillment for the degree of Doctor of Philosophy

in the

Faculty of Engineering and Physical Sciences

School of Electronics and Computer Science

January 2019

Abstract

University of Southampton
Faculty of Engineering and Physical Sciences
School of Electronics and Computer Science

A thesis submitted in partial fulfilment for the degree of Doctor of Philosophy

by Pavlos Vougiouklis

Most people need textual or visual interfaces in order to make sense of Semantic Web data. In this thesis, we investigate the problem of generating natural language summaries for structured data encoded as triples using neural networks.

We propose an end-to-end trainable architecture that encodes the information from a set of triples into a vector of fixed dimensionality and generates a textual summary by conditioning the output on this encoded vector. In order to both train and evaluate the performance of our approach, we explore different methodologies for building the required data-to-text corpora. We initially focus our attention on the generation of biographies. Using methods for both automatic and human evaluation, we demonstrated that our technique is capable of scaling to domains with challenging vocabulary sizes of over 400k words.

Given the promising results of our approach in biographies, we explore its applicability in the generation of open-domain Wikipedia summaries in two under-resourced languages, Arabic and Esperanto. We propose an adaptation of our original encoder-decoder architecture that outperforms a set of strong baselines of different nature. Furthermore, we conducted a set of community studies in order to measure the usability of the generated content by Wikipedia readers and editors. The targeted communities ranked our generated text close to the expected standards of Wikipedia. In addition, we found that the editors are likely to reuse a large portion of the generated summaries, thus, emphasizing the usefulness of our approach to the involved communities.

Finally, we extend the original model with a pointer mechanism that enables it to jointly learn 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 evaluate performance with a dataset encompassing the entirety of English Wikipedia. Results from both automatic and human evaluation highlight the superiority of the latter approach compared to our original encoder-decoder architecture and a set of competitive baselines.

Contents

List of Figures

List of Tables

Nomenclature

BLEU	Bilingual Evaluation Understudy (Papineni et al., 2002) (see Section 3.1.1)
BPTT	BackPropagation Through Time
D1	corpus of DBpedia triples aligned with English Wikipedia biographies
D2	corpus of Wikidata triples aligned with English Wikipedia biographies
D3	corpus of DBpedia triples aligned with English Wikipedia summaries, encompassing the entirety of Wikipedia
DB1	Mapping-based Objects DBpedia dataset (http://wiki.dbpedia.org/downloads-2016-10)
DB2	Mapping-based Literals DBpedia dataset (http://wiki.dbpedia.org/downloads-2016-10)
DB3	Instance Types DBpedia dataset (http://wiki.dbpedia.org/downloads-2016-10)
DB4	Genders DBpedia dataset (http://wiki.dbpedia.org/downloads-2016-10)
DB_{ar}	Arabic Long Abstract DBpedia dataset (http://wiki.dbpedia.org/downloads-2016-10)
DB_{en}	English Long Abstract DBpedia dataset (http://wiki.dbpedia.org/downloads-2016-10)
DB_{eo}	Esperanto Long Abstract DBpedia dataset (http://wiki.dbpedia.org/downloads-2016-10)
GPU	Graphics Processing Unit
GRU	Gated Recurrent Unit
GST	Greedy String-Tiling
HMM	Hidden Markov Model
k-NN	k-Nearest Neighbours algorithm
LSTM	Long Short-Term Memory

M1	corpus of Wikidata triples aligned with Arabic Wikipedia summaries
M2	corpus of Wikidata triples aligned with Esperanto Wikipedia summaries
NLG	Natural Language Generation
NLP	Natural Language Processing
One-Hot Vector	refers to a one-hot vector representation token, x_t, of a dictionary of, usually, characters or words. A one-hot vector contains a 1 at the index of this particular x_t token in the dictionary, and zero elsewhere.
ROUGE	Recall-Oriented Understudy for Gisting Evaluation (Lin, 2004) (see Section 3.1.1)
RDF	Resource Description Framework
RLM	Recurrent Language Model
RNN	Recurrent Neural Network
S3	Semantic Sentence Simplification (Mrabet et al., 2016)
WD1	Wikidata truthy dumps (https://dumps.wikimedia.org/wikidatawiki/entities)

$[\ldots;\ldots]$	vector concatenation.
$\lceil\ldots\rceil$	the ceiling function.
$\lfloor\ldots\rfloor$	the floor function.
$j\%i$	the remainder of $j \in \mathbb{R}$ when divided by $i \in \mathbb{R}$.
E	the number of input triples with which a system is provided.
E_{max}	the maximum number of input triples with which a system is provided.
f_i	an RDF triple (i.e. $f_i = (s_i, p_i, o_i)$, where s_i, p_i and o_i are the respective subject, predicate and objective of this i-th triple).
$h_t^l \in \mathbb{R}^n$	the aggregated output of a hidden unit at timestep $t \in [1 \ldots T]$ and layer depth $l \in [1 \ldots L]$.
$\text{ReLU}(z)$	non-linear activation function that applies the rectified linear unit function, $\text{ReLU}(z) = \max(0, z)$.
$\text{sigm}(z)$	the logistic sigmoid function, $\text{sigm}(z) = \dfrac{1}{1+\exp(-z)}$.
$\text{softmax}(y_t)$	the softmax activation function provides the probability distribution over the next token given a vector $y_t \in \mathbb{R}^V$, where V is the dimensionality of the target vocabulary. In word-level language modelling, it is used to predict the probability of the next word x_t given the previous history $x_1, \ldots, x_{t-1}$ (i.e. $p(x_t \mid x_1, \ldots, x_{t-1}) = \dfrac{\exp(y_t^{(x_t)})}{\sum_{v=1}^{V} \exp(y_t^{(v)})}$, where V is the size of the dictionary of words).

$\tanh(z)$	the hyperbolic tangent function, $\tanh(z) = \dfrac{\exp(z) - \exp(-z)}{\exp(z) + \exp(-z)}$.
$\mathbf{W_{x \to y}}$	the weights of the connections between the inputs x and the outputs y of an architecture.
$\lvert X \rvert$	the size of the fixed dictionary of all the available words and entities in the textual summaries dictionary.

Acknowledgements

I would like to express my sincere gratitude to both my supervisors, Prof Elena Simperl and Dr Jonathon Hare. Without their valuable guidance and feedback, none of the featured work would have been possible. I am also grateful to the Marie Skłodowska-Curie Innovative Training Networks, part of the Horizon 2020 programme, for providing me with the financial support for my research studies under the Answering Questions using Web Data (WDAqua) project.

I would also like to thank my lab mates; in particular Alexandry for our endless discussions about Machine Learning and Linux, Lucie for a memorable collaboration, and for urging me to explore multilingual text generation, and Valerio for all the Foosball games and drinks we shared. Outside of the lab, I would particularly like to thank my long-standing friends Andreas, Dimitrios, Ioannis and Panagiotis.

Last but not least, I would like to address a special thank you to my girlfriend Marianna for all her love and support through my abnormal work schedule.

This thesis is dedicated to my parents, Fotios and Evangelia.

Chapter 1

Introduction

Humans desire to store an exponentially increasing amount of data in a format that is easily processable by machines has led to the adoption of various database frameworks. The requirement to enable machines to understand complicated relations between the objects of a structured data source has led to the development of knowledge bases and knowledge graphs. The Semantic Web along with its surrounding technologies, such as the Web Ontology Language (OWL) and the Resource Description Framework (RDF) have emerged as a result of the latter (Shadbolt et al., 2006). While Semantic Web data, such as triples in RDF, is easily accessible by machines, it is difficult to be understood by humans. This is especially true for people who are unfamiliar with the underlying technologies. On the contrary, for humans, reading well-structured and informative text is not only a much more accessible, but also appealing, activity.

Natural Language Generation (NLG) is concerned with the development of the textual interfaces that generate text that describes the input records of a structured data source in a fluent and sensible manner (Reiter and Dale, 2000). In the context of the Semantic Web, the structured data is in the form of knowledge base triples. This thesis focuses on the development of those NLG techniques that would make the information that is stored in a knowledge base's triples more accessible to potential human users.

Most NLG systems that are employed over knowledge graphs, such as Wikidata, are responsible for generating a textual summary of an input sub-graph of triples (Bouayad-Agha et al., 2014). As such, their further development can have a dramatic impact on a variety of application domains. A typical application is their integration in Question Answering platforms, where a user's experience could be improved by the ability to automatically generate a textual description of an entity that is returned as a response to their query. The Google Knowledge Graph[1] and the Wikidata Reasonator[2] are some

[1] https://googleblog.blogspot.co.uk
[2] https://tools.wmflabs.org/reasonator

representative examples of such Question Answering systems. In the case of the former, the returned text is copied directly from a single source, and, thus, not generated. The latter is a rule-based system of community-curated templates. In its current state the system supports realisation of a limited type of properties and entity types in only English, French and German.

Another domain of application 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 (Janzen and Maass, 2009). Finally, the ability to generate coherent text that addresses a set structured records can significantly improve the coverage of Wikipedia or other collaborative knowledge bases, in which many topics, especially in the less popular languages, remain under-represented (Chisholm et al., 2017).

1.1 Aims and Objectives

The aim of the work in this thesis was to investigate how multilingual, open-domain textual summaries could be generated from knowledge base triples. Recent literature suggests that neural networks outperform rule- and k-NN-based approaches. However, they have either only been employed on single-domain datasets (Mei et al., 2016), or by restricting the generation procedure to a single sentence (Lebret et al., 2016; Chisholm et al., 2017). Furthermore, the application of neural networks on top of Semantic Web triples is still a relatively unexplored domain. To this end, the first objective was to develop a cross-lingual approach of building corpora of knowledge base triples aligned with texts. The subsequent objective was the design and implementation of a system capable of generating a textual summary of a given input sub-graph of triples.

With this objective complete, the work has evolved to investigate the actual usability of the generated content and the performance of more advanced architectures for textual summaries generation. For the first, we chose to address the communities of Wikipedias that lack content in order to evaluate the usability of the generated summaries by their corresponding readers and editors. For the latter, we enhanced the original model with a *pointer* mechanism that allows it to verbalise rare entities and numbers in the triples by simply *selecting* them from the input. The results are evaluated on both single and open-domain Wikipedia summaries generation.

1.2 Contributions

This thesis brings a number of clear contributions which are highlighted in brief below:

- A fully-automatic, cross-lingual, approach for building loosely aligned corpora of knowledge base triples with Wikipedia summaries
- An end-to-end trainable system that can generate a textual summary from knowledge base triples
- A set of different approaches that enable the verbalisation of frequent and infrequent entities in the generated text
- An adaptation of the above architecture for the generation of summaries in under-resourced languages
- An evaluation through a set of community studies that measures the usefulness of the above end-system to the needs of the underserved Wikipedias (equal contribution with two other PhD students from the WDAqua project, Lucie-Aimée Kaffee and Hady Elsahar)
- An end-to-end trainable neural architecture that jointly learns to verbalise in a different number of ways the content from triples, while retaining the ability to generate regular words from a fixed target vocabulary

The work has lead to a series of peer-reviewed publications. It should be noted that during the first year of his PhD, the author of this thesis familiarised himself with the fundamental neural network architectures by working on the domain of automatic response generation on social media. This work led to a refereed conference publication (Vougiouklis et al., 2016). The paper proposed an end-to-end learnable dialogue system which rather than generating a response explicitly based on the sequence of the most recent utterances of a conversation thread, it incorporates background knowledge in order to capture the context of a conversation better. However, this work is not described in the subsequent chapters in favour of the contextual consistency of the thesis. Table 1.1 presents the list of the resultant publications to which the work of this thesis is most related.

1.3 Thesis Structure

This thesis describes the work of the author in attempting to achieve the objectives outlined earlier in this chapter. Chapter 2 documents existing approaches for NLG, from rule-based, to statistical ones. It also provides a detailed description of some of the most recent neural network architectures from which some of the contributions of the subsequent chapters are inspired. Chapter 3 discusses the different methodologies that have been explored in the literature for the evaluation of NLG systems, and describes in detail the methods and criteria against which the systems' that are proposed in this thesis

TABLE 1.1: The main contributions of this work.

Main Contributions

Pavlos Vougiouklis†, Eddy Maddalena†, Jonathon Hare and Elena Simperl. How Biased Is Your NLG Evaluation?. In *Proceedings of the 1st Workshop on CrowdBias*, CrowdBias 2018, Zurich, Switzerland.

Pavlos Vougiouklis, Hady Elsahar, Lucie-Aimée Kaffee, Christophe Gravier, Frédérique Laforest, Jonathon Hare, and Elena Simperl. Neural Wikipedian: Generating Textual Summaries from Knowledge Base Triples. *Journal of Web Semantics*, 2018.

Lucie-Aimée Kaffee†, Hady Elsahar†, **Pavlos Vougiouklis**†, Christophe Gravier, Frédérique Laforest, Jonathon Hare, and Elena Simperl. Mind the (Language) Gap: Generation of Multilingual Wikipedia Summaries from Wikidata for ArticlePlaceholders. In *Proceedings of 15th International Conference, ESWC 2018*, Heraklion, Crete, Greece. Springer International Publishing.

Lucie-Aimée Kaffee†, Hady Elsahar†, **Pavlos Vougiouklis**†, Christophe Gravier, Frédérique Laforest, Jonathon Hare, and Elena Simperl. Learning to Generate Wikipedia Summaries for Underserved Languages from Wikidata. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, New Orleans, Louisiana. Association for Computational Linguistics.

Hady Elsahar, **Pavlos Vougiouklis**, Arslen Remaci, Christophe Gravier, Jonathon Hare, Elena Simperl, and Frédérique Laforest. T-REx: A Large Scale Alignment of Natural Language with Knowledge Base Triples. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation LREC 2018*, Miyazaki, Japan. Association for Computational Linguistics.

Pavlos Vougiouklis, Jonathon Hare, and Elena Simperl. A Neural Network Approach for Knowledge-Driven Response Generation. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics*, Osaka, Japan. Association for Computational Linguistics.

Yassine Mrabet, **Pavlos Vougiouklis**, Halil Kilicoglu, Claire Gardent, Dina Demner-Fushman, Jonathon Hare, and Elena Simperl. 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 '16, Edinburgh, Scotland. Association for Computational Linguistics.

†The authors contributed equally to this work.

are evaluated. Chapter 4 presents a fully-automatic approach for building large data-to-text corpora along with descriptions of the resultant corpora used in the subsequent chapters. Chapter 5 details a novel neural network architecture based on an encoder-decoder framework that generates textual summaries from knowledge base triples. The proposed approach is evaluated on the generation of biographies in English. In Chapter 6 the above approach is expanded in order to generate open-domain textual summaries in two under-resourced Wikipedia languages: (i) Esperanto and (ii) Arabic. Alongside, the regular automatic evaluation metrics, the generated summaries are evaluated through two different community studies that aim to explore the usability of the automatically

generated content by those underserved Wikipedia communities. Chapter 7 extends the encoder-decoder architecture of Chapter 5 with a pointer mechanism that enables the system to jointly learn to verbalise content from triples while retaining an ability to generate regular words from a fixed target vocabulary. The approach is evaluated on both single and open-domain Wikipedia summary generation. Finally, Chapter 8 summarises the results and contributions of this thesis, and highlights potential future directions with respect to all of the previous contribution chapters.

Chapter 2

Background

This chapter documents some of the most notable existing systems for Natural Language Generation. A range of approaches are discussed, from rule-based ones that had mostly focused on the verbalisation from domain ontologies using hand-coded linguistic features to the most recent, data-driven ones that attempt to "learn" directly from data how to perform the different phases of the NLG pipeline. In the latter sections, the chapter provides a detailed description of some of the most recent neural network architectures which have inspired the contributions of the subsequent chapters of this thesis.

2.1 Natural Language Generation

NLG systems typically work in three different stages: (i) document planning or content selection, (ii) microplanning, and (iii) surface realisation Bouayad-Agha et al. (2014); Reiter and Dale (2000). 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 (Reiter et al., 2005; Green, 2006; Galanis and Androutsopoulos, 2007; Turner et al., 2009).

Data-driven approaches have been proposed that attempt to learn individual modules for content selection, microplanning, and surface realisation. Duboue and McKeown and Barzilay and Lapata treat the problem of learning content selection as a classification task. They both propose different systems that learn from a corpus of aligned sentences

and records whether a database entry should be selected to verbalised or not. Liang et al. used a hierarchical Hidden semi-Markov model to generate textual descriptions of football matches and weather forecasts. Their proposed system worked by concatenating word sequences each one of which has been conditioned on a selected predicate from a predicted sequence of predicates that has been chosen based on an initial selection of record types. The resultant model is learned in an unsupervised manner using Expectation Maximisation (EM). Microplanning (or sentence planning) has been modelled as a supervised set partitioning task over records from American football matches where each partition corresponds to a sentence (Barzilay and Lapata, 2006).

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

Most of the previous work on NLG with Semantic Web data has focused on the verbalisation from domain ontologies using hand-coded rules. However, designing linguistic features requires significant effort in order for all the aspects of a specific language to be successfully captured. These systems work in domains with small vocabularies and restricted linguistic variability. Examples include systems that generate clinical narratives (Arguello et al., 2011), summaries of football matches (Bouayad-Agha et al., 2012), and, descriptions of museums' exhibits (Dannélls et al., 2012). Further Semantic Web-oriented NLG applications can be found in (Bouayad-Agha et al., 2014). The difficulty of transferring the involved rules across different domains or languages along with the

tedious repetition of their textual patterns has prevented them from becoming widely accepted (Socher and Manning, 2013; Bouayad-Agha et al., 2014).

Our work naturally lies on the path opened by recent unsupervised (Duma and Klein, 2013) and *distant-supervision* (Ell and Harth, 2014) 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, the models that are proposed in this thesis jointly perform content selection and surface realisation, without the inclusion of any hand-engineered rules or templates.

The recent success of neural networks in various text generative tasks, ranging from Machine Translation (Cho et al., 2014; Sutskever et al., 2014; Bahdanau et al., 2014) and automatic response generation (Vinyals et al., 2015a; Wen et al., 2015, 2016; Vougiouklis et al., 2016) to text summarisation (Rush et al., 2015; See et al., 2017), and generation of textual descriptions from visual data (Karpathy and Fei-Fei, 2017; Vinyals and Le, 2015) has fuelled NLG-oriented research that adapts models from the above domains to the requirements of the NLG tasks (Mei et al., 2016; Lebret et al., 2016; Chisholm et al., 2017). These systems have shown substantial improvement over other competitive data-driven approaches (Mei et al., 2016), and have proved to scale on challenging vocabulary sizes of over than 400k words (Lebret et al., 2016; Chisholm et al., 2017). More recently, adaptation of out-of-the-box Neural Machine Translation models, based on the general encoder-decoder framework (discussed in detail in Section 2.2.3), have shown great potential in tackling various aspects of triples-to-text tasks ranging from microplanning (Gardent et al., 2017b) to generation of paraphrases (Sleimi and Gardent, 2016). These systems work by conditioning a language model to a set of structured records.

In the following section we describe some of the most fundamental architectures that are used both by the aforementioned works and the systems that we propose in the subsequent chapters.

2.2 Neural Networks in Natural Language Processing

In the context of Natural Language Processing (NLP), the term statistical language model refers to a probability distribution over a sequence of input tokens, usually characters or words. Given a sequence of observed values, $x_1, x_2, \ldots, x_{\tau-1}$, a statistical language model computes the probability of the upcoming value x_τ to occur, $p(x_\tau|x_1, x_2, \ldots, x_{\tau-1})$. Among other approaches for statistical language modelling, such as n-gram models, neural network implementations have exhibited state-of-the-art performance over recent years (Bengio et al., 2003; Mikolov et al., 2010; Graves, 2013). Furthermore, in many of the cases, language models based on neural networks are one

of the fundamental components of many state-of-the-art NLP systems in tasks ranging from Question Generation (Serban et al., 2016; Du et al., 2017) and syntactic consistency parsing (Vinyals et al., 2015c) to image captioning (Karpathy and Fei-Fei, 2017; Vinyals and Le, 2015) and automatic response generation (Vinyals et al., 2015a; Wen et al., 2015, 2016; Vougiouklis et al., 2016). In the context of many statistical approaches (i.e. those that do not achieve text generation using templates (Konstas and Lapata, 2012a,b, 2013)), and especially, neural networks, a language model is essentially what interconnects Natural Language Processing and Generation (see Figure 2.1).

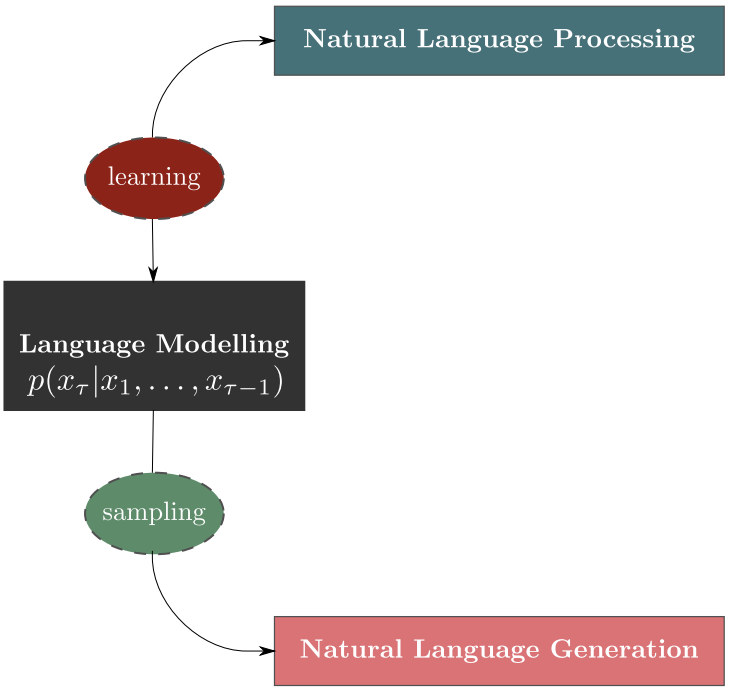


FIGURE 2.1: A statistical Language Model is the probability distribution that keeps NLP and NLG interconnected. As part of an NLP task, we would want to learn a language model and then sampling from this learned model is what carries out the NLG process.

2.2.1 Language Modelling with Neural Networks

In this section, we present some of the fundamental language models that are based on neural networks. We start with Bengio et al.'s feed-forward neural network model (Bengio et al., 2003), which introduced neural nets as a competent tool in the NLP field. We then explore the architecture of Recurrent Neural Networks along with their training difficulties that arise from the manifestation of the *vanishing* and *exploding* gradient problems. Subsequently, we focus our attention on the multi-gated RNN variants of the Long Short-Term Memory (LSTM) cell and the Gated Recurrent Unit (GRU). Lastly, the basic principles regarding the employment of neural network architectures as generative models are illustrated.

Please note that since *bias* terms can be included in each weight-matrix multiplication (Bishop, 1995), they are not explicitly displayed in the equations that describe the models of this chapter. Unless explicitly mentioned all weight matrices include a bias term, and corresponding input vectors are with an additional element with value 1.

2.2.1.1 Modelling with Feed-Forward Neural Networks

The feed-forward neural network language model was introduced by Bengio et al. in 2003 and constitutes the first known implementation of neural nets in the field of language modelling (Bengio et al., 2003). The model is based on the Markov assumption according to which the probability of a word x_i occurring is based only on the $k-1$ words that precede it. The model computes the probability of a sequence of words $\mathbf{x} = x_1, x_2, \ldots, x_T$ to occur as follows:

$$p(\mathbf{x}) = \prod_{t=1}^{T} p(x_t | x_{t-k+1}, \ldots, x_{t-1}) \ . \tag{2.1}$$

Given a sequence of one-hot[1] input vectors $x_{t-k+1}, x_{t-k+2}, ..., x_{t-1}$, the network computes the probability of generating the word that follows x_t as follows:

$$h_t^0 = [\mathbf{W_{x \to h}} x_{t-k+1}; \ldots; \mathbf{W_{x \to h}} x_{t-1}] \ , \tag{2.2}$$

$$h_t^1 = q(\mathbf{W^1} h_t^0) \ , \tag{2.3}$$

$$h_t^l = q(\mathbf{W^l} h_t^{l-1}) \ , \ 1 < l \leq L \ , \tag{2.4}$$

$$y_t = \mathbf{W_y} h_t^L \ , \tag{2.5}$$

where $l \geq 1$, $[\ldots;\ldots]$ represents vector concatenation and $q(z)$ is a non-linear activation function. Typical choices include the hyperbolic tangent and the logistic sigmoid function.

$$q(z) = \frac{1}{1 + \exp(-z)} \tag{2.6}$$

$$q(z) = \tanh(z) = \frac{\exp(z) - \exp(-z)}{\exp(z) + \exp(-z)} \tag{2.7}$$

The original model is parameterised by three weight matrices (Bengio et al., 2003):

- the input weight matrix $\mathbf{W_{x \to h}} : \mathbb{R}^V \to \mathbb{R}^m$ that contains the m-dimensional embeddings for each word in the dictionary of size V;
- the parameter matrix $\mathbf{W^1} : \mathbb{R}^{km} \to \mathbb{R}^m$ that compresses the concatenated vector of the input embeddings to a space of lower dimensionality m; and,

[1] One-hot is a vector that contains a 1 at the index of a particular token, usually character or word, in the vocabulary with all the other values set to zero.

- the output matrix $\mathbf{W_y} : \mathbb{R}^m \rightarrow \mathbb{R}^V$ that projects the hidden state on the output layer and similarly to the $\mathbf{W_{x \rightarrow h}}$ matrix, it has an entry for each word in the vocabulary V.

Eq. 2.4 is applied only in case of architecture with higher number of hidden layers. In such scenarios the parameter matrix $\mathbf{W^l} \in \mathbb{R}^{m \times m}$ acts as a biased linear mapping (Sundermeyer et al., 2015).

In case of a word-level model, the probability distribution over the next word given the previous history is obtained by applying the softmax activation function that is able to guarantee that the aggregated probability of all the observed words will sum up to 1:

$$p(x_t|x_1, \ldots, x_{t-1}) \approx p(x_t|x_{t-k+1}, \ldots, x_{t-1}) = \frac{\exp(y_t^{x_t})}{\sum_{v=1}^{V} \exp(y_t^{(v)})} \quad . \tag{2.8}$$

The model achieved substantial perplexity[2] improvements over the state-of-the-art n-gram models. Furthermore, Bengio et al. introduced a novel mixture model by combining the predictions that are computed by the neural network with the ones that are calculated by an interpolated tri-gram model (Chen and Goodman, 1996). The fact that the mixture architecture achieved even greater performance in terms of perplexity dictates that both its component-models make enough mistakes that benefit from simple averaging (Bengio et al., 2003).

2.2.1.2 Modelling with Recurrent Neural Networks

Mikolov et al. suggested a word-level language model based on the Recurrent Neural Networks (RNNs). The proposed Recurrent Language Model (RLM) (Mikolov et al., 2010, 2011) is based on the architecture of the *simple* RNN or Elman network (Elman, 1990) that is displayed in Figure 2.2.

Let $h_t^l \in \mathbb{R}^m$ be the aggregated output of a hidden unit at timestep $t \in [1 \ldots T]$ and layer depth $l \in [1 \ldots L]$. The vectors at zero layer depth, $h_t^0 = \mathbf{W_{x \rightarrow h}} x_t$, represent vectors that are given to the network as an input. The parameter matrix $\mathbf{W_{x \rightarrow h}}$ contains the m-dimensional embeddings for each token usually, words or character, in the dictionary of size V. The vectors at layer depth $l = L$, h_t^L, are used to predict the output vectors y_t represent the vectors of our architecture. All the weight matrices that follow have dimension $[m, m]$.

$$h_t^l = q\left(\mathbf{W_{in}^l} h_t^{l-1} + \mathbf{W_{h \rightarrow h}^l} h_{t-1}^l\right) \quad , \tag{2.9}$$

$$y_t = \mathbf{W_y} h_t^L \quad , \tag{2.10}$$

[2]Perplexity is an evaluation metric that describes how good a model's probability distribution predicts a sample. In information theory, it is usually used to compare language models.

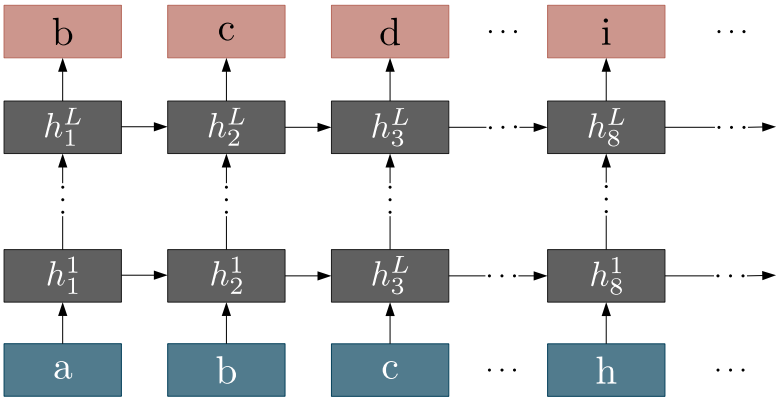


FIGURE 2.2: The general architecture of a multi-layer Recurrent Neural Network (RNN). The nodes in cyan, orange and dark grey represent inputs, outputs and hidden states respectively. The weights are shared across all the unrolling timesteps. At each timestep t the network is provided with the token of the current input (e.g. "a", "b", "c" and "h") and the hidden states of the previous timestep $h_{t-1}^{1\cdots L}$ and makes a prediction about the following token. Essentially, the vectors of the hidden states across its layers enable the model to keep track of the information that has been processed up to the current timestep in the sequence.

where $q(z)$ is the sigmoid activation function. The model is parameterised by three weight matrices:

- the weight matrix $\mathbf{W_{in}^l}$ that has dimension $[m, m]$ and contains the weights of the connections between the hidden nodes of the l and the $l-1$ layer;
- the recurrent matrix $\mathbf{W_{h\to h}^l} \in \mathbb{R}^{m\times m}$ that decides which part of h_{t-1}^l should be considered for the computation of the hidden state at the current timestep t; and,
- an output matrix $\mathbf{W_y} : \mathbb{R}^m \to \mathbb{R}^V$ that projects the hidden state on the output layer, and similarly to the $\mathbf{W_{x\to h}}$ matrix, it has an entry for each word or character, depending on the level of the model, in the vocabulary V.

Similar to the feed-forward neural network model discussed in Section 2.2.1.1, the probability distribution over the next word, given the previous history, is obtained by applying the softmax activation function as follows:

$$p(x_t|x_1,\ldots,x_{t-1}) = \text{softmax}(y_t) \quad . \tag{2.11}$$

Due to their intrinsic ability to cycle information inside the nodes of their high-dimensional hidden state, RNNs are, in principle, extremely powerful sequence models (Sutskever

et al., 2011; Graves, 2013). In contrast to Hidden Markov Models (HMMs), a widely used approach for sequence processing (Gers et al., 2003), they are not limited to discrete internal states but store information as continuous high-dimensional distributed representations (Hochreiter et al., 2001; Martens and Sutskever, 2011).

Mikolov et al. proposed an extended version of the initial RLM model, by introducing context-related parameters. They implemented a feature layer that is connected with the hidden and the output layers of the original RLM architecture (see Figure 2.2). The initial complementary values of the feature vector are computed in a pre-training phase by applying Latent Dirichlet Allocation (LDA) (Blei et al., 2003) on a preceding textual dataset. This approach has given rise to mechanisms that bootstrap additional features in the high-dimensional hidden state of the RNNs, leveraging their ability to retain relevant information across timesteps (Sordoni et al., 2015; Wen et al., 2015).

On the Difficulty of Training Recurrent Neural Nets. Training RNNs with gradient descent is not trivial. Substantial amount of literature (Bengio et al., 1994; Kolen and Kremer, 2001; Pascanu et al., 2012) has been published regarding the *exploding* and *vanishing gradient* problem which constitutes the most fundamental difficulty of training RNNs. The explosion of the long-term components that can occur during training is responsible for the exponential increase, or exploding, of the Euclidean norm of the gradient. Similarly, when the norm of the gradient of the long-term components diminishes quickly to 0, the vanishing problem manifests itself; thus, any information correlation between distant timesteps is deemed impossible. To sidestep this problem, Mikolov suggested a technique for the initialisation and monitoring of the weights of the hidden layers' connections (Mikolov et al., 2010, 2011). He calculated the gradients of the RNN with BackPropagation Through Time (BPTT) (Rumelhart et al., 1986; Bishop, 1995) by propagating the error for only a specific number of timesteps, without, however, being able to capture contextual information that spans hundreds of timesteps.

During the 90s there were many attempts to resolve the problem that is associated with the long-term dependencies of the RNNs. The proposed solutions introduced either non-gradient based training algorithms (Bengio et al., 1994) or *weight noise* to the predictions (Graves, 2013) before feeding them again into the network. However, none of these approaches were able to introduce a concrete methodology that would allow RNNs to achieve the necessary stability during training (Hochreiter and Schmidhuber, 1996). In order to sidestep the *exploding* and *vanishing gradients* training problem of RNNs, multi-gated RNN variants that are not affected by the long-term dependency issues, such as the Long Short-Term Memory (LSTM) cell (Hochreiter and Schmidhuber, 1997) and the Gated Recurrent Unit (GRU) (Cho et al., 2014), have been proposed. Initial implementations of LSTMs exhibited promising results in the field of robotic control (Mayer et al., 2006) and handwritten text recognition (Graves and Schmidhuber, 2009). The suggested gated mechanisms essentially allow gradients to flow backward more easily

without suffering too much from the vanishing effect (Bengio et al., 1994; Pascanu et al., 2012; Bahdanau et al., 2014). More details regarding the architecture of the LSTM and GRU models are provided in the Sections 2.2.1.3 and 2.2.1.4.

Recently, Martens employed a variant of the Hessian-Free optimisation (Martens, 2010) for training RNNs (Martens and Sutskever, 2011). The combination of the Hessian-Free optimisation method along with the introduction of a novel mechanism of "structural damping" was able to train RNNs on two different long-term dependant datasets (Sutskever et al., 2011). However, it should be mentioned that Martens's approach for training RNNs was found to be more computational expensive in comparison with the state-of-the-art LSTM-based model, trained with simple stochastic gradient descent.

2.2.1.3 Models based on Long Short-Term Memory Cells

The architecture of an RNN based on Long Short-Term Memory (LSTM) neural net is identical to the one of a simple RNN (see Figure 2.2). The main difference is that each node of the hidden state is implemented as an LSTM cell, and, as a result, the hidden state h_t^l at each timestep t and layer depth l is computed through the LSTM operations that are described below rather than with Eq. 2.9. The architecture of the LSTM cell is displayed in Figure 2.3.

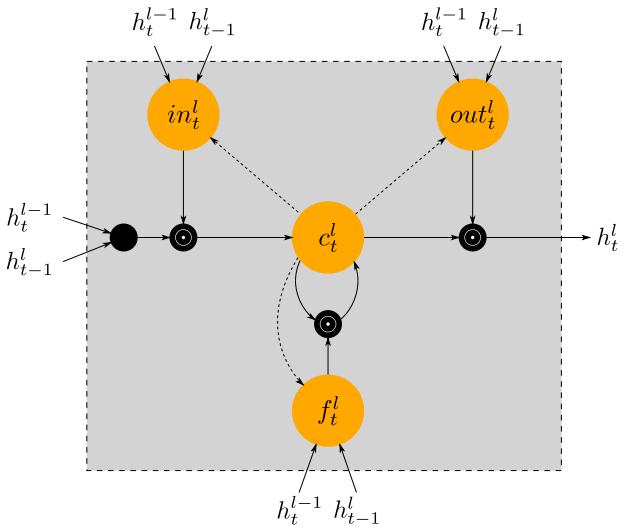


FIGURE 2.3: The architecture of the Long Short-Term Memory (LSTM) cell (Graves, 2013; Zaremba and Sutskever, 2014).

Based on the architecture that has been proposed by Hochreiter and Schmidhuber, Graves introduced an LSTM implementation for language modelling purposes. Inspired by Graves, Zaremba and Sutskever proposed an almost identical architecture for the LSTM absolved from the inclusion of the cell state in the computation of the input, output and forget gates (see connections depicted as dashed lines in Figure 2.3). The resultant architecture was used extensively in many generative tasks (Vinyals et al.,

2015c,b; Wen et al., 2015; Karpathy et al., 2015; Dong and Lapata, 2016). Its functionality is described below.

Let $h_t^l \in \mathbb{R}^m$ be the aggregated output of a hidden unit at timestep $t \in [1 \dots T]$ and layer depth $l \in [1 \dots L]$. All subsequent matrices have dimension $[m, m]$ unless stated otherwise. The model achieves its functionality by computing the following vectors:

$$\begin{pmatrix} in_t^l \\ f_t^l \\ out_t^l \\ \tilde{c}_t^l \end{pmatrix} = \begin{pmatrix} \text{sigm} \\ \text{sigm} \\ \text{sigm} \\ \tanh \end{pmatrix} \mathbf{W_c^l} \begin{pmatrix} h_t^{l-1} \\ h_{t-1}^l \end{pmatrix} \ , \tag{2.12}$$

$$c_t^l = f_t^l \odot c_{t-1}^l + in_t^l \odot \tilde{c}_t^l \ , \tag{2.13}$$

$$h_t^l = out_t^l \odot \tanh(c_t^l) \ , \tag{2.14}$$

where $\mathbf{W_c^l} : \mathbb{R}^{2m} \rightarrow \mathbb{R}^{4m}$ 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.

2.2.1.4 Models based on Gated Recurrent Units

The GRU (Cho et al., 2014) is a less complex variant of the LSTM cell with comparable performance (Chung et al., 2014). Similarly to the LSTM cell, it adaptively captures dependencies of different time scales without, however, the inclusion of a memory cell mechanism. The architecture of the GRU is displayed in Figure 2.4.

Let $h_t^l \in \mathbb{R}^m$ be the aggregated output of a hidden unit at timestep $t \in [1 \dots T]$ and layer depth $l \in [1 \dots L]$. All subsequent matrices have dimension $[m, m]$ unless stated otherwise. The model computes:

$$\begin{pmatrix} r_t^l \\ u_t^l \end{pmatrix} = \begin{pmatrix} \text{sigm} \\ \text{sigm} \end{pmatrix} \mathbf{W_u^l} \begin{pmatrix} h_t^{l-1} \\ h_{t-1}^l \end{pmatrix} \ , \tag{2.15}$$

$$\tilde{h}_t^l = \tanh(\mathbf{W_{in}^l} h_t^{l-1} + \mathbf{W_{h \rightarrow h}^l}(r_t^l \odot h_{t-1}^l)) \ , \tag{2.16}$$

$$h_t^l = (1 - u_t^l) \odot h_{t-1}^l + u_t^l \odot \tilde{h}_t^l \ , \tag{2.17}$$

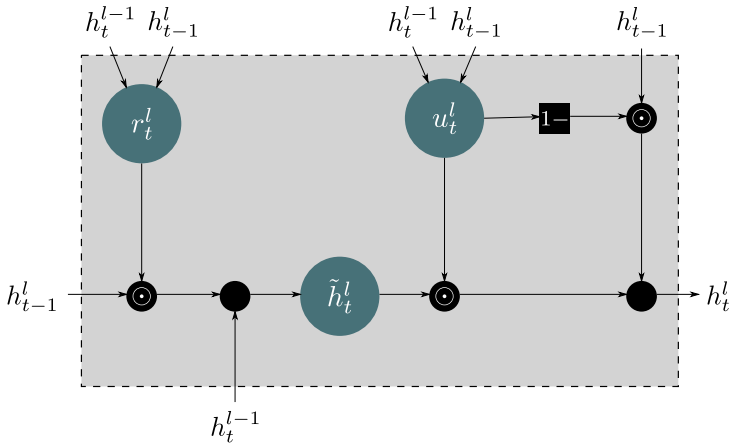


FIGURE 2.4: The architecture of the Gated Recurrent Unit (GRU) (Cho et al., 2014).

where $\mathbf{W_u^l} : \mathbb{R}^{2m} \rightarrow \mathbb{R}^{2m}$ is a biased linear mapping, and r_t^l, u_t^l and $\tilde{h}_t^l$ 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 $\tilde{h}_t^l$. 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.

2.2.2 Neural Networks as Generative Models

In the context of language modelling, the goal of the architectures that we discussed in Sections 2.2.1.1—2.2.1.4 is to calculate a sequence of probability distributions using their output vectors $y_1, y_2, \ldots, y_T$, given an one-hot input sequence $x_1, x_2, \ldots, x_T$:

$$p(x_t|x_1, \ldots, x_{t-1}) = \text{softmax}(y_t) \ . \tag{2.18}$$

Consequently, each model learns a probability distribution over sequences by utilising the negative cross-entropy[3] criterion (Sutskever et al., 2011; Graves, 2013; Sordoni et al., 2015). The model computes how far the generated sequence of tokens is from the empirical, actual text that is used for its training by utilising the negative logarithmic probability of the generated summary:

$$\text{cost} = -\sum_{t=1}^{T} \log \ p(x_t|x_1, \ldots x_{t-1}) \ . \tag{2.19}$$

[3]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 a set of parameters θ. It is defined as: $H(X) = -\sum q(y^{(i)}) \log p(y^{(i)}|x^{(i)}, \theta)$.

Consequently, our model tries to minimise the above cost function. This non-convex optimisation problem is solved using Back-Propagation (Rumelhart et al., 1986).

Text can be generated by sampling from the above mentioned conditional distribution to retrieve the next embedding and use it as the new input to the neural network. All of the models that are presented in this section can be implemented either as character-level or word-level models. Word-level implementations have been found to outperform equivalent character-level approaches (Sutskever et al., 2011; Mikolov, 2012). However, it should be noted that in a word-level model, the length of the one-hot input vectors is equal to the number of words in the dictionary, which proves to be extremely challenging not only in terms of the size of the vectors that need to be computed, but also in terms of the size of the dataset that is required for all the possible syntactic variations of each word to be successfully captured. Additionally, the existence of non-word strings, such numbers or URLs in the training data, raises additional challenges in the applicability of word-level implementations. However, despite their reported limitations, the majority of NLG-oriented research using neural networks has focused on word-level representations (Mei et al., 2016; Lebret et al., 2016; Chisholm et al., 2017; Wiseman et al., 2017).

Reflecting on Neural Language Models. In this section we attempt to make a brief comparison of the above-discussed neural architectures.

A major defect of the feed-forward approaches that are based on Bengio et al. is that they require a fixed-length context window that should be set every time before the training procedure (Mikolov et al., 2010). Consequently, when the network attempts to predict the next word has a relatively minimal contextual knowledge of the $5-10$ words that precede it. The superiority of recurrent neural networks over their feed-forward equivalents for languages modelling has been discussed extensively in the scientific literature (Mikolov et al., 2011; Arisoy et al., 2012). When tested on datasets from the Wall Street Journal the RLM performed significant better than the feed-forward model by achieving at least 10% less perplexity (Mikolov et al., 2011).

In terms of models that are based on RNN, the multi-gated-paradigms of LSTM and GRUs have proven reliable at absolving the recurrent architecture from the inherent exploding and vanish gradient problems. These models can be efficiently trained with stochastic gradient descent or rmsprop[4] (Graves, 2013), without the employment of sophisticated optimisation techniques (Bengio et al., 1994; Martens and Sutskever, 2011; Sutskever et al., 2011) that in most cases seem to be computational expensive to use. As a result, neural nets based on LSTMs and GRUs are currently the state-of-the-art approaches for language modelling, and are a fundamental component of many systems that are employed in the NLP field (Sundermeyer et al., 2012, 2015).

[4]rmsprop is a variant of stochastic gradient descent where the gradients are divided by a running average of their recent magnitude.

Recent literature shows that by training even on relatively small datasets ($\leq$ 5Mbytes), RNNs, regardless of the level of their model (i.e. character or word), are able to learn to generate grammatically and syntactically correct textual content (Graves, 2013; Karpathy et al., 2015).

However, the goal of this research is to generate a textual summary of a given set of Semantic Web triples. Consequently, in the following section we describe a framework that conditions the generative process of a language model to an input data structure.

2.2.3 Encoder-Decoder Framework

Based on the success of RNNs and their multi-gated variants on many tasks involving sequence modelling (Mikolov et al., 2010; Sundermeyer et al., 2012; Graves, 2013; Sordoni et al., 2015), Cho et al. and Sutskever et al. have proposed the encoder-decoder framework using GRUs and LSTM cells respectively. The architecture of the general encoder-decoder framework is displayed in Figure 2.5. Implementations based on the encoder-decoder framework work by mapping sequences of source tokens to sequences of target tokens. The model essential consists of two RNNs, one that *encodes* the sequence of source tokens into a vector of fixed dimensionality, and one that accepts the computed vector and starts to *decode* the expected sequence of target tokens. The decoder is essentially a language model conditioned on the input sequence. Given a sequence of input tokens $\mathbf{x} = x_1, \ldots x_\tau$, the model computes the probability of its corresponding output sequence of tokens $y_1, \ldots, y_T$ whose length T may be different from τ as follows:

$$p(y_1, \ldots, y_T) = \prod_{t=1}^{T} p(y_t | y_1, \ldots y_{t-1}, \mathbf{x}) \ , \tag{2.20}$$

The framework learns to encapsulate all the information that it requires from the input sequence in the hidden state of the last encoding timestep, h_τ. In order for the framework to work properly the existence of a special end-of-sequence token (i.e. `<end>` in Figure 2.5) in the single dictionary of the encoder[5] and both the source and target dictionaries of the decoder is required. Essentially, this is the token that enables the model to define a distribution over sequences of all possible lengths. Some approaches have employed an additional special start-of-sequence token (`<start>`) (Dong and Lapata, 2016). A special start-of-sequence token (`<start>`) is not explicitly required to be appended in the source dictionaries of the encoder and the decoder. In Figure 2.5, the `<start>` token is used in favour of a better representation of the start and the end of the encoding and the decoding process. In case the system is not equipped with such a

[5]In most of the cases, the framework does not do any predictions on the encoder level, and, as a result, the encoder is not required to have a target dictionary.

token, decoding starts when the decoder is provided with the hidden state of the input sequence $h_\tau^{1...L}$ and the corresponding end-of-sequence token (Sutskever et al., 2014).

Adaptations of this framework have demonstrated state-of-the-art performance in many generative tasks, such as machine translation (Cho et al., 2014; Sutskever et al., 2014), and conversation modelling and response generation (Vinyals et al., 2015a; Shang et al., 2015).

Handling Longer Sequences with an Attentive Encoder. One of the challenges associated with the general encoder-decoder framework that is presented above is its ability of handling very long sequences (Bahdanau et al., 2014). The problem is two-fold. First, in the case of long inputs, the encoder is forced to compress all the information that is relevant for the text generation procedure in a single vector (Bahdanau et al., 2014). Second, in the case of long expected outputs, it is challenging for the decoder to retain the information from the input at very distant timesteps. In order to alleviate this problem Bahdanau et al. and Luong et al. introduced an attention mechanism over the computed hidden states (e.g. $h_1^L, \ldots, h_\tau^L$ in Figure 2.5) of the encoder. The inclusion of this mechanism allowed them to generate high quality translations even for very long sentences (more than 50 words).

Rather than enforcing the model to compress in a single vector all the available information that is contained in an input sequence, the attention mechanism allows the model to learn to "align" the source information to the requirement of each expected output. At each decoding timestep, the model computes a relevance score of the current hidden state with each one of the hidden states (e.g. $h_1^L, \ldots, h_\tau^L$ in Figure 2.5) of the encoder. Based on this score a weighted average of the source hidden states is calculated which is fed into the decoder before it makes the prediction about the next token. Luong et al. has also explored a set of different scoring functions (i.e. "dot", "general" and "concat") without, however, conclusive remarks about the best one.

Table 2.1 attempts to document some of the key contributions in the field of neural network approaches for NLG.

2.3 Summary

This chapter has attempted to review existing approaches for NLG. Among other data-driven approaches neural networks have presented state-of-the-art performance in many NLG tasks. However, the application of neural networks on top of data encoded in triples is still a relatively unexplored domain. Our systems are inspired by the general encoder-decoder framework which we wish to adapt to the requirements of the Semantic Web.

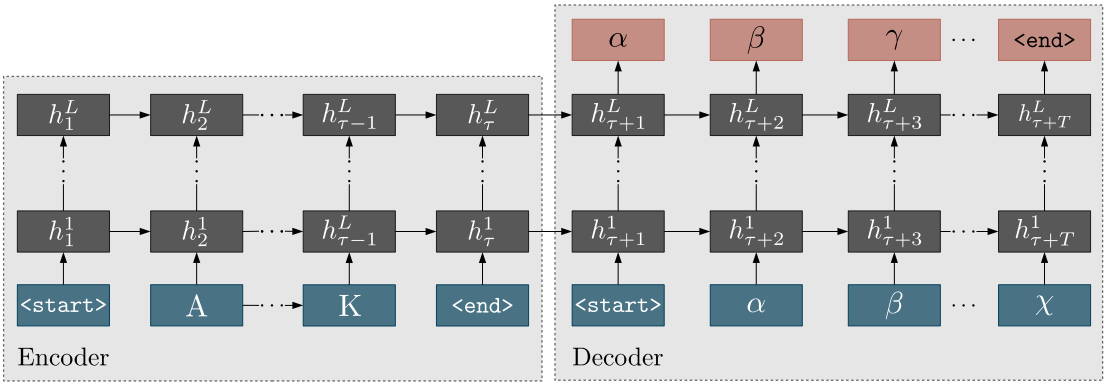


FIGURE 2.5: The architecture of the general encoder-decoder framework. The nodes in cyan, orange and dark grey represent inputs, outputs and hidden states respectively. The encoder captures the relevant information from the input sequence (i.e. "A", ..., "K") in the hidden state that is computed at the last encoding timestep h_τ^L. The decoding process starts when the decoder is provided with this hidden state and the special start-of-sequence token (i.e. `<start>`).

This framework along with other fundamental architectures based on neural networks were described in this chapter.

The next chapter discusses the different methodologies that have been explored in the literature for the evaluation of NLG systems, and describes in detail the methods and criteria against which the systems' that are proposed in this thesis are evaluated.

TABLE 2.1: Overview of major contributions in the field of neural network approaches for NLG.

Field of Contribution	Year	Author(s)	Summary
Language Modelling	2003	(Bengio et al., 2003)	Bengio et al. introduced the first neural network approach in the field of language modelling. The proposed feed-forward neural net achieved better perplexity than the state-of-the-art n-gram models.
	2010	(Mikolov et al., 2010)	Mikolov et al. introduced the Recurrent Language Model (RLM). The model was trained using BackPropagation Through Time (BPTT) (Rumelhart et al., 1986).
	2013	(Graves, 2013)	Graves trained a language model based on the LSTM architecture (Hochreiter and Schmidhuber, 1997) on the Wikipedia dataset, to generate encyclopedia-oriented content.
Optimisation and Neural Architectures	1986	(Rumelhart et al., 1986)	Rumelhart et al. introduced a way of propagating errors through a network backwards. Backpropagation still is the mostly-used algorithm for training neural networks.
	1997	(Hochreiter and Schmidhuber, 1997)	Hochreiter and Schmidhuber proposed the Long Short-Term Memory (LSTM) architecture as a modified RNN variant, absolved from the long-term dependencies issues.
	2014	(Cho et al., 2014; Sutskever et al., 2014)	Cho et al. and Sutskever et al. proposed the encoder-decoder framework using GRUs and LSTM cells respectively. The two systems achieved similar state-of-the-art performance in Machine Translation tasks.
	2014	(Bahdanau et al., 2014)	Bahdanau et al. introduced an attention mechanism on top of an encoder-decoder architecture based on GRUs. The inclusion of this mechanism allowed them to generate high quality translations even for very long sentences.
Text Generation	2016	(Mei et al., 2016)	Mei et al. use an encoder-decoder model equipped with LSTM cells, on both the encoder and the decoder side, and attention (Bahdanau et al., 2014) to generate textual description of robotic football matches (i.e. RoboCup dataset) and weather forecasts (i.e. WeatherGov).
	2016	(Lebret et al., 2016)	Lebret et al. employ an adaptation of the general encoder-decoder in order to generate one-sentence Wikipedia biographies from Wikipedia infoboxes. Their system works by conditioning a neural language model based on a feed-forward architecture on the tabular data from a Wikipedia infobox. They showed extremely promising results on a dataset with a challenging vocabulary size of 400k.
	2017	(Chisholm et al., 2017)	Similarly to Lebret et al., Chisholm et al. focus on Wikipedia biographies generation. They use an encoder-decoder model equipped with LSTM cells, on both the encoder and the decoder side, in order to generate one-sentence biographies from Wikidata slot-value pairs.

Chapter 3

Evaluation Methodology

In this chapter, we will present the methodology that we follow in order to evaluate the performance of our proposed NLG systems in Chapters 5—7. The employed methods for automatic and human evaluation along with the baselines, against which we compared our architectures, are described in detail in the following sections.

3.1 Evaluation Methods

Related literature suggests three ways of determining the efficacy with which an NLG system achieves its communicative goal. The first approach, which is usually referred to as *metric-based corpus evaluation* (Reiter and Belz, 2009; Reiter, 2010), use text-similarity metrics, such as BLEU (Papineni et al., 2002), ROUGE (Lin, 2004) and METEOR (Lavie and Agarwal, 2007). These metrics essentially compare how similar the generated texts are to the empirical texts of a designated corpus. In contrast to the above approaches for automatic evaluation, the other two alternatives involve human subjects, and are performed through either: (i) *task-based* or (ii) *judgement-based* (*rating-based*) evaluation (Reiter and Belz, 2009; Reiter, 2010). Since NLG systems are intended to assist human in carrying out a particular task (e.g. learning about a topic or improving an article's writing experience), task-based evaluations seek to measure the direct impact of the generated texts on the end user. Judgement-based evaluation is based on compiling a set of criteria against which human evaluators rate the quality of the generated texts (Reiter and Belz, 2009; Reiter, 2010).

3.1.1 Automatic Evaluation

Metrics for automatic evaluation have gained increased amount of interest in the most recent NLG-related literature. The BLEU, ROUGE and METEOR metrics are extensively used in machine translation (Cho et al., 2014; Sutskever et al., 2014; Bahdanau et al., 2014) and text summarisation (Rush et al., 2015; Chen et al., 2016; Nallapati et al., 2016) tasks, offering a quick and cheap way of evaluating the performance of their systems (Reiter and Belz, 2009). BLEU scores have been used to evaluate the quality of automatically generated descriptions of football (Angeli et al., 2010; Kim and Mooney, 2010; Konstas and Lapata, 2012b, 2013; Mei et al., 2016) and basketball matches (Wiseman et al., 2017), biographies (Lebret et al., 2016; Chisholm et al., 2017), weather forecasts (Reiter and Belz, 2009; Konstas and Lapata, 2012b, 2013; Mei et al., 2016), descriptions of diseases (Sauper and Barzilay, 2009), microplanning and realisation of knowledge graphs (Gardent et al., 2017b) and dialogue utterances on a flight-booking system (Konstas and Lapata, 2012a) given a set of structured records as input. In some cases the reported BLEU scores are accompanied by results using perplexity (Lebret et al., 2016), the METEOR (Konstas and Lapata, 2012a; Gardent et al., 2017b) and the ROUGE (Sauper and Barzilay, 2009; Reiter and Belz, 2009; Lebret et al., 2016) metrics.

A short description of the aforementioned metrics is provided below.

Perplexity. is regarded as one of the most standard methods for language modelling (Lebret et al., 2016). It measures the cross-entropy between the predicted sequence of words and the actual, empirical, sequence of words.

BLEU (Bilingual Evaluation Understudy). BLEU (Papineni et al., 2002) 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 (i.e. 1-grams, ..., n-grams) in the generated and the actual text. It is marked on the range of 0 to 100.

ROUGE (Recall-Oriented Understudy for Gisting Evaluation). 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 (Lin, 2004). It is marked on the range of 0 to 100.

METEOR. METEOR computes a weighted average of the precision and recall of unigrams in the generated and the empirical text by considering stemming, synonyms and paraphrases (Lavie and Agarwal, 2007). It is marked on the range of 0 to 100.

Perplexity indicates how well the model learns its training objective (c.f. Section 5.1.4); BLEU and ROUGE measure how close the generated text is to the actual real piece

of text. BLEU and ROUGE are complimentary to each other. The first computes a modified version of n-gram precision[1], whereas the latter computes the n-gram recall, of the automatically generated sentences with respect to the actual Wikipedia summaries.

We report the performance of the proposed systems using perplexity, the BLEU, ROUGE and METEOR metrics on the validation and test set of each dataset (see Sections 5.3.2, 6.3.2 and 7.3.2). We adapt the code from the evaluation package that was released by Peter Anderson[2], which was originally implemented to evaluate the quality of textual descriptions from images. 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 for all our models in the subsequent chapters.

Putting the Scores Into Perspective. In order to put our reported scores into perspective, we briefly survey how other state-of-the-art systems involving surface realisation perform with respect to the above automatic evaluation metrics. The CNN/Daily Mail corpus (Hermann et al., 2015) has been used for the training and evaluation of systems that seek to generate multi-sentence summaries of news articles (Nallapati et al., 2016, 2017; See et al., 2017; Paulus et al., 2018). The reported ROUGE_L scores of those systems ranges from 32.65 (Nallapati et al., 2016) to 39.08 (Paulus et al., 2018). See et al. also report their performance using METEOR at 18.72 (with 36.38 respective ROUGE_L score). BLEU 4 has been the predominant metric for the evaluation of machine translation systems. However, the reported scores vary according to the complexities of the source and target language. For instance, the state-of-the-art performance for translating from English to German and vice versa is around 48 BLEU whereas for English to Ethiopian and Finnish is around 25 and 19 BLEU respectively (Bojar et al., 2017, 2018).

In the context of NLG, the expected range of scores using methods for automatic evaluation depends on the challenges of each task and the length of the expected generated content (e.g. number of sentences and length of each generated sentence). The system proposed Mei et al. achieved state-of-the-art performance for generation of single-sentence descriptions of robotic football matches and weather forecast with respective BLEU 4 scores of 25.28[3] and 61.01. The best performing systems in the WebNLG challenge that involves microplanning and realisation of knowledge graphs in single-sentence descriptions scored around 45 and 39 BLEU 4 and METEOR points respectively (Gardent et al., 2017b). The reported BLEU scores for a more challenging task involving generation of multi-sentence descriptions of basketball matches are considerably lower at 14.49 (Wiseman et al., 2017). Both Lebret et al. and Chisholm et al. focus on the

[1]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.

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

[3]Similarly to our case, the experiments involved full selective generation. Angeli et al.; Kim and Mooney; Konstas and Lapata; Konstas and Lapata and Mei et al. considered an alternate experiment in which ground truth content selection was assumed at test time resulting in higher BLEU scores.

generation of single-sentence biographies in English. The best performing system on this task achieves a BLEU 4 score of 41 (Chisholm et al., 2017).

Evaluating NLG Systems Beyond Text Similarity. While automatic text similarity metrics (i.e. BLEU and ROUGE) can be used to evaluate the linguistic quality of the generated texts, they do not always correlate with the respective human ratings on the content quality (Reiter and Belz, 2009; Reiter, 2010). BLEU, ROUGE and METEOR are well-established metrics in domains (e.g. machine translation and text summarisation) where there is a tight alignment between the source and the generated language. Our generative task is more challenging since it consists of learning to generate text from a corpus of knowledge base triples loosely associated with text. As a result, 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).

In order to address the limitation of the above automatic evaluation metrics, we further evaluate our proposed systems in pilot user studies. In the next section, we describe the criteria that have been selected in the conducted user studies, and we relate them to existing works.

3.1.2 Human Evaluation

In the context of evaluating textual realisation of structured records with human subjects, task-based evaluations are considered the most trusted ones. These are extrinsic evaluations that seek to directly measure the impact of an NLG system to the end-user (Mellish and Dale, 1998; Reiter and Belz, 2009; Reiter, 2010). While reliable, this kind of human evaluation tend to be extremely time-consuming and expensive (Reiter and Belz, 2009; Reiter, 2010). Representative examples are the extrinsic evaluations of the SkillSum (Williams and Reiter, 2008) and STOP (Reiter et al., 2003) systems that, including data analysis and planning, costed £75k and £25k respectively (Reiter, 2010). The former was evaluated for the accuracy of its generated literacy and numeracy assessments by a sample of 230 participants. The latter in a clinical trial with 2000 smokers, all of whom completed a smoking questionnaire in the first stage of the experiment, in order to find what portion of those that received the automatically generated letters from STOP had managed to stop smoking.

Given the challenges at carrying out extrinsic evaluations, research has mostly used judgement-based human evaluations for the performance assessment of NLG systems (Lester and Porter, 1997; Sun and Mellish, 2007; Chen and Mooney, 2008; Reiter and Belz, 2009; Chen et al., 2010; Angeli et al., 2010; Konstas and Lapata, 2012a,b, 2013;

Duma and Klein, 2013; Ngonga Ngomo et al., 2013; Ell and Harth, 2014; Chisholm et al., 2017; Wiseman et al., 2017). These are performed by asking human subjects to rate generated texts against a set of criteria that explore various aspects of the generated text's quality, and its semantic coherence with the given input (i.e. usually the structured records that have been provided to the system as input for the generation of the resultant text), or to compare texts that have been generated by different pipelines (i.e. other competing systems or humans). The evaluation is conducted by either a small group of experts, usually consisting of not more than 10 people[4], (Lester and Porter, 1997; Sun and Mellish, 2007; Chen and Mooney, 2008; Reiter and Belz, 2009; Duma and Klein, 2013; Ngonga Ngomo et al., 2013; Ell and Harth, 2014), or crowdworkers (Angeli et al., 2010; Konstas and Lapata, 2012a,b, 2013; Chisholm et al., 2017; Wiseman et al., 2017).

Fluency. Inspired by Chen and Mooney, Angeli et al., Konstas and Lapata and Ngonga Ngomo et al., in our experiments, we opted to evaluate the eloquence of the generated text (i.e. whether the text is comprehensible, grammatically and syntactically correct, and without any unnecessary redundancies) using a single criterion, *fluency*. Evaluating the eloquence of texts using several metrics (e.g. grammaticality, non-redundancy and coherence in (Sun and Mellish, 2007) or syntax, comprehensibility and overall quality in (Duma and Klein, 2013)) usually requires the employment of domain experts or linguists for the human evaluation. On the contrary, using a single metric provided us with the necessary flexibility to approach wider communities and crowdworkers for our evaluation purposes. Our participants (either experts or crowdworkers) were asked to rate the *fluency* of texts, following Ngonga Ngomo et al.'s rating scale[5]. Consequently, 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 (Ngonga Ngomo et al., 2013).

Coverage and Contradiction. Chen and Mooney, Angeli et al., and Konstas and Lapata asked their participants to rate the *semantic correctness* of the automatically generated weather forecasts and descriptions of robotic-football matches on a scale from 1 to 5. However, we believe that due to the nature of our task, a more descriptive set of criteria with respect to the extent that triples are explicitly or implicitly realised in a summary would enable us to evaluate our systems more effectively. Inspired by Ell and

[4] Ngonga Ngomo et al. had divided their human evaluation in three different tasks, two of which were exclusively conducted by domain experts. The recruitment was carried out using various Semantic Web- and NLP- oriented mailing lists and community channels resulting in a total of 115 participants (Ngonga Ngomo et al., 2013).

[5] Ngonga Ngomo et al. used fluency to evaluate the comprehensibility and readability of a generated question in natural language.

Harth, we use *coverage*[6] in order to identify the portion of input triples whose information is mentioned either implicitly or explicitly in the text. Furthermore, we introduce the notion of *contradiction*, and we define it as the percentage of input triple-facts that conflict with information that exists in the generated summary. Our participants (either crowdworkers or experts) evaluated the coverage of each summary by a competing system with respect to a set of triple-facts that generated it. Consequently, they had to identify the relation of each triple to the summary as either "Absent" or "Present" and "Direct Contradiction" or "Not a Contradiction" for the coverage and contradiction evaluation respectively. The final result for both coverage and contradiction is computed after the total number of triples that have been identified as "Present" and "Direct Contradiction" respectively is computed and normalised by the total number of input triples.

Additional Information. In our initial experiments, we included one additional criterion about the *additional information* that is presented in the generated summary, but it is not included in the provided input set of triples. This criterion is mostly intended for Semantic Web experts who are asked to define the number of triples to which potential additional information in the text can be interpreted. 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.

One of the challenges that is usually associated with adaptations of the general encoder-decoder framework is that occasionally their output, albeit fluent, is unrelated to their given input (Chisholm et al., 2017; Koehn and Knowles, 2017; Wiseman et al., 2017). Literature refers to such behaviour as "hallucinations". The two latter criteria, contradiction and additional information, provide us with the opportunity to investigate the extent to which our proposed models "hallucinate" factual statements that neither exist nor can be deduced from the input triples.

3.1.3 Evaluating Multilingual Summaries from the Perspective of Wikipedia Readers and Editors

As part of the collaborative work in which we investigated the generation of Wikipedia summaries for underserved languages, we wished to explore the usability of the automatically generated content by Wikipedia readers and editors. To this end, we designed a community study that adopts a middle-ground approach between task- and rating-based evaluation. In addition to reporting the performance of our approach against the

[6] Adapted from *coverage* in Ell and Harth (2014) which used to measure the number of included sub-graphs in the text.

competing baselines using automatic evaluation metrics, we extended the human evaluation methodology of Sauper and Barzilay. Sauper and Barzilay manually analysed the nature of human edits that have been performed on a set of 15 automatically generated descriptions of deceases that have been appended as new articles to Wikipedia. However, the fact that they were experimenting on the live version of Wikipedia constrained them from conducting a comparative analysis on a larger sample using descriptions generated by other competing systems (Sauper and Barzilay, 2009). In our experiments, we opted for imitating the reader and editor experience. This granted us the flexibility to explore the feasibility of our approach on a larger sample, against other baselines.

We conducted two community studies, one for readers and one for editors, of a particular Wikipedia language. The readers were asked to evaluate summaries generated by our proposed system and the competing baselines against two criteria: (i) fluency (as described in the "Fluency" part of 3.1.2) and (ii) *appropriateness*. In the second study, we evaluated the automatically generated summaries against a single criterion: *editors' reuse*. Appropriateness and editors' reuse are described in detail below.

Appropriateness We use this criterion in order to investigate whether our approach generates sentences that "feel" like Wikipedia text. The participants were asked in a binary format (i.e. "Yes" or "No") whether a displayed sentence could be part of a Wikipedia article.

Similarly to Lester and Porter, Sun and Mellish, Duma and Klein and Chisholm et al., we compare the performance of our systems against texts that have been generated by humans (cf. Section 6.4.2). We used the original reference Wikipedia sentences in order to compute the expected upper bound for this task. In addition, we used a sample of retrieved sentences from news. While we expected the latter to receive low appropriateness ratings, we wished to test the participants' ability to distinguish a Wikipedia sentence from others that serve unrelated purposes.

Editors' Reuse Our goal is to assess how editors use our generated summaries in their work by measuring the amount of text they reuse. Each editor is provided with the automatically generated summary that corresponds to a randomly selected item from the test set of the dataset of interest (either Esperanto or Arabic) along with its corresponding triples. We mock the editing experience by asking each participant to write a short summary of up to 3 sentences about the allocated item in a dedicated text field. The editors had the freedom to copy parts or the entirety of the provided summary or completely start from scratch. In order to quantify the amount of text that has been reused from the automatically generated summary, we use the Greedy String-Tiling (GST) algorithm (Wise, 1996). GST is a sub-string matching algorithm that computes the degree of reuse or copy from a source text. GST is able to deal with

cases when a whole block of text is transposed, unlike other string matching metrics, such as the Levenshtein distance (Levenshtein, 1966), that calculate it as a sequence of single insertions or deletions. Let $Y = y_1, y_2, \ldots, y_T$ be the sequence of tokens of an automatically generated summary and $Y' = y'_1, y'_2, \ldots, y'_{T'}$ be the sequence of tokens of an edited one. The GST metric identifies the set $\Phi = \{\phi_1, \phi_2, \ldots, \phi_\Theta : \phi_i \cap \phi_j = \emptyset \; \forall \phi_i, \phi_j \in \Phi, i \neq j\}$ of all the disjoint longest sequences of tokens in the edited text that also appear in the source text. These sequences of tokens, $\phi_1, \ldots, \phi_\Theta$, are usually referred to as *tiles*.

Named entities and common stop words are expected to be present in both the source and the edited text. However, we are mostly interested in exploring how much of the general structure of a generated summary is used in the edited one. Consequently, we retain the minimum-match-length parameter (i.e. the minimum length of tiles that are identified) to its default 3 value s.t. $\forall \phi_j \in \Phi : \phi_j \subseteq Y$ and $\phi_j \subseteq Y'$ and $|\phi_j| \geq 3$. This essentially means that transposed sequences of uni-grams or bi-grams are not regarded as tiles, and, as a result, are not counted in the calculation of reuse. We calculate a reuse score $\text{score}_{\text{GST}}$ by computing the accumulated length of the detected tiles which we subsequently normalise by the length of the generated summary.

$$\text{score}_{\text{GST}}(Y, Y') = \frac{\sum_{\phi_j \in \Phi} |\phi_j|}{|Y|} \quad . \tag{3.1}$$

where $\text{score}_{\text{GST}} \in [0, 1]$. Furthermore, we adapt a GST-based categorisation based on the categories that have been proposed by Clough et al.. While Clough et al.'s approach has been applied on classifying the extent to which text from newspaper articles has been reused by other news sources, we believe that their categories are directly applicable to our task. Consequently, we identify the following categories based on the $\text{score}_{\text{GST}}$ of a generated summary Y and edited one Y': (i) wholly-derived (**WD**) ($0.66 \leq \text{score}_{\text{GST}}$) when the generated summary has been fully reused in the text that has been written by the editor, (ii) partially-derived (**PD**) ($0.33 \leq \text{score}_{\text{GST}} < 0.66$) when a part of the generated summary has been used in the edited paragraph, and (iii) non-derived (**ND**) ($\text{score}_{\text{GST}} < 0.33$) when the part of the generated text that has been included in the edited one is either negligible or nonexistent.

3.2 Baselines

To demonstrate the effectiveness of our systems, we compare them to different baselines of different natures.

3.2.1 Random

We compute the expected lower bounds for the selected metrics for automatic evaluation by using a random Wikipedia summary generation baseline. For each triple set in the validation and test sets, the random system generates a response by uniformly randomly selecting a Wikipedia summary from the corresponding training set.

In order to make this baseline more competitive, we equip it with the mechanism of surface form tuples (see Section 5.2.1.2), and the `<item>` (see Section 4.2) and property-type or property placeholders (described in Section 5.1.3 and 6.1.1 respectively). We refer to the enhanced version of the random baseline as Random+. After a summary is selected, its `<item>` placeholders along with any potential property or 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 Sections 5.1.3 and 6.1.1).

3.2.2 Kneser-Ney (KN) Language Model

Similarly to (Lebret et al., 2016), we use the KenLM toolkit (Heafield et al., 2013) in order to build a 5-gram Kneser-Ney (KN) language model. During testing, we use beam-search with a beam of size B[7] (see Section 5.1.5), to generate the B most probable summaries for each triple set in the validation and test set. Similarly to the case of the Random baseline, we equip it with surface form tuples, and the `<item>` and property-type or property placeholders. We refer to this alteration of the original baseline as KN+.

3.2.3 Information Retrieval (IR)

Previous baselines do not incorporate the triples corresponding to each summary directly in the model. As a result, we build another baseline that conditions the output summary to the input triples by adapting an Information Retrieval (IR) baseline. Our implementation is similar to the baseline that has been used in other text generative tasks, such as textual summarisation (Rush et al., 2015) and generation of natural language questions (Du et al., 2017). First, the baseline encodes the sets of input triples that corresponds to each summary in a dataset into a sparse vector of TF-IDF weights (Joachims, 1997), the dimensionality of which is based on the corresponding training set. We reduce the dimensionality of the resultant sparse vectors to 200 using LSA (Halko et al., 2011). Finally, for each item (i.e. the vector representation of its set of triples) in the corresponding validation and test set, we perform k-NN to retrieve the closest vector based on

[7]We select the minimum beam size B for which there is no longer any improvement in performance by increasing it.

the Euclidean distance (l_2) from the training set, and output its corresponding summary. The summary that corresponds to the retrieved vector is used as the output summary for this item in the validation and test set.

We experimented with two variations of this baseline. The first one (IR) retrieves the raw summaries from the training set. The second one (i.e. IR+) retrieves summaries that are equipped with surface form tuples (see Section 5.2.1.2), and the `<item>` (see Section 4.2) and property-type or property placeholders (described in Section 5.1.3 and 6.1.1 respectively). In this format, a retrieved summary is, essentially, a template whose special tokens are replaced, such as the `<item>` and property or property-type placeholder, with their corresponding labels of the corresponding entities in the input triples' set at a post-processing step. Since IR+ outputs "memorised" templates of summaries, it can be considered as information retrieval over a set of templates from the training data. This approach is very similar to recent template-based system for document generation that has been proposed by Ell and Harth. The main difference is that our version selects the "templated" summary that corresponds to the set whose vector is the closest without considering the extent to which this "templated" summary is covering its corresponding set. Furthermore, Ell and Harth did not suggest a method for approximating the best template from a training set for verbalising a templated, but not always known, set of triples from the validation and test set.

3.2.4 Machine Translation (MT)

We also compare the performance of our approach at generating multilingual summaries (i.e. in Arabic and Esperanto) against a Machine Translation (MT) baseline. We used Google Translate on English Wikipedia summaries. These translations are compared to the actual target language's Wikipedia entry. This limited us to articles that exist in both English and the target language. The concepts in Esperanto that are not covered by English Wikipedia account for 4.3%. The concepts in Arabic that are not covered by English Wikipedia account for 30.5%. This also indicates the content coverage gap between different Wikipedia languages (Hecht and Gergle, 2010).

Translating content from popular and more complete Wikipedia pages has been proposed as method of enriching underserved ones, such as in Esperanto and Arabic, using the Content Translation Tool (Laxström et al., 2015). Nonetheless, we believe that there are certain limitations in adapting such a baseline for multilingual knowledge base completion. First, translating information from a source language constrains the textual information to potential biases introduced by the editor of the original text. Secondly, it is highly unlikely that the output text from MT would address the writing style and culture of the target Wikipedias (the information is neither the same nor is it presented in a similar manner across all different languages). Finally, post-editing of translations would require a lot of work by the community. This is partially because of the different

writing style of the translated sentence compared to the expected one; but more notably because ensuring that the information has been translated correctly prerequisites knowledge of both the source and the target language by the potential editors. As a result, comparing this baseline to our approach would also provide us with the opportunity to test the accuracy of our hypothesis regarding the inadequacy of MT for the above task.

3.3 Summary

This chapter has presented methods for the automatic and human evaluation of triples-to-text generation systems. We emphasise on the fact that while the available text similarity metrics have gained increased attention over the most recent years for the evaluation of NLG systems, human evaluation is still essential in determining the performance of an NLG system. The different criteria against which the human evaluation of our systems is performed are also discussed in detail. In the final section, the baselines against which our proposed systems are compared are presented.

In the following chapter, we propose a fully-automatic, cross-lingual, approach for building data-to-text corpora for the training and evaluation of our proposed NLG systems. We also explore the statistics of the resultant corpora that are used in the subsequent chapters.

Chapter 4

Building Corpora of Natural Language Texts Aligned with Knowledge Base Triples

Training data for NLG models is not always readily available. In our case the difficulty is that data that is available in knowledge bases needs to be aligned with the corresponding texts. In order to both train and evaluate the performance of our model, we propose different methodologies for building data-to-text corpora of rich linguistic variability.

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 SumTime-Meteo (Sripada et al., 2002), WeatherGov (Liang et al., 2009) and RoboCup (Chen and Mooney, 2008) datasets, or rely heavily on crowdsourcing (Gardent et al., 2017a), which makes them costly to apply in large domains.

In this chapter, we propose a fully-automatic approach, based on distant-supervision (Mintz et al., 2009), for building large corpora of loosely aligned Wikipedia snippets with triples from DBpedia and Wikidata. Our methodology can be easily extended to multiple domains and languages. Using this approach we built a set of different data-to-text corpora which we use, in the subsequent chapters of this thesis, for the training and evaluation of our proposed NLG systems. The methodology itself along with the resultant corpora are described in detail below.

4.1 An Automatic Approach of Building a Corpus of Triples Aligned with Natural Language Texts

Previous work has explored a set of different strategies in order to create the data-to-text corpora that are required for learning and testing. Chen and Mooney manually aligned database information from the RoboCup game finals with corresponding sentences extracted from the web. The resultant dataset has a small vocabulary of 214 words and a simple syntax (e.g. a transitive verb with its subject and object). Similar approaches have been adopted for the creation of datasets in the weather forecasts and the air travel domain. The SumTime-Meteo dataset consists of 1045 human-written weather forecasts that are aligned with numerical weather predictions (e.g. temperature, wind speed and precipitation) (Sripada et al., 2002). A much larger dataset that is employed in the same domain is WeatherGov. The WeatherGov dataset contains 29528 weather scenarios for 3753 major US cities and it has a vocabulary of 345 words (Liang et al., 2009). The ATIS dataset Dahl et al. consists of 5426 scenarios. It consists of transcriptions of spontaneous utterances of users interacting with a hypothetical online flight-booking system. Typically, such corpora are domain specific and of relatively small size while their linguistic variability is often restricted.

Our approach is inspired by Lebret et al. who automatically extracted a corpus consisting of 728321 biographies from English Wikipedia, each one of whom was associated to its corresponding Wikipedia infobox. In our case our goal is to align knowledge base triples with texts. We rely on the alignment of DBpedia and Wikidata with Wikipedia in order to create corpora of knowledge base triples from DBpedia and Wikidata, and their corresponding textual summaries. For the experiments presented in this thesis, we have extracted five different corpora that consist of: (i) 260k English Wikipedia biographies aligned with a total of 2.7M DBpedia triples (D1), (ii) 360k English Wikipedia biographies allocated to a total of 4.3M Wikidata triples (D2), (iii) 865k open-domain, English Wikipedia summaries aligned with a total of 7.4M DBpedia triples (D3), (iv) 256k open-domain Wikipedia summaries in Arabic aligned with a total of 2.1M Wikidata triples (M1), and (v) 127k open-domain, Wikipedia summaries in Esperanto allocated to a total of 1.4M Wikidata triples (M2). The first two associate English biographies with triples from different knowledge bases, and aim to explore the performance of our systems on a single-domain scenario. The third was designed to allow us to explore our proposed systems capability to generate open-domain summaries given varied sets of input triples. Finally, the latter two seek to evaluate whether our approach can be extended to other languages that either lack training data (i.e. Esperanto) or introduce further grammatical and syntactical complexities compared to English (i.e. Arabic).

We extracted the Wikipedia summaries and their corresponding DBpedia triples (i.e. for D1 and D3) from the Mapping-based Objects[1] (DB1) and Literals[1] DBpedia dataset (DB2),

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

retaining only summaries for which an infobox exists. For the datasets that leverage Wikidata (i.e. D2, M1 and M2), we used the Wikidata truthy dumps[2] (WD1) and we kept only items for which Wikidata triples exist. For all corpora, the relevant Wikipedia summaries were extracted using the Long Abstracts[1] DBpedia datasets (i.e. DB_{en} for the English summaries, and DB_{ar} and DB_{eo} for the ones in Arabic and Esperanto respectively).

In addition to the above datasets, we also leverage two DBpedia datasets: (i) the Instance Types[1] (DB3) and (ii) the Genders[1] (DB4) 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. 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 along with their intermediate components are enumerated below:

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

- English Long Abstracts[1] DBpedia dataset (DB_{en})
- Mapping-based Objects[1] DBpedia dataset (DB1)
- Mapping-based Literals[1] DBpedia dataset (DB2)
- Instance Types[1] DBpedia dataset (DB3)
- Genders[1] DBpedia dataset (DB4)

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

- English Long Abstracts[1] DBpedia dataset (DB_{en})
- Wikidata truthy dumps[2] (WD1)
- Instance Types[1] DBpedia dataset (DB3)

D3 DBpedia triples aligned with Wikipedia summaries; its intermediate components are listed below:

- English Long Abstracts[1] DBpedia dataset (DB_{en})

[2] https://dumps.wikimedia.org/wikidatawiki/entities

- Mapping-based Objects[1] DBpedia dataset (DB1)
- Mapping-based Literals[1] DBpedia dataset (DB2)
- Instance Types[1] DBpedia dataset (DB3)
- Genders[1] DBpedia dataset (DB4)

M1 Wikidata triples aligned with Arabic Wikipedia summaries that has been formed from:

- Arabic Long Abstracts[1] DBpedia dataset (DB_{ar})
- Wikidata truthy dumps[2] (WD1)

M2 Wikidata triples aligned with Esperanto Wikipedia summaries that has been formed from:

- Esperanto Long Abstracts[1] DBpedia dataset (DB_{eo})
- Wikidata truthy dumps[2] (WD1)

In the following sections, we describe the pre-processing steps that we carried out to bring both the textual summaries and the knowledge base triples of each corpus in the desired format.

4.2 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."[3], 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 (Daiber et al., 2013), 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.

[3] https://en.wikipedia.org/wiki/Barack_Obama

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) (Bird et al., 2009). We retained the first introductory sentence in the case of M1 and M2 (Lebret et al., 2016; Chisholm et al., 2017), and the first two sentences, in the case of D1, D2 and D3, 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 the most frequent tokens (i.e. either words or entities) of the summaries that are aligned with the DBpedia and Wikidata triples. 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[4] (Lebret et al., 2016).

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 (Graves, 2013; Cho et al., 2014; Sutskever et al., 2014; Bahdanau et al., 2014; Luong et al., 2015a). 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. In order to circumvent this, we propose three different approaches: (i) the *property-type placeholders*, (ii) the *property placeholders*, and (iii) a realisation mechanism on top of a pointer-generator network. More details regarding the above are provided in their respective sections (see Section 5.1.3 and 6.1.1).

4.3 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

[4] A slightly different strategy has been followed for the experiments using our pointer-generator architecture (cf. Section 7.2).

TABLE 4.1: 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>

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 (Lebret et al., 2016). 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 4.1 presents an example of our date encoding approach.

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 (i.e. for $D1$ and D3 datasets), and the Wikidata truthy dump in Wikidata's case (i.e. for $D2$, M1[5] and M2[5] datasets). 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. In the case of the datasets that are based on DBpedia triples, this results in 450 and 609k unique predicates and entities in D1's case and in 1124 and 1723k unique predicates and entities respectively in D3's case. Equivalently, D2, M1 and M2 consist of (i) 378 and 278k, (ii) 1021 and 355k, and (iii) 965 and 352k, unique predicates and entities respectively.

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.

[5]DBpedia Spotlight is not available in Arabic and Esperanto. Consequently, for the M1 and M2 corpora we used an approach based on keyword-matching in order to identify realisation of entities in the text. Further details are provided in Section 4.4.3.

TABLE 4.2: The left and middle columns display the distribution of the 10 most common predicates and entities in the DBpedia triples that have been allocated to the D1 dataset. The right column depicts the distribution of the 10 most common entities in the Wikipedia summaries as they have been identified with DBpedia Spotlight.

Predicates In Triples		**Entities In Triples**		**Entities In Summaries**	
Predicate	%	Entity	%	Entity	%
`dbo:birthDate`	12.43	`dbr:United_States`	0.49	`dbr:United_States`	2.82
`dbo:birthPlace`	10.67	`dbr:England`	0.19	`dbr:Actor`	2.14
`dbo:careerStation`	5.47	`dbr:United_Kingdom`	0.14	`dbr:Association_-football`	1.02
`dbo:deathDate`	5.11	`dbr:France`	0.14	`dbr:Politician`	0.97
`dbo:occupation`	5.06	`dbr:Canada`	0.12	`dbr:Singing`	0.90
`dbo:team`	4.18	`dbr:India`	0.11	`dbr:United_Kingdom`	0.59
`dbo:deathPlace`	3.51	`dbr:Actor`	0.10	`dbr:England`	0.58
`dbo:genre`	3.22	`dbr:Italy`	0.10	`dbr:Writer`	0.53
`dbo:associatedBand`	2.85	`dbr:London`	0.10	`dbr:Canada`	0.50
`dbo:associated-MusicalArtist`	2.85	`dbr:Japan`	0.09	`dbr:France`	0.49

TABLE 4.3: The left and middle columns display the distribution of the 10 most common predicates and entities in the Wikidata triples that have been allocated to the D2 dataset. The right column depicts the distribution of the 10 most common entities in the Wikipedia summaries as they have been identified with DBpedia Spotlight.

Predicates In Triples		**Entities In Triples**		**Entities In Summaries**	
Predicate	%	Entity	%	Entity	%
`wikidata:P569` (place of birth)	14.15	`wikidata:Q5` (human)	3.96	`wikidata:Q30` (United States of America)	3.20
`wikidata:P106` (occupation)	11.63	`wikidata:Q6581097` (male)	3.27	`wikidata:Q33999` (actor)	1.56
`wikidata:P31` (instance of)	8.29	`wikidata:Q30` (United States of America)	1.13	`wikidata:Q82955` (politician)	1.02
`wikidata:P21` (sex or gender)	7.92	`wikidata:Q6581072` (female)	0.70	`wikidata:Q21` (England)	0.87
`wikidata:P570` (date of death)	7.58	`wikidata:Q145` (United Kingdom)	0.44	`wikidata:Q145` (United Kingdom)	0.85
`wikidata:P27` (country of citizenship)	6.75	`wikidata:Q82955` (politician)	0.42	`wikidata:Q27939` (singing)	0.79
`wikidata:P735` (given name)	6.53	`wikidata:Q1860` (English)	0.39	`wikidata:Q36180` (writer)	0.71
`wikidata:P19` (place of birth)	5.20	`wikidata:Q33999` (actor)	0.36	`wikidata:Q2736` (association football)	0.68
`wikidata:P5` (member of sports team)	2.64	`wikidata:Q36180` (writer)	0.24	`wikidata:Q183` (Germany)	0.61
`wikidata:P69` (educated at)	2.58	`wikidata:Q177220` (singer)	0.20	`wikidata:Q16` (Canada)	0.58

4.4 Aligned Corpora

Tables 4.2—4.6 present the distribution of the 10 most common predicates, and entities in the aligned datasets that have been built for the purposes of this thesis. The experiments in which these aligned corpora have been used are presented in the following chapters of this thesis.

TABLE 4.4: The left and middle columns display the distribution of the 10 most common predicates and entities in the Wikidata triples that have been allocated to the D3 dataset. The right column depicts the distribution of the 10 most common entities in the Wikipedia summaries as they have been identified with DBpedia Spotlight.

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

TABLE 4.5: The left and middle columns display the distribution of the 10 most common predicates and entities in the Wikidata triples that have been used in the M1 corpus. The right column depicts the distribution of the 10 most common entities in the Wikipedia summaries as they have been identified using our keyword-matching approach.

Predicates In Triples		**Entities In Triples**		**Entities In Summaries**	
Predicate	%	Entity	%	Entity	%
`wikidata:P31` (instance of)	12.59	`wikidata:Q5` (human)	1.49	`wikidata:Q805` (Yemen)	10.54
`wikidata:P17` (country)	5.87	`wikidata:Q6581097` (male)	1.24	`wikidata:Q794` (Iran)	6.04
`wikidata:P569` (date of birth)	4.82	`wikidata:Q486972` (human settlement)	0.86	`wikidata:Q30` (United States of America)	5.67
`wikidata:P131` (located in the administrative territorial entity)	4.76	`wikidata:Q30` (United States of America)	0.79	`wikidata:Q532` (village)	5.37
`wikidata:P106` (occupation)	4.20	`wikidata:Q805` (Yemen)	0.75	`wikidata:Q7432` (species)	3.88
`wikidata:P21` (sex or gender)	2.99	`wikidata:Q794` (Iran)	0.49	`wikidata:Q515` (city)	2.67
`wikidata:P54` (member of sports team)	2.92	`wikidata:Q532` (village)	0.48	`wikidata:Q937857` (association football player)	2.23
`wikidata:P150` (contains administrative territorial entity)	2.88	`wikidata:Q16521` (taxon)	0.37	`wikidata:Q33999` (actor)	1.54
`wikidata:P47` (shares border with)	2.59	`wikidata:Q7432` (species)	0.33	`wikidata:Q11424` (film)	1.14
`wikidata:P27` (country of citizenship)	2.41	`wikidata:Q2736` (association football)	0.28	`wikidata:Q79` (Egypt)	1.11

TABLE 4.6: The left and middle columns display the distribution of the 10 most common predicates and entities in the Wikidata triples that have been used in the M2 corpus. The right column depicts the distribution of the 10 most common entities in the Wikipedia summaries as they have been identified using our keyword-matching approach.

Predicates In Triples		**Entities In Triples**		**Entities In Summaries**	
Predicate	%	Entity	%	Entity	%
`wikidata:P31` (instance of)	9.54	`wikidata:Q6655` (UTC+01 : 00)	0.99	`wikidata:Q183` (Germany)	6.73
`wikidata:P47` (shares border with)	8.42	`wikidata:Q5` (human)	0.61	`wikidata:Q515` (city)	4.87
`wikidata:P150` (contains administrative territorial entity)	5.62	`wikidata:Q183` (Germany)	0.58	`wikidata:Q3863` (asteroid)	4.09
`wikidata:P131` (located in the administrative territorial entity)	5.10	`wikidata:Q6581097` (male)	0.54	`wikidata:Q747074` (comune of Italy)	3.98
`wikidata:P17` (country)	4.87	`wikidata:Q262166` (municipality of Germany)	0.39	`wikidata:Q2179` (asteroid belt)	3.94
`wikidata:P910` (topic's main category)	4.24	`wikidata:Q38` (Italy)	0.33	`wikidata:Q28` (Hungary)	1.85
`wikidata:P106` (occupation)	3.11	`wikidata:Q3863` (asteroid)	0.29	`wikidata:Q484170` (commune of France)	1.65
`wikidata:P2044` (elevation above sea level)	2.73	`wikidata:Q747074` (comune of Italy)	0.29	`wikidata:Q577` (year)	1.46
`wikidata:P421` (located in time zone)	2.68	`wikidata:Q2179` (asteroid belt)	0.28	`wikidata:Q214` (Slovakia)	1.32
`wikidata:P569` (date of birth)	2.44	`wikidata:Q16521` (taxon)	0.25	`wikidata:Q29` (Spain)	1.28

In the following sections we provide further details regarding the two corpora of biographies (i.e. D1 and D2), and the three open-domain corpora that consist of English, Arabic and Esperanto summaries (i.e. D3, M1 and M2) respectively.

4.4.1 Biographies

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 4.8). Their linguistic variability is also supported by the fact that our D1 and D2 datasets contain summaries (95^{th} 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 and regular structure allowing us to explore the strengths and limitations of our purely data-driven approach. Table 4.7 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.

TABLE 4.7: Distribution of the 10 most frequent DBpedia instance types of the main discussed entities of the two corpora of biographies, 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

We used PetScan[7] to collect a list of 1479k Wikipedia articles that have been curated by the WikiProject Biography[8]. Subsequently, we leveraged this list in order to retain only the relevant summaries from the English Long Abstracts[1] DBpedia dataset (DB_{en}) and align them with their corresponding triples from DB1 and DB2 in the case of D1, and WD1 in the case of D2.

Following the rest of the pre-processing steps that are described in Section 4.2 and 4.3 resulted in 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. Firstly, there are Wikipedia biographies without an infobox (i.e. and, thus, without any available triples in the Mapping-based Objects and Literals DBpedia datasets). Secondly, 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 (Section 4.3). 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. Table 4.8 provides statistics of the D1 and D2 corpora.

[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

TABLE 4.8: Statistics of the two corpora of biographies, D1 (i.e. based on DBpedia triples) and D2 (i.e. based on Wikidata triples). Average parameters are shown with standard deviations in brackets.

Parameter	**D1**	**D2**
Total # of Articles	256850	358908
Total # of Entities	609k	278k
Total # of Predicates	450	378
Avg. # of Triples (incl. Encoded Dates) per Article	10.68 (7.87)	12.09 (6.16)
Max. # of Alloc. Triples (incl. Encoded Dates) per Article	175	255
Avg. # of Tokens per Summary	41.30 (17.83)	39.90 (17.33)
Total # of Words In the Summaries	400k	500k
Total # of Annotated Entities In the Summaries (excl. the Main Entity `<item>`)	194k	222k

TABLE 4.9: Statistics of the D3 corpus. Average parameters are shown with standard deviations in brackets.

Parameter	**D3**
Total # of Articles	864862
Total # of Entities	1173k
Total # of Predicates	1124
Avg. # of Triples (incl. Encoded Dates) per Article	8.59 (6.33)
Max. # of Alloc. Triples (incl. Encoded Dates) per Article	175
Avg. # of Tokens per Summary	39.95 (21.05)
Total # of Words In the Summaries	1114k
Total # of Annotated Entities In the Summaries (excl. the Main Entity `<item>`)	475k

4.4.2 The D3 Corpus

We adopt the above methodology in order to build a dataset encompassing the entirety of Wikipedia rather than just the biographies. This corpus will allow us to train and evaluate our approach on the generation of open-domain Wikipedia summaries. Table 4.9 provides statistics of the D3 corpus. It should be noted that the total number of unique entities and predicates in the triples of D3 along with its vocabulary size are double the size of the biographical-oriented corpora.

4.4.3 Building Multilingual Corpora

We follow a similar approach for building our non-English corpora. Every entity and property in Wikidata is represented by a Uniform Resource Identifier (URI) which is the same across all the available languages (e.g. "Berlin" is `Q64` and "part of" is `P361`). Each URI is connected to labels in multiple languages (Kaffee et al., 2017). We leverage this cross-lingual nature of Wikidata in order to build two corpora Wikidata triples aligned with open-domain Wikipedia summaries.

Inspired by Lebret et al.; Chisholm et al. in the experiments that involve Arabic and Esperanto, we focus on generating single-sentence summaries. While as explained in Section 4.2 this effectively reduces the computational cost of the tasks, it also provides better quality guarantees. Data-driven approaches generally struggle to generate long snippets of text (Bahdanau et al., 2014; Wiseman et al., 2017). Since the goal of these experiments was to explore the usability of the automatically generated content by Wikipedia readers and editors (see Section 3.1.3), we decided not to sacrifice any potential quality of the generated content in favour of longer summaries. For each Wikipedia article, we extract and tokenize the first introductory sentence using a multilingual Regular-Expression Tokenizer from the NLTK toolkit (Bird et al., 2009). Afterwards, we retrieve the corresponding Wikidata item to this article and query all triples where this item appears as a subject or an object in the Wikidata truthy dump[2].

Due to the lack of reliable Named Entity Recognition (NER) tools for underserved languages, we rely on keyword-matching against Wikidata labels for the corresponding language. In order to increase our coverage, we use the global language fallback chain introduced by Wikimedia[9] to extend the set of labels of each entity in the corresponding language. This approach allows us to cover as many labels as possible and to overcome the lack of non-English labels in Wikidata (Kaffee et al., 2017).

The above approach, along with the fact that Wikidata is a language-independent knowledge base, allows us to create corpora in multilingual setting. This dataset aligns Wikidata triples with the first, introductory sentence of its corresponding Wikipedia articles. Table 4.10 provides statistics on the resultant M1 and M2 corpora.

4.5 Discussion

While the above approach allows us to build large data-to-text corpora in different languages, there are no guarantees that the information that exists in the triples does necessarily appear in the corresponding text, and vice versa. We have explored an alternate methodology for semi-automatically building high quality data-to-text corpora.

[9] https://meta.wikimedia.org/wiki/Wikidata/Notes/Language_fallback

TABLE 4.10: Statistics of the two non-English corpora, M1 (i.e. with the textual summaries in Arabic) and M2 (i.e. with the textual summaries in Esperanto). Average parameters are shown with standard deviations in brackets.

Parameter	**Arabic**	**Esperanto**
Total # of Articles	255741	126714
Total # of Entities	355k	352k
Total # of Predicates	1021	965
Avg. # of Triples (incl. Encoded Dates) per Article	8.10 (11.23)	11.23 (13.82)
Max. # of Alloc. Triples (incl. Encoded Dates) per Article	885	883
Avg. # of Tokens per Summary	27.98 (28.57)	26.36 (22.71)
Total # of Words In the Summaries	433k	324k
Total # of Annotated Entities In the Summaries (excl. the Main Entity `<item>`)	22k	18k

This methodology leverages crowdsourcing and introduces an automatic scoring function for the quality of the contribution to ensure better systematic alignment between text and data, and works as follows. Firstly, similarly to Section 4.2, we annotate a textual corpus in order to identify the realisations of entities that exist in a knowledge base. Secondly, we collect triples from the corresponding knowledge base that link the entities mentioned in a given sentence. Thirdly, we perform Semantic Sentence Simplification (S3) through a crowdsourcing task in which we aim to retain only the natural language elements that have correspondents in the allocated triples. Finally, we apply a novel automatic scoring function to keep the most relevant semantic simplifications of each sentence as a natural language expression of the corresponding set of collected triples. Table 4.11 presents the final statistics on the resultant corpus based on sentences from Wikipedia and MedlinePlus[10] using this methodology.

TABLE 4.11: Statistics on the resultant corpus using Semantic Sentence Simplification (S3) based on sentences from Wikipedia and MedlinePlus.

Parameter	**Wikipedia**	**MedlinePlus**
Total # of Sentences	600	450
Total # of of Triples	1119	766
Total # of Predicates	146	30
Total # of Words	13641	11167
Total # of Words In the Summaries After S3	9011	6854

[10] medlineplus.gov

This work has led to a peer-reviewed publication (Mrabet et al., 2016). While it departs from fully-automatic approaches (Lebret et al., 2016; Chisholm et al., 2017), which are similar to the methodology in Section 4.1, offering greater systematic guarantees between text and data, its initial resultant corpus did not offer the size nor the lexical variation (see Table 4.11) that would allow us to explore the main objective of this thesis, which is open-domain textual summaries generation. The fact that the approach relies on crowdsourcing raises also additional challenges should we opt to apply it in less popular languages. Consequently, the experiments that led to the creation of that corpus are not described in this thesis. For further details, we urge interested readers to refer to the corresponding published article (Mrabet et al., 2016).

4.6 Summary

In this chapter, we presented a fully-automatic, cross-lingual, approach for building large corpora of loosely aligned knowledge base triples with Wikipedia summaries. We believe that this methodology could be of great value to other research domains besides NLG, such as Relation Extraction and Question Answering. Tailoring our approach to the domain of Relation extraction, by excluding triples whose objects are not mentioned in the corresponding text, resulted also in a peer-reviewed conference publication (Elsahar et al., 2018).

In the subsequent chapters of this thesis, we focus our experiments on the five corpora that were built using our fully-automatic approach, D1, D2, D3, M1 and M2.

Chapter 5

Neural Wikipedian: Generating Biographies from Knowledge Base Triples

An idealised example of our NLG task is presented in Table 5.1; our system takes as an input a set of triples about *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. This essentially means that the information that is presented in the text is a subset of the total knowledge that is enclosed in the input set of triples (e.g. the fact that *Walt Disney* was born on "1901-12-05" is not realised in the generated summary).

Our approach is inspired by the general encoder-decoder framework (Cho et al., 2014; Sutskever et al., 2014) with multi-gated Recurrent Neural Network (RNN) variants, such as the Gated Recurrent Unit (GRU) (Cho et al., 2014) and the Long Short-Term Memory (LSTM) cell (Hochreiter and Schmidhuber, 1997). 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 the 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). We explore a set of different approaches that enable our models to verbalise entities from the input set of triples in the generated text.

In this chapter, we focus on the generation of two-sentence biographies given a set of input triples. This task is most similar to recent work by Lebret et al. and Chisholm

TABLE 5.1: 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.

et al., who both employ adaptations of the encoder-decoder framework to generate the first sentence of a Wikipedia biography (Lebret et al., 2016; Chisholm et al., 2017). Lebret et al. propose a system that relies on slot-value templating in order to generate 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 4.3, 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. For instance the first two triples of Table 5.1 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.

The experiments and results that are presented in this chapter have been published as a journal article (Vougiouklis et al., 2018a). In the following sections, we describe in detail our proposed model, and the experiments that we conducted using the D1 and D2 corpora for its training and evaluation.

5.1 The Model

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 $f_1, f_2, \ldots, f_E$:

$$p(y_1, \ldots, y_T | f_1, f_2, \ldots, f_E) = \prod_{t=1}^{T} p(y_t | y_1, \ldots y_{t-1}, F) \ . \tag{5.1}$$

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 5.1. Note that since *bias* terms can be included in each weight-matrix multiplication, they are not explicitly shown in the equations of this section (Bishop, 1995).

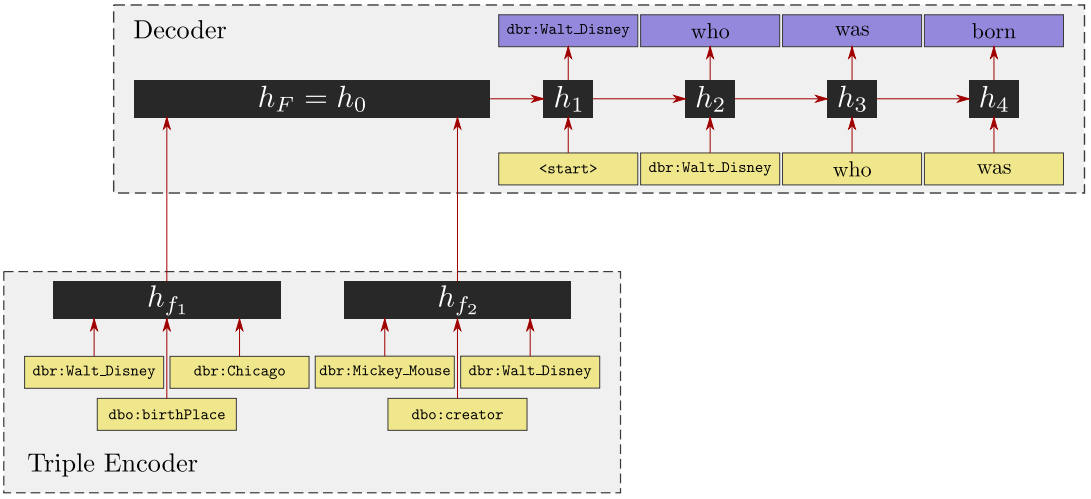


FIGURE 5.1: The architecture of our model based on the encoder-decoder framework. The triple encoder computes a vector representation, h_{f_1} and h_{f_2}, for each one of the two input triples. Subsequently, the decoder uses the concatenation of the two vectors, $[h_{f_1}; h_{f_2}]$ to initialise the 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 the 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.

5.1.1 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[1] vector representations of the respective subject, predicate and object of

[1]One-hot is a vector that contains a 1 at the index of a particular token (i.e. entity or predicate on the encoder's side, or regular word, entity or surface form of an entity on the decoder's side) in the vocabulary with all the other values set to zero.

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:

$$\tilde{h}_{f_i} = [\mathbf{W}_{\mathbf{x}\to\tilde{\mathbf{h}}} s_i; \mathbf{W}_{\mathbf{x}\to\tilde{\mathbf{h}}} p_i; \mathbf{W}_{\mathbf{x}\to\tilde{\mathbf{h}}} o_i] \ , \tag{5.2}$$

$$h_{f_i} = \text{ReLU}(\mathbf{W}_{\tilde{\mathbf{h}}\to\mathbf{h}} \tilde{h}_{f_i}) \ , \tag{5.3}$$

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 predicates and entities 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.

5.1.2 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. As explained in latter part of Section 2.2.1.2, the main advantage of using multi-gated units is their ability to process information from much longer sequences compared to the *simple* RNN (Chung et al., 2014). We adopt the architectures of the LSTM cell and the GRU from Zaremba and Sutskever; Cho et al. respectively.

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 (Cho et al., 2014; Sutskever et al., 2014). 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:

$$\tilde{h}_F = [h_{f_1}; h_{f_2}; \ldots; h_{f_{E-1}}; h_{f_E}] \ , \tag{5.4}$$

$$h_F = \mathbf{W}_{\mathbf{h_F}\to\mathbf{h_0^1}} \tilde{h}_F \ , \tag{5.5}$$

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 $t \in [1 \dots T]$ and layer depth $l \in [1 \dots L]$. 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. At each timestep t, h_t^l is computed as follows:

$$h_t^l = \begin{cases} \text{LSTM}(h_{t-1}^l, h_t^{l-1}) & \text{LSTM-based decoder} \\ \text{GRU}(h_{t-1}^l, h_t^{l-1}) & \text{GRU-based decoder} \end{cases} \tag{5.6}$$

We adopt the architecture for the LSTM cells and GRUs from (Zaremba and Sutskever, 2014) and (Cho et al., 2014) respectively. The functionality of LSTM cells and the GRUs is described in details in Sections 2.2.1.3 and 2.2.1.4 respectively.

5.1.3 Property-Type Placeholders

Conventional systems based on neural networks when applied to other tasks related to NLG involving surface realisation, such as Machine Translation (Sutskever et al., 2014; Luong et al., 2015b) or Question Generation (Dong and Lapata, 2016; Serban et al., 2016), are incapable of learning high quality vector representations for the infrequent tokens (i.e. either words or entities) in their training dataset. Inspired by other multi-placeholder approaches that have been proposed by Serban et al.; Dong and Lapata, 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.

5.1.4 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:

$$p(y_t|y_1, \dots y_{t-1}, F) = \text{softmax}(\mathbf{W_y} h_t^L) \ , \tag{5.7}$$

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[2] 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:

$$\text{cost} = -\sum_{t=1}^{T} \log \, p(y_t|y_1, \dots y_{t-1}, F) \ . \tag{5.8}$$

Consequently, our model tries to minimise the above cost function. This non-convex optimisation problem is solved using Back-Propagation (Rumelhart et al., 1986) with a dynamic learning rate update provided by the RMSProp[3] algorithm.

[2] 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 a set of parameters θ. It is defined as: $H(X) = -\sum q(y^{(i)}) \log p(y^{(i)}|x^{(i)}, \theta)$.

[3] 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 (Tieleman and Hinton, 2012).

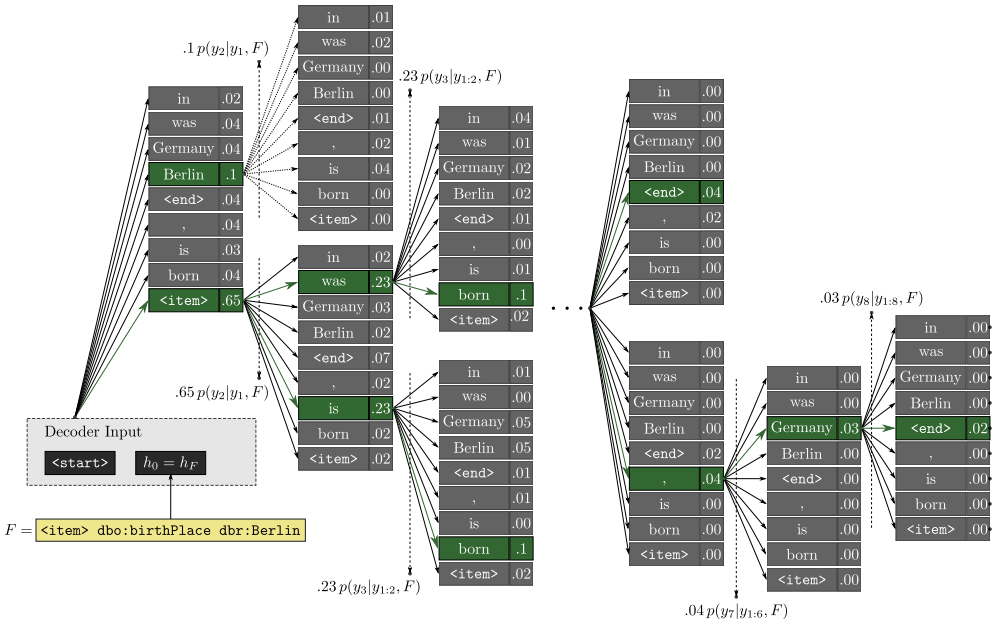


FIGURE 5.2: A simplified example of a beam-search decoder with a beam B of size 2 and target vocabulary size $|X|$ equal to 9. After the vector representation h_F for the whole triples' set is computed, it is provided as input to the decoder along with the `<start>` token. The scores at the right-hand 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 4.2 and 4.3) 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.

5.1.5 Generating Summaries

During testing, our goal is to find:

$$\mathbf{y}^* = \arg\max_{\mathbf{y}} \sum_{t=1}^{T} \log\, p(y_t|y_1, \ldots y_{t-1}, F) \ , \tag{5.9}$$

where $\mathbf{y}^*$ is the optimal summary computed by the model. Recall from Eq. 5.7 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 (Rush et al., 2015). 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 (Rush et al., 2015).

A compromise between a strictly-greedy decoding algorithm and Viterbi is to adopt a beam-search decoder (Sutskever et al., 2014; Rush et al., 2015), 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 5.2.

5.2 Dataset Preparation

We train and evaluate our model on the corpora of biographies, D1 and D2. 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.

5.2.1 Modelling the Textual Summaries

We used a fixed vocabulary of 30000 and 33000 of the most frequent tokens (i.e. either words or entities) of the summaries that are aligned with the DBpedia and Wikidata triples. Each summary is augmented with the respective start-of-summary `<start>` and end-of-summary `<end>` tokens (Sutskever et al., 2014; Luong et al., 2015a).

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.

5.2.1.1 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 4.2).

5.2.1.2 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 j-th *surface form tuple* of the entity $k_d \in K$ as: $u_j^{k_d} = (k_d, g_j^{k_d}) : k_d \in K$, where $g_j^{k_d}$ is the j-th surface form that is associated with the entity k_d. Similarly to Section 5.2.1.1, 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 verbalised accordingly (as *actor* or *actress*) based on the gender of the main entity of interest.

TABLE 5.2: Statistics regarding the initial and the training versions of the two corpora of biographies, 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	217
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

5.2.2 Modelling the Input Triples

The shared, fixed input dictionary of all subjects, predicates and objects was formed as follows. 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 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:

$$\lfloor E_{\min} + 0.25\sigma_E \rfloor \leq E \leq \left\lfloor \overline{E} + 1.5\sigma_E \right\rfloor \quad . \tag{5.10}$$

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 4.3). In case their number still exceeds the limit, we retain only the first ones until the threshold is reached.

Table 5.2 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 5.3. Details about the two different types of summaries ("Summary w/ URIs" and "Summary w/ Surface Form Tuples") are provided in Sections 5.2.1.1 and 5.2.1.2 respectively.

5.3 Experiments

Both, D1 and D2, 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[4] 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 baselines are available at `github.com/pvougiou/Neural-Wikipedian`.

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

5.3.1 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 (Triples2LSTM). 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, 5.2.1.1) and once as a mixture of words and surface form tuples (w/ Surf. Form Tuples, 5.2.1.2).

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 5.4 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: $[E_{\max} \cdot m, m]$, 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 (Ioffe and Szegedy, 2015). 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 (Ioffe and Szegedy, 2015). The networks are trained with mini-batch RMSProp with an initial learning rate value of 0.002. RMSProp was found to work better than Stochastic Gradient Descent, RMSProp and AdaGrad by resulting into lower error (i.e. perplexity) on the validation sets. Each update is computed using a mini-batch of 85 dataset instances. An l_2 regularisation of 0.1 term over each network's parameters (weights) is also included in the cost function in order to prevent overfitting (Wen et al.,

2015, 2016). 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.

TABLE 5.3: 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, it 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>`

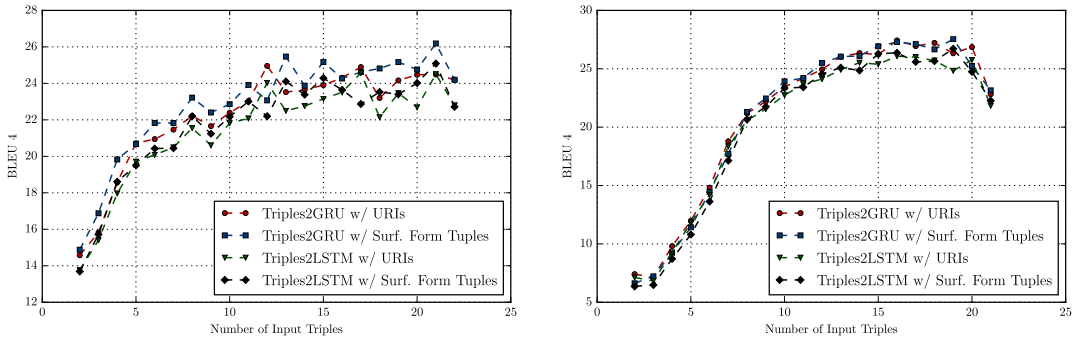
Table 5.4: Training Hyperparameters of the Systems

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
# of Training Epochs[a]	12	16	12	16	12	15	13	22

[a]The epoch at which the model converges to the lowest possible validation error. After this epoch, the error on the validation set either does not improve further or it increases, and, thus, the model is overfitting.

TABLE 5.5: Automatic evaluation with perplexity (lower values are better), and the BLEU, METEOR and $ROUGE_L$ metrics (higher values are better) on the validation and test sets. 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**		**METEOR**		**$ROUGE_L$**	
	Valid.	Test	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)	18.191 (±.03)	18.261 (±.05)	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)	**34.041** (±.01)	**34.161** (±.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	26.734	26.681	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	**27.183**	27.226	**47.092**	47.100
Triples2LSTM on D1 w/ Tuples	19.171	19.086	40.679	40.763	30.809	30.904	25.234	25.344	21.287	21.393	24.752	24.885	44.973	45.143
Triples2GRU on D1 w/ Tuples	20.164	20.007	**41.350**	**41.457**	**31.877**	**31.991**	**26.387**	**26.510**	**22.419**	**22.531**	26.250	26.506	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)	18.660 (±.02)	18.677 (±.04)	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)	**33.836** (±.00)	**33.898** (±.00)	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	25.575	25.655	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	**26.057**	**26.083**	47.577	47.761
Triples2LSTM on D2 w/ Tuples	20.779	20.403	40.660	40.604	31.158	31.144	25.776	25.783	21.874	21.898	25.462	25.597	47.031	47.140
Triples2GRU on D2 w/ Tuples	21.493	21.200	**41.527**	41.566	**32.072**	**32.097**	**26.645**	**26.673**	**22.679**	**22.708**	25.451	25.576	**47.979**	**48.100**



(A) D1 (B) D2

FIGURE 5.3: Performance of our models with the BLEU 4 metric across the different number of input triples on D1 (a) and D2 (b).

5.3.2 Automatic Evaluation

We use perplexity, BLEU (Papineni et al., 2002), METEOR (Lavie and Agarwal, 2007), and ROUGE (Lin, 2004) on the validation and test set of each corpus. Perplexity, BLEU 1, BLEU 2, BLEU 3, BLEU 4, METEOR, and ROUGE_L results are reported in Table 5.5.

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 (see Section 3.2.1). We consider this a particularly strong baseline due to the fact that Wikipedia biographies tend to follow the same text structure. For each triple set on the validation and test set, the random system generates a response by randomly selecting a Wikipedia summary from our training set. Secondly, we use the KenLM toolkit (Heafield et al., 2013) in order to build a 5-gram Kneser-Ney (KN+) language model (see Section 3.2.2). 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. The results are illustrated in Table 5.5. Note that our scores are not directly comparable to similar works by Lebret et al. and Chisholm et al. that focus on English biographies. Both Lebret et al. and Chisholm et al. constrain the generative process to only the first sentence of a Wikipedia summary. Nonetheless, generating shorter snippets of text simplifies the task and results in higher automatic evaluation scores (Bahdanau et al., 2014; Wiseman et al., 2017) (see "Putting the Scores Into Perspective" part of Section 3.1.1) compared to those of our more elaborate summaries.

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. The slightly higher METEOR scores of the w/ URIs systems compared to the w/ Surf. Form Tuples ones shows that they occasionally generate text that differs from the empirical summaries due to synonymic and morphological variations. We believe that this along with lower BLEU and ROUGE might also be indicative of text with minor grammatical or syntactical errors that, however, addresses the entities from the triples correctly. We are exploring this hypothesis in the subsequent human evaluation (see Section 5.3.3).

Interestingly the KN+ is scoring higher METEOR scores than our neural network-based systems, while its BLEU and ROUGE scores are much lower than the competition. This shows that in the context of a single domain corpus with limited structural paradigms, the KN+ baseline is capable of capturing the uni-grams (METEOR is a uni-gram based metric) that have high probability of occurring in the empirical summaries. However, the fact that is not fully conditioned on the input triple set is reflected in its low BLEU scores (i.e. n-grams precision). We show also in a subsequent chapter (see Chapter 7) that its METEOR performance is much lower when faced with a more challenging, multi-domain corpus (i.e. D3).

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 5.3 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, ROUGE and METEOR 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 (Cho et al., 2014; Sutskever et al., 2014). Our generative task is more challenging since it consists of learning 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 proposed architectures serves the needs of our task the best.

TABLE 5.6: Examples of textual summaries that are generated by our proposed systems given an unknown set 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>

5.3.3 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

TABLE 5.7: Further examples of textual summaries that are generated by our proposed systems given an unknown set 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: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>

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, triple sets with fewer triples usually lack enough information for our systems to generate a summary (cf. Section 5.3.2). 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,

TABLE 5.8: 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, _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)

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, (ii) coverage (triples whose information is mentioned either implicitly or explicitly in the text), (iii) contradiction (information that exists in the sentence but it conflicts with one or more of triples from the input set), and (iv) additional information (the portion of the number of triples to which potential additional information in the text can be interpreted with respect to the total number of input triples). These criteria are described in detail in Section 3.1.2. The performance of our systems against the human evaluation criteria is presented in Table 5.9.

The results of the human evaluators are in agreement with the results of the automatic evaluation with the BLEU, ROUGE and METEOR metrics (i.e. Section 5.3.2). For the same training setup (i.e. w/ URIs or surface form tuples) and with the 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 contradicting information. Since they are more reluctant to introduce out-of-context

TABLE 5.9: The average rating of our systems against the human evaluation criteria. For fluency and coverage the higher the score the better; for contradiction and additional information, the lower the score the better. The results are reported in the "mean (± standard deviation)" format.

Model	**Fluency**	**Coverage**	**Contradiction**	**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)

information in the text, their generated textual content is better aligned with the input triples.

Interestingly, all the models that were trained on D2 scored consistently better in terms of the portion of input triples that are summarised in the text (i.e. coverage). 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).

5.4 Discussion

Two examples of textual summaries that are generated by our models are shown in Table and 5.6 and 5.7. 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, are we 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 5.3.3).

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. a 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 5.8 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 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 triple set (see Section 5.1.3). 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 in the input that share the same property. An alternate method would be to adapt the architecture of a pointer-generator network, similar to (See et al., 2017), that would theoretically be able to directly copy from the source a numerical value in the generated text. We investigate this approach in Chapter 7.

5.5 Conclusion

We propose an end-to-end trainable system that can generate a textual summary from triples. Our approach does not require any manually defined templates Using the D1 and D2 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 during 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 fluency of the generated summaries compared to a purely deterministic replacement of the generated entities' URIs as reported by our human evaluators. 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.

In the following chapter, we adapt the Triples2GRU architecture equipped with surface form tuples in order to generate textual summaries in two underserved Wikipedia languages: (i) Esperanto and (ii) Arabic. We investigate the performance on this under-resourced domain using automatic evaluation metrics and community studies through which we seek to examine the usability of the automatically generated text in those two Wikipedias.

Chapter 6

Learning to Generate Wikipedia Summaries for Underserved Languages

An important aspect of the ability to generate coherent text that addresses a set structured records, is that it can be used to improve the coverage of Wikipedia, or other collaborative knowledge bases, which lack content in the less popular languages (Chisholm et al., 2017). Given the promising results of our systems in the generation of biographies, we wished to explore the applicability of our approach to languages that do not offer the same training data abundance as English. We leverage the cross-lingual nature of Wikidata (see Section 4.4.3), and we adapt the Triples2GRU architecture equipped with surface form tuples (which was found to outperform the other alternatives in the previous chapter's experiments) in order to generate textual summaries in two underserved Wikipedia languages: (i) Esperanto and (ii) Arabic. Esperanto is an an easily acquired, artificially created language for which we severely lack training data (see number of available articles in Table 6.1). Our assumption is that the simple syntax and morphology of Esperanto (Li, 2006) compensates for the lack of training data, making it a suitable starting point for exploring the challenges of this task. On the contrary, Arabic, due to its grammatical and syntactical complexities along with its significantly larger vocabulary, is much more challenging to work with (Habash and Rambow, 2005; Badr et al., 2008). As shown in Table 6.1 both Arabic and Esperanto suffer a significant lack of content and active editors compared to the English Wikipedia which is currently the biggest one in terms of number of articles. Our approach can not only dramatically enhance the coverage of the impoverished Wikipedias but also provide their corresponding editors with a "starting point" to write their article.

In order to evaluate the usability of the automatically generated content by both the readers and the editors of these involved Wikipedias, we propose a novel evaluation framework. In addition to reporting the performance of our approach against the competing baselines using automatic evaluation metrics (see Section 3.1.1), we conducted

TABLE 6.1: Wikipedia page statistics along with the total number of unique words (i.e. vocabulary size) in Arabic, Esperanto and English.

Parameter	Arabic	Esperanto	English
Total # of Articles	541k	242k	5484k
Total # of Active Users	7818	2849	129237
Vocab. Size	2.2M	1.5M	2.0M

two community studies for each investigated Wikipedia language, one for its readers and one for its editors.

In this chapter, we first describe the alteration to the original Triples2GRU w/ Surf. Form Tuples (see Section 5.1 and 5.2.1.2), and then we present the necessary training details and the results of the automatic and human evaluation on the M1 and M2 corpora.

It should be noted that the experiments that are presented in this chapter are part of a collaborative work, the results of which have been published in the following two papers:

- Lucie-Aimée Kaffee[†], Hady Elsahar[†], **Pavlos Vougiouklis**[†], Christophe Gravier, Frédérique Laforest, Jonathon Hare, and Elena Simperl. Mind the (Language) Gap: Generation of Multilingual Wikipedia Summaries from Wikidata for ArticlePlaceholders. In *Proceedings of 15th International Conference, ESWC 2018*, Heraklion, Crete, Greece. Springer International Publishing.
- Lucie-Aimée Kaffee[†], Hady Elsahar[†], **Pavlos Vougiouklis**[†], Christophe Gravier, Frédérique Laforest, Jonathon Hare, and Elena Simperl. Learning to Generate Wikipedia Summaries for Underserved Languages from Wikidata. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, New Orleans, Louisiana. Association for Computational Linguistics.

The original idea for the two community studies with which we explored the performance of the proposed approach in generating a textual summary in a non-English language was originally conceived by Lucie-Aimée Kaffee, who as part of her PhD investigates the multi-linguality in collaborative knowledge bases.

6.1 Model

We use the Triples2GRU architecture that has been described in detail Section 5.1. The architecture consists of a feed-forward architecture (that we refer to as Triple Encoder

[†]The authors contributed equally to this work.

TABLE 6.2: The Triples2GRU architecture takes as an input a set of RDF triples about *Floridia*, whose either subject or object is related to the item of Floridia. The Processed Summary and Triples present the format of the respective summary and triples that is used for the training of the neural architecture. Each entity in the triples is stored along with its label (e.g. "komunumo de Italio" and "Floridia"). These are used at a post-processing step in order for any generated special tokens (e.g. `<item>` or property placeholders) to be replaced. As explained in Sections 4.2 and 4.3, any reference to the main discussed entity is replaced by the `<item>` token both in the text and the triples. When a rare entity in the text is matched to an entity of the corresponding triples' set, then it is replaced by the token of the predicate of the relevant triple (e.g. "Italio" is replaced by the property placeholder `[[P17]]`). Any occurrence of an infrequent entity in the triples is replaced by the special `<resource>` token (e.g. `Q38` is replaced by the `<resource>` token before the triples are inputted to the neural architecture).

Triples	f_1 :	`Q490900` (Floridia)	`P17` (ŝtato)	`Q38` (Italio)
	f_2 :	`Q490900` (Floridia)	`P31` (estas)	`Q747074` (komunumo de Italio)
	f_3 :	`Q30025755` (Floridia)	`P1376` (ĉefurbo de)	`Q490900` (Floridia)
Processed Triples	f_1 :	`<item>` (Floridia)	`P17` (ŝtato)	`<resource>` (Italio)
	f_2 :	`<item>` (Floridia)	`P31` (estas)	`Q747074` (komunumo de Italio)
	f_3 :	`Q30025755` (Floridia)	`P1376` (ĉefurbo de)	`<item>` (Floridia)
Original Wikipedia Summary	Floridia estas komunumo de Italio.			
Processed Summary	`<start>` `<item>` estas komunumo de `[[P17]]`. `<end>`			

in Section 5.1.1) that encodes an input set of triples into a vector of fixed dimensionality, and an RNN that uses GRUs (Cho et al., 2014) to generate the textual summary one token at a time. In the experiments of the previous chapter, the surface form tuples mechanism (see Secion 5.2.1.2) significantly enhanced the fluency of the systems compared to a deterministic replacement of the generated entities' URIs. Consequently, we also use this mechanism in order to replace textual mentions of entities in the text with their corresponding surface form tuples.

Table 6.2 presents an example of how a set on input triples and its corresponding summary are pre-processed in order to be used for the training of our neural architecture.

6.1.1 Property Placeholders

The systems that we used for the generation of biographies leverage instance-type-related information from DBpedia in order to address rare or unseen entities in the text. However, in this chapter we wish to tailor our approach to the Wikidata requirements without using external information from other knowledge bases, and as a result, we explore a much broader solution. In the current setup, surface form tuples in the text that correspond to rare entities that are found to participate in relations in the input triple set are replaced by the token of the property of the matched relationship. We refer to those placeholder tokens (Serban et al., 2016; Dong and Lapata, 2016) as *property placeholders*. Similarly to the property-type placeholders (see Section 5.1.3), the property

placeholders are appended to the target dictionary of the generated summaries. This approach does not rely on the assumption that unseen triples will adopt to the same pattern of properties and entities instance types pairs as the ones that have been used for training, and while a potential accuracy is sacrificed by not appending any instance-type-related information to the property placeholder tokens, it should generalise better than the property-type placeholders.

In Table 6.2, `[[P17]]` in the processed summary is an example of a property placeholder. In case it is generated by our model, it is replaced with the label of the object of the triple with which they share the same property (i.e. `Q490900` (Floridia) `P17` (ŝtato) `Q38` (Italio)). When all the tokens of the summary are sampled, each property placeholder that is generated is mapped to the triple with which it shares the same property, and is subsequently replaced with the textual label of the entity. We randomly choose an entity, in case there are more than one triple with the same property in the input triple set.

6.2 Dataset Preparation

We train and evaluate our approach on the M1 and M2 corpora that are described in detail in Section 4.4.3. We used a fixed target vocabulary that consisted of the 15000 and 25000 of the most frequent tokens (i.e. either words or surface form tuples of entities) of the summaries in Arabic and Esperanto respectively. Using a larger size of target dictionary in Arabic is due to its greater linguistic variability; the Arabic vocabulary is 47% larger than Esperanto vocabulary (cf. Table 6.1).

Similarly to Section 5.2.2, we limit the maximum number of triples $E_{\max}$ that are allocated to each summary according to Eq 5.10 in order to contain the space complexity of the task. In case the number of triples of a set exceeds this threshold, we follow the same approach as in Section 5.2.2 to exclude any redundant ones. In case the size of the set is still greater than $E_{\max}$, we pick the first ones, but unlike in Section 5.2.2, we prioritise those whose entities have been identified in the text.

We replaced any rare entities in the text that participate in relations in the aligned triple set with the corresponding property placeholder of the upheld relations. We include all property placeholders that occur at least 20 times in each training dataset. Subsequently, the dictionaries of the Esperanto and Arabic summaries are expanded by 80 and 113 property placeholders respectively. In case the rare entity is not matched to any subject or object of the set of corresponding triples, it is replaced by the special `<resource>` token.

Each summary is augmented with the respective start-of-summary `<start>` and end-of-summary `<end>` tokens (Sutskever et al., 2014; Luong et al., 2015a).

6.3 Experiments

Both, M1 and M2, datasets are split into training, validation and test, with respective portions of 85%, 10%, and 5%. The neural network models are implemented using the Torch[4] 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, the competing baselines and the results of the automatic and human evaluation are available at `github.com/pvougiou/Wikidata2Wikipedia` and `github.com/pvougiou/Mind-the-Language-Gap`.

6.3.1 Training Details

We train the Triples2GRU / Surf. Form Tuples model in two different setups, one in which it is equipped with the property placeholders, and one without. In the latter scenario, all textual mentions to rare entities in the text are replaced by the special `<resource>` token. In the experiments that are presented in this Chapter, we refer to the first system as Triples2GRU w/ Property Type Placeholders, and the second as Triples2GRU. Please note that neither setup leverages the property-type placeholders that have been proposed in Chapter 5. For each dataset, M1 and M2, we train each different on the task of generating textual summaries in Arabic and Esperanto respectively.

For the decoder, we use 1 layer of GRUs. We set the dimensionality of the decoder's hidden state to 500 in Esperanto and 700 in Arabic. We initialise all parameters with random uniform distribution between -0.001 and 0.001, and we use Batch Normalisation before each non-linear activation function and after each fully-connected layer (Ioffe and Szegedy, 2015) on the encoder side. During training, the model tries to learn those parameters that minimise the sum of the negative log-likelihoods of a set of predicted summaries. The networks are trained using mini-batch of size 85. The weights are updated using Adam (Kingma and Ba, 2014) (i.e. it was found to work better than Stochastic Gradient Descent, RMSProp and AdaGrad) with a learning rate of 10^{-5}. An l_2 regularisation term of 0.1 over each network's parameters is also included in the cost function as an overfitting countermeasure (Wen et al., 2015, 2016).

The networks converge[1] after the 9th epoch in the Esperanto case and after the 11th in the Arabic case. During evaluation and testing, we do beam search (see Section 5.1.5) with a beam size of 20, and we retain only the summary with the highest probability. We found that increasing the beam size resulted not only in minor improvements in terms of performance but also in a greater number of fully-completed generated summaries

[1]The epoch at which the model converges to the lowest possible validation error. After this epoch, the error on the validation set either does not improve further or it increases, and, thus, the model is overfitting.

(i.e. summaries for which the special end-of-summary <end> token is generated). Triple sets whose generated summaries are incomplete[2] (i.e. summaries for which the special end-of-summary <end> token is generated) have been excluded from the evaluation.

6.3.2 Automatic Evaluation

We use BLEU (Papineni et al., 2002), METEOR (Lavie and Agarwal, 2007), and ROUGE (Lin, 2004) on the validation and test set of each corpus. BLEU 1, BLEU 2, BLEU 3, BLEU 4, METEOR, and ROUGE_L results are reported in Table 6.3.

To demonstrate the effectiveness of our system, we compare it to three baselines. First, we use the KenLM toolkit (Heafield et al., 2013) in order to build a 5-gram Kneser-Ney (KN) language model. We provide two versions of this baseline, KN and KN+ (see Section 3.2.2). The first is trained on the raw textual summaries from each training set, and the second is equipped with the special tokens of the neural network-based systems (i.e. surface form tuples and <item> and property placeholders). During testing, similarly to the case of our neural network approaches, we use beam-search with a beam of size 20, to generate the 20 most probable summaries for each triple set in the validation and test sets. Secondly, we employ a MT baseline (see Section 3.2.4), with which we translate the first introductory sentence of the English Wikipedia article of each item in the validation and test set to Arabic and Esperanto. Lastly, we compare our approach to an IR baseline (see Section 3.2.3). For each set of triples in the validation and test sets, we perform k-NN to retrieve the closest vector from the training set, and output its corresponding summary. Similarly to the case of KN, we explore the performance of this baseline with and without including special tokens.

As displayed in Table 6.3, our approach shows a significant enhancement compared to the baselines across the majority of the evaluation metrics in both languages. We achieve at least an enhancement of at least 5.25 and 1.31 BLEU 4 score in Arabic and Esperanto respectively over the IR+, the strongest baseline. The MT baseline is not competitive. We attribute this result to the different writing styles across different Wikipedia languages. We believe that this finding also inhibits MT from being sufficient for Wikipedia document generation.

The introduction of the property placeholders to the Triples2GRU architecture enhances our performance further by 0.61−1.10 BLEU (using BLEU 4). In general, the inclusion of special tokens (i.e. surface form tuples and <item> and property placeholders) benefited the performance of all the competitive systems.

[2] Around $\leq$ 1% and 2% of the input validation and test triple sets in Arabic and Esperanto respectively led to the generation of summaries without the <end> token. We believe that this difference is explained by the limited size of the Esperanto dataset that increases the level of difficulty that the trained models (i.e. with or without property placeholders) to generalise on unseen data.

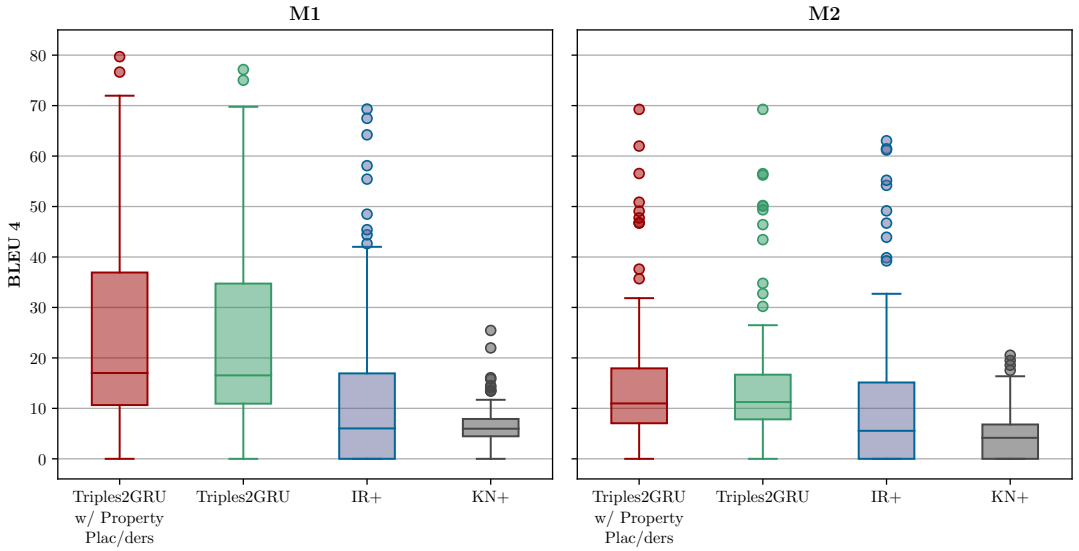


FIGURE 6.1: A box plot showing the distribution of BLEU 4 scores of all the systems for each category of generated summaries.

Furthermore, while our tasks consists in generating open-domain Wikipedia summaries, our performance in terms of BLEU is competitive with other similar approaches that constrain themselves to single-domain summaries (i.e. biographies) in English (Lebret et al., 2016; Chisholm et al., 2017). Both in Arabic and Esperanto, the systems equipped with property placeholders achieve BLEU 4 scores that are higher than the ones reported by Lebret et al.. The performance in Arabic, where we have significantly more training examples (see Table 4.10), is also very close to the one achieved by Chisholm et al. (around $40.50 - 41.00$ BLEU 4).

Generalisation Across Domains. In order to further investigate how well the systems generalise across different categories, we group the Wikipedia summaries from each dataset according to the instance type of their main entity (e.g. dbo:Village and dbo:SoccerPlayer). The instance type of each main entity was identified using the DB3 dataset, after its Wikidata URI was mapped to their corresponding DBpedia one using the sameas-all-wikis[6] dataset. Figure 6.1 shows the performance of our systems against the most competing baselines, according to the results of automatic evaluation, across the 99th percentile of the included instance types (i.e. 140 and 107 included instances types in M1 and M2 respectively).

We show that (i) the high performance of our systems is not skewed towards some domains at the expense of others, and that (ii) our architectures has a good generalisation across domains – better than any other baseline. Furthermore, the lowest performance bounds of the Triples2GRU w/ Property Placeholders system are similar to the simple Triples2GRU. However, the successful realisation of a property placeholder provides the

first with a performance boost in cases where the latter would most likely generate a `<resource>` token, whose existence in a summary is essentially punished in the metrics.

TABLE 6.3: Automatic evaluation with the BLEU, METEOR and $ROUGE_L$ metrics (higher values are better) on the validation and test sets.

Model	**BLEU 1**		**BLEU 2**		**BLEU 3**		**BLEU 4**		**METEOR**		**$ROUGE_L$**	
	Valid.	Test	Valid.	Test	Valid.	Test	Valid.	Test	Valid.	Test	Valid.	Test
KN on M1	12.84	12.85	2.28	2.4	0.95	1.04	0.54	0.61	29.04	29.02	17.08	17.09
KN+ on M1	28.93	28.84	21.21	21.16	16.78	16.76	13.42	29.04	29.02	13.42	28.57	28.52
MT on M1	31.12	33.48	19.31	21.12	12.69	13.89	8.49	9.11	31.05	30.1	29.96	30.51
IR on M1	41.39	41.73	34.18	34.58	29.36	29.72	25.68	25.98	32.99	33.33	43.26	43.58
IR+ on M1	49.87	48.96	42.44	41.5	37.29	36.41	33.27	32.51	34.39	34.25	51.66	50.57
Triples2GRU on M1	53.61	54.26	47.38	48.05	42.65	43.32	38.52	39.20	45.89	45.99	64.27	64.64
Triples2GRU w/ Property Placeholders on M1	**54.10**	**54.40**	**47.96**	**48.27**	**43.27**	**43.60**	**39.17**	**39.51**	**46.09**	**46.17**	**64.60**	**64.69**
KN on M2	18.12	17.8	6.91	6.64	4.18	4.00	2.90	2.79	31.05	30.74	37.48	36.90
KN+ on M2	25.17	24.93	16.44	16.30	11.99	11.92	8.77	8.79	33.77	33.71	44.93	44.77
MT on M2	5.35	5.47	1.62	1.62	0.59	0.56	0.26	0.23	0.66	0.68	4.67	4.79
IR on M2	43.01	42.61	33.67	33.46	28.16	28.07	24.35	24.30	20.71	20.46	46.75	45.92
IR+ on M2	**52.75**	**51.66**	43.57	42.53	37.53	36.54	33.35	32.41	31.21	31.04	58.15	57.62
Triples2GRU on M2	49.34	49.40	42.83	42.95	38.28	38.45	34.66	34.85	40.62	**41.13**	66.43	**67.02**
Triples2GRU w/ Property Placeholders on M2	50.22	49.81	**43.57**	**43.19**	**38.93**	**38.62**	**35.27**	**34.95**	**40.80**	40.74	**66.73**	66.61

While automatic evaluation metrics are indicative of "how close" the generated text is to the source summaries, they do not capture its usability by the end user (Reiter, 2010). We conducted two community studies for each investigated language, one for its readers and one for its editors. Our goal was to address the Wikipedia community in order to further evaluate how well the generated text can meet the Wikipedians' standards. The following section first provides further details about the community studies, and then, presents their results and discusses the findings.

6.4 Community Study

We followed the evaluation methodology that was described in Section 3.1.3. Our evaluation targets two different communities: (i) readers (i.e. any speaker of Arabic or Esperanto, that reads Wikipedia, independent of their activity on Wikipedia), and (ii) any active contributor to the corresponding Wikipedia in Arabic or Esperanto. The readers were asked to evaluate summaries generated by our proposed system and the competing baselines against two criteria: (i) fluency (see the **Fluency** paragraph in Section 3.1.2) and (ii) appropriateness (see the **Appropriateness** paragraph in Section 3.1.3). Both those criteria were included in a single survey form for each summary. A different survey form was used for the editors. They were provided with the automatically generated summary that corresponds to a randomly selected item from the test set of the dataset of interest (i.e. either in Esperanto or Arabic) along with its corresponding triples, and they were asked to write a short summary of up to 3 sentences about the allocated item in a dedicated text field. Our goal was to measure the amount of text that the editors reuse from the provided summary (i.e. editors' use; see **Editors' Reuse** in Section 3.1.3). Examples of the two surveys are can be found at: github.com/pvougiou/Mind-the-Language-Gap/tree/master/-crowdevaluation/Examples. In order to sample only participants with previous relevant activity on Wikipedia, we asked them for their reading and editing activity on Wikipedia. The surveys' instructions[3] and announcements[4] were translated in Arabic and Esperanto.

6.4.1 Recruitment

For the recruitment of readers, we wanted to reach fluent speakers of the language. For Arabic, we got in contact with Arabic speaking researchers from research groups working on Wikipedia-related topics. For Esperanto, as there are fewer speakers, and they are harder to reach, we promoted the survey on social media platforms, such as Twitter

[3] https://tinyurl.com/y7cgmesk
[4] github.com/luciekaffee/Announcements

and Reddit[5] using the researchers' accounts. For the recruitment of editors, we posted on the editors' mailing lists[6]. Additionally, for Esperanto we posted on the Wikipedia discussion page[7]. The Arabic editors survey was also promoted at WikiArabia, the conference for the Arabic speaking Wikipedia community. The numbers of participation in all surveys can be found in Table 6.4.

TABLE 6.4: Participation numbers for the community studies in Arabic (using the M1 corpus) and in Esperanto (using the M2 corpus).

Parameter	M1			M2		
	Readers		Editors	Readers		Editors
	Fluency	Approp.	Reuse	Fluency	Approp.	Reuse
# of Participants	27	27	7	27	27	8
# of Sentences	60	60	30	60	60	30
# of Participants Evaluated $\geq$ 50% of Sentences	5	5	2	3	3	2
Avg. # of Sentences per Participant	15.03	14.78	4.00	8.70	8.63	4.75
Total # of Annotations	406	399	33	235	233	38

6.4.2 Readers' Evaluation

We answer whether we can generate summaries that match the quality and style of Wikipedia content in a study with 54 Wikipedia readers from two different Wikipedia languages, Arabic and Esperanto. We created two sets of summaries, one in Arabic and one in Esperanto, consisting of 60 summaries. From those 60 summaries, 30 are generated by the Triples2GRU w/ Property Placeholders system, 15 are from news, and 15 from Wikipedia summaries of the training datasets. For news in Esperanto, we chose introduction sentences of articles of the Le Monde diplomatique[8]. For news in Arabic, we used introduction sentences from the RSS feed of the BBC Arabic[9]. The readers were asked to evaluate each allocated summary against two criteria: (i) fluency (see the **Fluency** paragraph in Section 3.1.2) and (ii) appropriateness (see the **Appropriateness** paragraph in Section 3.1.3). Participants were explicitly asked not to use any external tools for this task. For each sentence, we calculate the mean value for both fluency and appropriateness given by all the participants, and then we average over all the summaries in each set. The results of the readers' evaluation are summarised in Table 6.5.

[5] https://www.reddit.com/r/Esperanto/comments/75rytb/help_in_a_study_using_ai_to_create_esperanto

[6] Esperanto: eliso@lists.wikimedia.org; Arabic: wikiar-l@lists.wikimedia.org

[7] https://eo.wikipedia.org/wiki/Vikipedio:Diskutejo/Diversejo#Help_in_a_study_-improving_Esperanto_text_for_Editors

[8] http://eo.mondediplo.com. Accessed 28 Sep. 2017.

[9] http://feeds.bbci.co.uk/arabic/middleeast/rss.xml. Accessed 28 Sep. 2017.

TABLE 6.5: The average fluency and appropriateness scores of our system against the competing baselines in the readers' evaluation. The results are reported in the "mean (± standard deviation)" format.

	Method	Fluency	Appropriateness
M1	Triples2GRU w/ Property Placeholders	4.7 (±1.2)	77%
	Wikipedia	4.6 (±0.9)	74%
	News	5.3 (±0.4)	35%
M2	Triples2GRU w/ Property Placeholders	4.5 (±1.5)	69%
	Wikipedia	4.9 (±1.2)	84%
	News	4.2 (±1.2)	52%

Overall, the quality of our generated summaries is high (4.7 points in average in Arabic, 4.5 in Esperanto). In Arabic (i.e. M1), 63.3% of the summaries were evaluated to have at least 5 out of 6 in average. In Esperanto, 50% of the summaries have at least a quality of 5 out of 6 in average, with 33% of all summaries given a score of 6 by all participants. This means that similarly to the case of English biographies generation in Chapter 5, the majority of our summaries is highly understandable and grammatically correct. It should also be noted that, our generated summaries were also considered by the participants to have a similar average quality as Wikipedia summaries and news from widely-read media organisations.

Regarding the appropriateness scores, 77% and 69% of the generated summaries in Arabic (using M1) and Esperanto (using M2) respectively were categorised as being part of Wikipedia. In comparison, news sentences were identified as more likely not to fit. In only 35% (Arabic) and 52% (Esperanto) of the cases, readers have mistaken them for Wikipedia sentences. Wikipedia sentences were clearly recognised as such with respective scores of 74% in M1 and 84% in M2 respectively. Wikipedia has a certain writing style, that seems to differ clearly from news, and which the participants were able to recall successfully given the low appropriateness scores of the news sentences compared to the high ones of those from the actual Wikipedia. Our summaries are able to reflect this writing style; the summaries that have been generated by our model (i.e. Triples2GRU w/ Property Placeholders) were evaluated with appropriateness scores close to those of the actual Wikipedia sentences.

6.4.3 Editors' Evaluation

For each corpus, M1 and M2, we compiled a list of 30 randomly selected sets of triples along with the textual summaries that have been generated by our proposed model (i.e.Triples2GRU w/ Property Placeholders). The sets of triples are sampled from the test sets. Our goal is to measure the amount of text that the editors reuse from the

provided summary. Based on the average $\text{score}_{\text{GST}}$ of the generated summaries with their corresponding edited ones, we identify the following categories: (i) wholly-derived (**WD**) ($0.66 \leq \text{score}_{\text{GST}}$), (ii) partially-derived (**PD**) ($0.33 \leq \text{score}_{\text{GST}} < 0.66$), and (iii) non-derived (**ND**) ($\text{score}_{\text{GST}} < 0.33$). Further details about how we compute the editors' reuse are provided in Section 3.1.3.

As shown in Table 6.6, our generated summaries exhibit a high level of re-usability. 78% and 93% of the generated summaries in Arabic and Esperanto respectively were either wholly (**WD**) or partially (**PD**) reused when the edited paragraphs were composed. It is important to note that those percentages demonstrate copies of actual sentence structure rather than copies of names entities or co-occurrence of stop-words.

TABLE 6.6: Percentage of summaries (%) in each category of reuse. Each example consists of the generated summary as it was generated by our neural network architecture (top) and after it is was edited (bottom). Solid lines represent reused tiles, while dashed lines represent overlapping sub-sequences not contributing to the $\text{score}_{\text{GST}}$.

	Reuse Category	%	Example
M1 (Arabic)	**WD**	45.45	خماسي كلوريد الزرنيخ مركب كيميائي له الصيغة(كلمة ناقصة)، ويكون على شكل بلورات بيضاء. خماسي كلوريد الزرنيخ هو مُركب كيميائي له الصيغة(AtClu2085)، ويكون على شكل بلورات بيضاء.
	PD	33.33	بيتش باتوم أوهايو (بالإنجليزية (كلمة ناقصة) Ohio)هي منطقة سكنية تقع في الولايات المتحدة في(كلمة ناقصة.) بيتش باتوم (بالإنجليزية:Beach Batom)هي قرية تقع في الولايات المتحدة الامريكية في بروك كاونتي.
	ND	21.21	دير علا هي بلدة تقع في جنوب غرب إيران. دير علا، أو بيثر، هي قرية أردنية
M2 (Esperanto)	**WD**	78.98	Zederik estas komunumo en la nederlanda provinco Zuid-Holland. Zederik estas komunumo en la nederlanda provinco Zuid-Hooland kaj estas ĉirkaŭata de la municipoj Lopik kaj Zederik.
	PD	15.79	Nova Pádua estas municipo en la brazila subŝtato Suda Rio-Grando, kiu havis (manka nombro) loĝantojn en (jaro). Nova Pádua estas municipo en la brazila subŝtato Suda Rio-Grando.
	ND	5.26	Ibiúna estas municipo de la brazila subŝtato San-Paŭlio, kiu taksis (manka nombro) enloĝantojn en (jaro). Ibiúna estas brazila [[municipo]] kiu troviĝas en la administra unuo [[San-Paŭlo]].

For the wholly derived edits, editors tended to copy the generated summary with minimal modifications, such as tiles A and B in Arabic or G in Esperanto in the examples of Table 6.6. One of the common challenges that inhibits the full re-usability of our generated summaries are the tokens that represent occurrence of a rare word or entity (i.e. entity that has not been matched with any of the input triples; see Sections 4.2 and 6.2 for more details regarding the handling of rare words and entities respectively). In order to provide the participants with a better idea of the nature of those tokens, we replace the tokens of a rare regular word (i.e. `<rare>`) and entities (i.e. `<resource>`) with "(mankas vorto)" and (كلمة مفقودة) in Arabic and Esperanto respectively. Occurrences

of numbers and years are replaced by "(manka nombro)" and "(jaro)" in Esperanto, and (رقم مفقود) and (عام) in Arabic. Usually, those tokens are yielded by our model to represent a missing information in the triples, or when the output word is not in the model's fixed target vocabulary, such as names in different languages. As it can be seen in tiles E and D in the Arabic examples in Table 6.6, editors prefer in those cases to adapt the generated sentences. This can occasionally go as far as making the editor delete the whole sub-sentence in case it contains a high number of such tokens (e.g. in Table 6.6, the sequence of words that is subsequent to tile H is completely deleted by the editor).

By examining our generated summaries we find that such missing tokens are more likely to appear in Arabic than in Esperanto. In the generated summaries from the validation and test set of Esperanto (i.e. M1 corpus), a token that refers to a rare word or entity appears every 0.73 summaries, whereas in Arabic (i.e. M2 corpus) similar tokens occur every 1.21 summaries. We believe that this explains the lower scores of editors' reuse in Arabic (45.45% wholly-derived) compared to those in Esperanto (78.98% wholly-derived) in comparison to Arabic (45.45% **WD**). The difference in the number of special "rare" tokens across the two languages can be explained as follows. First, the significant larger vocabulary size of Arabic, which lowers the probability of a word to be seen by the Arabic model. Second, since the majority of rare tokens are named entities mentioned in foreign languages and since the Latin script of Esperanto is similar to many other languages, the Esperanto model has an advantage over the Arabic one when capturing words representing named entities.

6.5 Conclusion

In this chapter, we explored an extension of the Triples2GRU architecture in order to generate textual summaries from Wikidata triples in underserved Wikipedia languages. Our approach was able to outperform and generalise across domains better than strong baselines of different natures, including MT and a template-based baseline (i.e. the IR+ baseline). This is achieved by leveraging data from a structured knowledge base and careful data preparation in a multilingual fashion, which are of the utmost practical interest for our under-resourced task, that would have otherwise require a substantial amount of additional data.

Furthermore, we conducted a set of community studies in order to measure to what extend our generated summaries match the quality and style of Wikipedia articles, and whether they are useful in terms of reuse by the Wikipedia editors. Members of the targeted communities ranked our generated text close to the expected standards of Wikipedia, and were also likely to assume the generated text is part of Wikipedia. Lastly, we measured the reuse of these summaries by the Wikipedia editors. We found

that the editors are likely to reuse a large portion of the generated summaries, thus, emphasizing the usefulness of our approach to the involved community.

In the next chapter, we investigate the generation of open-domain Wikipedia summaries in English. Rather than handling the realisation of rare entities in the text with additional placeholder tokens that are replaced at a post-processing step, we propose an architecture that "learns" to copy directly in the text the appropriate realisation of an entity or a number from the input triple set.

Chapter 7

Point at the Triple: Improving Neural Wikipedian with a Pointer Mechanism

In the previous chapters we presented how an adaptation of the general encoder-decoder framework can be used for the generation of textual summaries given knowledge base triples as input. Our proposed system exhibited promising results in the generation of single domain summaries (i.e. biographies), and open-domain Wikipedia summaries in under-resourced languages (i.e. Arabic and Esperanto). Similarly to recent related works from Lebret et al. and Chisholm et al., our proposed architectures leverage a set of *placeholder* tokens (i.e. property or property-type placeholders) in order to verbalise rare or unseen entities. As explained in Sections 5.1.3 and 6.1.1, when these tokens are generated by a system, they are replaced in a post-processing step with the label of the entity that satisfies the requirements of the original placeholder. However, this introduces a certain degree of stochasticity in the case that multiple relations from the input meet the requirements of the predicted placeholders.

In this chapter, we introduce an approach that addresses this concern. The NLG task that is investigated in this chapter is identical to the one presented in Chapters 5 and 6. Table 7.1 presents an example of this task. Our systems summarises a set of triples about the book *Atlas Shrugged* (i.e. `Atlas_Shrugged` participates as either the subject or the object of the given input relationships) in the form of sensible and coherent text. Our model is inspired by *pointer-generator* networks which have been recently introduced in text summarisation tasks (Gu et al., 2016; See et al., 2017). These constitute expansions of the original pointer network suggested by Vinyals et al. which was only able to generate an output sequence just by *copying* tokens from the source sequence. This functionality is essentially achieved by selecting the token that an attentive decoder pays most attention to.

TABLE 7.1: An idealised example of our NLG task. Our system generates a textual summary of an input set of un-ordered triple-facts about *Atlas Shrugged*. The red-coloured part of each triple is the property or predicate of the triple. The items before and after each predicate are the subject and object respectively of each triple. Numerical values (e.g. `1957`) can only appear at the object's position.

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`

Textual Summary	Atlas Shrugged is a science fiction novel by Ayn Rand.

Our novel approach jointly learns (i) to verbalise in a different number of ways the entities of *pointed* triples, (ii) to copy the label or the number in the case that the pointed triple consists of either infrequent entities or numbers, or (iii) to generate words or realisations of entities from a fixed target vocabulary.

We train and evaluate our approach on the two corpora that are based on DBpedia triples, D1 and D3. D1 enables us to explore the efficacy of our approach in a single domain textual generation scenario. Since D3 is a dataset encompassing the entirety of Wikipedia, rather than just biographies, it allows us to demonstrate our model's ability to generalise on a much more challenging task. We compare our performance against a set of baselines from Section 3.2 and the architecture that has been described in Chapter 5.

7.1 The Model

We assume our model is trained with records consisting of an English language summary and an aligned 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. The alignment of the triple: `Dwayne_Johnson occupation Actor` to the sentence: "Dwayne Johnson is **an actor** ..." is an example of explicit realisation; whereas 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 of 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 $f_1, f_2, \ldots, f_E$:

$$p(\mathbf{y}|F_z) = \prod_{t=1}^{T} p(y_t|y_1, \ldots y_{t-1}, F_z) \ , \tag{7.1}$$

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

Our approach is inspired by recent work by See et al. on text summarisation. The former employs a pointer-generator capable of both copying tokens from the input sequence and generating words from the fixed vocabulary of the decoder. While this model handles sequential inputs and outputs, in our case the sets of input triples are un-ordered, and not sequentially correlated. Consequently, we adopt the triple encoder proposed in Section 5.1.1, and we compute the relevant attention scores on top of this feed-forward architecture.

In many cases the entities that participate in the relationships of the input sets cannot be directly copied to the generated text. For example, the entities of `dbr:Actor` and `dbr:United_States` can be expressed, based on the context, as "actor" or "actress" and as "United States" or "American" respectively. Consequently, we propose a mechanism that enables our model to learn different realisations for the entities of the pointed triples. Our mechanism also allows the network to handle the occurrence of rare entities (for which we are unable to learn good vector representations) or numbers that participate in the pointed relationships by directly copying them (e.g. years) or their labels (see Section 7.1.3) in the generated summary.

7.1.1 Decoder

Given the improved performance of the GRU-based architectures in Chapter 5 and their lower complexity compared to the LSTMs (Chung et al., 2014), we implement the decoder as a multi-gated RNN variant with GRUs. Let $h_t^l \in \mathbb{R}^m$ be the aggregated output of a hidden unit at timestep $t \in [1 \ldots T]$ and layer depth $l \in [1 \ldots L]$. All the subsequent 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:

$$h_t^l = \text{GRU}(h_{t-1}^l, h_t^{l-1}) \ . \tag{7.2}$$

7.1.2 Triple Encoder

By leveraging the triple encoder that has been presented in Section 5.1.1, our model computes the vector representation h_{f_i} of the i-th triple by forward propagating the triple encoder as follows:

$$\tilde{h}_{f_i} = [\mathbf{W}_{\mathbf{x}\to\tilde{\mathbf{h}}} s_i; \mathbf{W}_{\mathbf{x}\to\tilde{\mathbf{h}}} p_i; \mathbf{W}_{\mathbf{x}\to\tilde{\mathbf{h}}} o_i] \ , \tag{7.3}$$

$$h_{f_i} = \text{ReLU}(\mathbf{W}_{\tilde{\mathbf{h}}\to\mathbf{h}} \tilde{h}_{f_i}) \ , \tag{7.4}$$

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 predicates and entities 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 enforcing the model to compress all the available information that is contained in the triples in a single vector (see Eq. 5.5), we investigate the implementation of an attention mechanism over all the given triples. We adopt the global attention mechanism that has already been successfully employed in the domain of semantic parsing (Dong and Lapata, 2016) and machine translation (Luong et al., 2015b). 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:

$$a_t^{(i)} = \frac{\exp[(h_t^L)^T \mathbf{W}_\mathbf{a} h_{f_i}]}{\sum_{j=1}^{E} \exp[(h_t^L)^T \mathbf{W}_\mathbf{a} h_{f_j}]} \ . \tag{7.5}$$

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

$$c_t = \sum_{i=1}^{E} a_t^{(i)} h_{f_i} \ . \tag{7.6}$$

The above context vector allows the decoder to selectively decide to which part of the source triples it should pay attention to. The alignment between the context vector and the information that has already 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:

$$h_t^{L+1} = \tanh\left(\mathbf{W}_\mathbf{c} c_t + \mathbf{W}_\mathbf{h} h_t^L\right) \ . \tag{7.7}$$

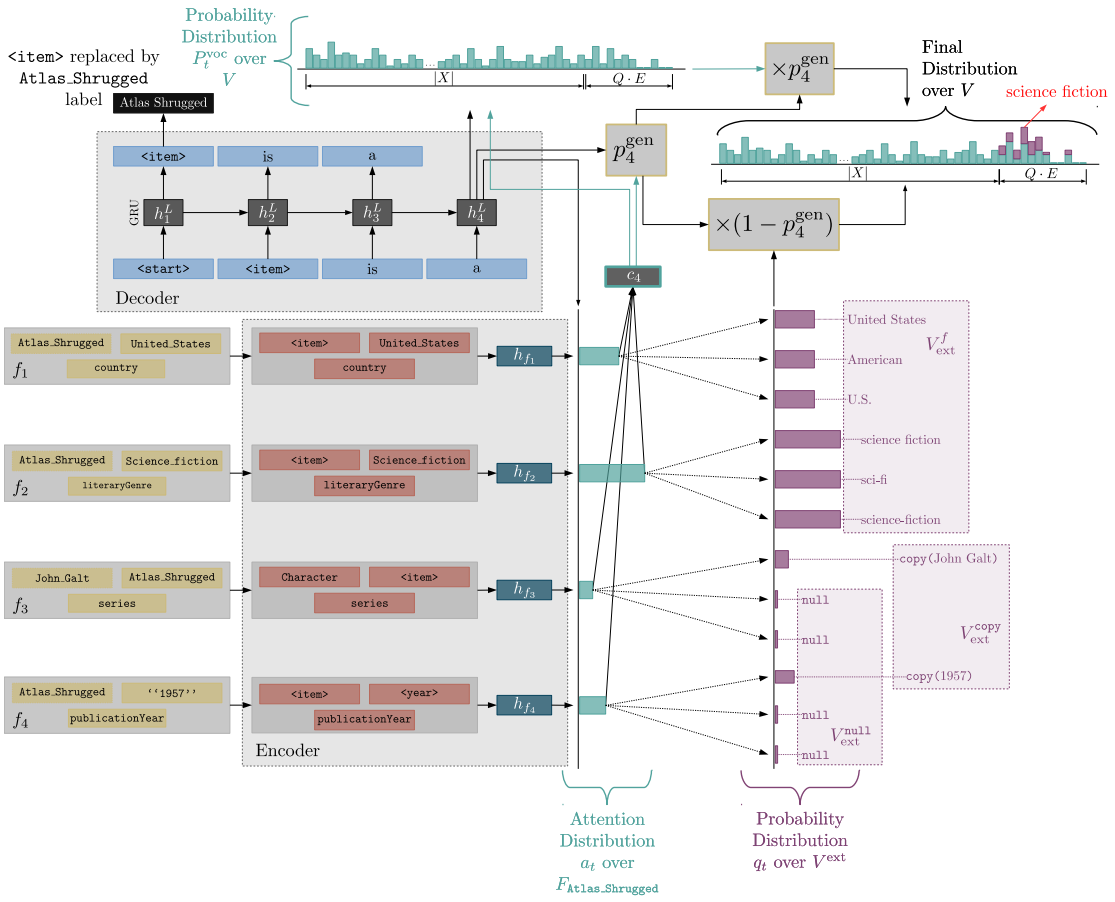


FIGURE 7.1: The architecture of our pointer-generator network. At timestep $t = 4$, p_4^{gen} weights the probability of copying a word from V^{ext} higher than generating a word from the fixed vocabulary $V^{\dagger}$. The decoder learns to interpret the weighted sum of h_4^L and c_4 in order to compute a probability distribution about which realisation is the most appropriate given the context from 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.

7.1.3 Dynamically Expanding the Vocabulary

As we explained above, the pointer-generator network proposed by See et al. would be unfit for the task. See et al.'s model is capable of both copying tokens from the input sequence and generating words from the decoder's fixed vocabulary. In the context of its copying functionality, their model learns to copy only a single realisation for its input token. This would result in using the same label for each copied entity regardless of the context of the text. Similarly to case of our w/ URIs systems (described in detail in Section 5.2.1.1), such an approach would essentially result in less fluent text (see Section 5.3.3).

Addressing the limitation of the architecture of See et al., we propose a mechanism that

enables our model to learn different realisations for the entity of a *pointed* triple. Our architecture is capable of (i) generating words from a fixed target vocabulary, (ii) copying a number of different surface forms for the entities in the input triple set, and (iii) copying the number or the label in the case of 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.

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 tuples that are associated with the entity k_d, s.t. $z_j^{k_d} \geq z_{j+1}^{k_d} \forall j \in [1, R]$, where R is the total number of realisations of k_d. We compute the 95^{th} 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 with which our model learns to express k_d in the generated summary. Q is a dataset-relevant 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. 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.:

$$v_j^{\text{ext}} = \begin{cases} g_{j\%Q}^{e_{\lceil j/Q \rceil}} \in V_{\text{ext}}^f & e_{\lceil j/Q \rceil} \in H^{(f)} \\ g_1^{e_{\lceil j/Q \rceil}} \in V_{\text{ext}}^{\texttt{copy}} & e_{\lceil j/Q \rceil} \in H^{(i)} \text{ and } j\%Q = 1 \\ g_1^{e_{\lceil j/Q \rceil}} \in V_{\text{ext}}^{\texttt{copy}} & e_{\lceil j/Q \rceil} \in \mathbb{R} \text{ and } j\%Q = 1 \\ \texttt{null} \in V_{\text{ext}}^{\texttt{null}} & \text{otherwise} \end{cases} \tag{7.8}$$

$\forall j \in [1, Q \cdot E]$, 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 the example of Figure 7.1 where $Q = 3$, the frequent entity of `United_States` $\in H^{(f)}$,

results in the inclusions 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 triples encoder in Figure 7.1). In such scenario, all the values of V^{ext} that correspond to this particular triple are filled with `null`, except than the first one which refers to the copy of the label of this rare entity. Similar methodology is used in the case of numbers (e.g. ``1957'' is replaced by the `<year>` token, and the first slot of the positions in V^{ext} that correspond to this triple is filled with the copy of the year token in Figure 7.1).

7.1.4 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:

$$\widetilde{q}_t^{(i)} = \begin{cases} \exp[a_t^{(\lceil i/Q \rceil)}] & i \in V_{\text{ext}}^{f} \cup V_{\text{ext}}^{\texttt{copy}} \\ 0 & i \in V_{\text{ext}}^{\texttt{null}} \end{cases} \tag{7.9}$$

$$q_t^{(i)} = \frac{\widetilde{q}_t^{(i)}}{\sum_{j=1}^{Q \cdot E} \widetilde{q}_t^{(j)}} \ . \tag{7.10}$$

We adopt the notion of the *generation probability* $p_t^{\text{gen}} \in [0, 1]$, which is used 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).

$$p_t^{\text{gen}} = \text{sigm}(\mathbf{W}_{\hat{\mathbf{c}}} c_t + \mathbf{W}_{\hat{\mathbf{h}}} h_t^L) \ , \tag{7.11}$$

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:

$$P_t(w) = \begin{cases} p_t^{\text{gen}} P_t^{\text{voc}}(w) + \\ (1 - p_t^{\text{gen}}) q_t^{(\lceil w/Q \rceil)} & w \in V_{\text{ext}}^{f} \cup V_{\text{ext}}^{\texttt{copy}} \\ p_t^{\text{gen}} P_t^{\text{voc}}(w) & w \in V^{\dagger} \\ 0 & w \in V_{\text{ext}}^{\texttt{null}} \ , \end{cases} \tag{7.12}$$

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.

7.2 Dataset Preparation

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 replaced by the token of the position of the surface form in V^{ext}. Table 7.2 displays an example of the alignment of the datasets after the necessary pre-processing.

For each input triple set, we are forming the values of the extended vocabulary V^{ext}. Each triple is provided with Q slots in 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 models. In such scenario, all the values of V^{ext} that correspond to this triple are filled with `null`, except than the first one which refers to the copy of the label of the rare entity. Similar methodology is used in the case of years[1]. We computed a Q value of 2 (for the D1 dataset), and 3 (for the D3 corpus) which results in $\sim 98\%$ coverage of the total number of textual realisations of the triples entities for both corpora.

Years which have not been identified in the input set of triples are mapped to the special `<year>` token. For simplicity, we choose not to map the occurrence of regular numbers in the text to the corresponding input triples. Consequently, all numbers in the text are replaced by the special `0` token. Every out-of-vocabulary token in the textual summaries is represented by the special `<rare>` token in the case of regular 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 DB3 dataset.

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

$$\lfloor E_{\text{min}} + 0.25\sigma_E \rfloor \leq E \leq \left\lfloor \overline{E} + 2\sigma_E \right\rfloor \quad . \tag{7.13}$$

This leads to a slightly greater range of number of input triples compared to the experiments in Chapter 5 and 6 for which the range of had been set according to Eq 5.10.

[1] While in our approach we limit our approach to years, theoretically we could address in the text the existence of all numbers in the triples.

TABLE 7.2: An example of the alignment of the datasets. The main-discussed entity both in the triples and the corresponding summary is replaced with the `<item>` token. Each triple is stored along with the instance type of the other than the main entity that participates in it. The word "village" is recognised as the DBpedia entity `dbr:Village`, which also participates in the triples of the input set. Since "village" is also one of the realisations associated with `dbr:Village` in the 10^{th} position of V^{ext}, it is replaced by the `[ext_10]` token in the text. Each summary is augmented with the respective start-of-summary `<start>` and end-of-summary `<end>` tokens.

`<item>`	`dbr:Čizma`
Triples	`<item> dbo:country dbr:Bosnia_and_Herzegovina [dbo:Country]`
	`<item> dbo:isPartOf dbr:Kiseljak [dbo:Settlement]`
	`<item> dbo:timeZone dbr:Central_European_Time [unknown_type]`
	`<item> dbo:type dbr:Village [owl#Thing]`
	`<item> dbo:utcOffset 0 [unknown_type]`
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>`

V^{ext}
1: Bosnia
2: Bosian
3: Bosnia-Herzegovina
4: Kiseljak
5: `null`
6: `null`
7: CET
8: cet
9: UTC+2
10: village
11: rural community
12: selo
13: 0
14: `null`
15: `null`

This maximum number of triples resulted in a model size (i.e. in the case of the pointer-generator systems) of around $\geq$ 10 GB, consuming almost the entirety of the available GPU memory of the Titan X (Pascal) GPU that was used for these experiments. We follow the same methodology as in Section 6.2 in order to filter out any potential redundant triples from the oversized sets, and prioritise triples whose objects or subjects have been mentioned in the text. We perform this in order to maximise the effect of the proposed pointer mechanism and the property-types placeholders of the competing Triples2GRU and Triples2LSTM systems (both are discussed in detail in Chapter 5).

7.3 Experiments

We train and evaluate the performance of our approach on the D1 and D3 corpora. Both datasets are split into training, validation and test, with respective portions of 85%, 10%, and 5%. Furthermore, we investigate whether the inclusion of the frequent surface form tuples (see Section 5.2.1.2), whose entities have not been associated with any triple from the input set, enhances the performance of our model. Consequently, for each dataset we run two sets of experiments one in which the surface form tuples are part of the fixed vocabulary of the decoder (w/ Surf. Form Tuples), and one in which they are treated as regular words (w/o Surf. Form Tuples).

7.3.1 Training Details

On the encoder side, we include all entities and properties that occur at least 30 times in the related dataset. Triples with rare properties are excluded. Infrequent entities are replaced by their respective instance type tokens. This results in a source vocabulary size of $|N| = 5785$ and 17146 tokens in the case of the D1 and D3 dataset respectively.

On the decoder side, we use a single layer of 500 GRUs, and we include 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 respective D1 and D3 datasets. In all the experiments, we set the dimensionality of the hidden states to $m = 500$. We initialise all parameters with random uniform distribution between -0.1 and 0.1, and we use 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 is to minimise the sum of the negative log-likelihoods of a mini-batch of 80 predicted summaries. Optimisation is performed using Adam[2] (Kingma and Ba, 2014) with a learning rate of $5 \cdot 10^{-5}$. An l_2 regularisation term of 0.05 over the parameters is also included in the cost function (Wen et al., 2015, 2016).

The networks converge[3] after the 13^{th} epoch in the case of D1 (i.e. biographies generation). In the case of the D3 dataset, convergence is achieved after 20 epochs in the w/o Surf. Form Tuples system, and after 75 in the case of w/ Surf. Form Tuples. All of our systems are trained on a single Titan X (Pascal) GPU. The pointer-generator networks

[2]While both RMSProp and Adam were converging after the same epoch, we noticed a minor improvement with respect to the lowest achieved validation error using Adam on our pointer-generator architectures. In the case of Triples2GRU, the lowest validation error was almost identical using both RMSProp and Adam.

[3]The epoch at which the model converges to the lowest possible validation error. After this epoch, the error on the validation set either does not improve further or it increases, and, thus, the model is overfitting.

complete an epoch of training in around 36 minutes and around 2 hours when trained on the D1 and D3 corpora respectively.

During testing and evaluation, we do beam-search (see Section 5.1.5) with a beam size of 8, and we retain only the summary with the highest probability.

7.3.2 Automatic Evaluation

We use BLEU (Papineni et al., 2002), ROUGE (Lin, 2004), and METEOR (Lavie and Agarwal, 2007) on the validation and test set of each corpus.

We demonstrate the effectiveness of our approach by comparing it against a set of competitive baselines. First, we compute the expected lower bounds for BLEU scores by using a random Wikipedia summary generation baseline (cf. Section 3.2.1). 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. Secondly, we use the KenLM toolkit (Heafield et al., 2013) in order to build a 5-gram Kneser-Ney (KN+) language model (cf. Section 3.2.2). Thirdly, we compare our approach to the IR+ baseline (see Section 3.2.3). Finally, we compare our systems against the Triples2GRU and Triples2LSTM architectures that have been proposed in Chapter 5. We equip both architectures with the mechanism of surface form tuples (w/ Surf. Form Tuples), and we set the dimensionality of their hidden state to 500. All baselines are equipped with the `<item>` and the property-type placeholders. After a summary is sampled, the first are replaced by the label of the main discussed entity, and the second by the entity of the triple from the input set that satisfies the requirements of the placeholder. During testing, in the case of KN, Triples2LSTM and Triples2GRU, we do beam-search (see Section 5.1.5) with a beam size of 8 in order to sample the most probable summaries for each set of triples.

BLEU 1, BLEU 2, BLEU 3, BLEU 4, ROUGE_L (computed on the longest common sub-sequence) and METEOR results are reported in Table 7.3. Please note that the reason for the performance of the Triples2GRU and Triples2LSTM systems to be lower than the ones reported in the experiments in Section 5.3.2 is that in the former setup all occurrences of number and years in the text are replaced by the special `0` and `<year>` tokens (see Section 4.2) (Lebret et al., 2016). Since the systems presented in Chapter 5 have not been modelled with the ability to handle numbers, they are not punished in the corresponding automatic evaluation section (see Section 5.3.2) when they generate such tokens. Since our current setup is capable of generating years[4], generating years correctly in the text is properly reflected in the scores.

[4]In theory, the current pointer-generator network can work for regular numbers as well. We leave this for future work.

Interestingly the IR+ is the best and third-best performing system on D3 according to the BLEU 1 and 2 metrics. This shows that a template-retrieval system can achieve high precision in low order n-grams. However, it is much less competitive compared to the neural network approaches according to BLEU $3-4$, METEOR and ROUGE. In almost all scenarios (except BLEU 1 on D3), our systems outperform the baselines. In the case of D1 (i.e. generation of biographies), our systems achieve an improvement of at least 4.8 BLEU 4 and 0.91 ROUGE points in comparison to the Triples2GRU, which is our strongest competitor. In the case of the D3 dataset, our pointer-generator network provides an improvement that ranges from 0.27 to 1.36 and 0.99 to 2.96 BLEU and ROUGE points respectively. The high METEOR scores of our model w/o Surf. Form Tuples also indicate that it often generates text that differs from the empirical summaries due to morphological or synonymic variations. We believe that the lower performance difference of our systems from the baselines on the D3 dataset is partially attributed to the lower coverage of the input triples with respect to their corresponding summaries in the non-biographical articles. We found that in D1 the number of tokens that are identified as realisations of entities or years from the input triples is 2.51 ($\pm$1.62) tokens whereas in the D3 dataset is 2.03 ($\pm$1.50). We believe that the fact that those numbers would have been higher should we have opted to relate the occurrence of numbers (not only years) in the corresponding input triples emphasises the superiority of our approach.

Table 7.4 presents the performance of all the models that are trained on the D3 dataset on the triple sets of their validation and test sets that are also part of D1 (i.e. the dataset of biographies). The experiment shows that training our systems on a much more challenging dataset (see Table 4.8 and 4.9) with the same hyper-parameters in terms of the dimensionality of the hidden states and the number of layers results in only minor performance drop against systems trained specifically for the task.

In addition to the above experiments, we group Wikipedia summaries that are allocated to the same number of input triples and compute a BLEU score per group. Figure 7.2 displays the performance of our models with the BLEU 4 metric on the 99th percentile of the D1 and D3 test sets across different numbers of input triples. Please note that sets of triples that consist of more than 26 and 21 triples in the case of D1 and D3 corpus respectively are inputted to the systems after they are stripped of their additional triples, according to the methodology described in Section 7.2. Similarly to the experiments presented in Section 5.3.2, we observed that when the systems are initialised with a low number of triples are lacking information required to form a two-sentence summary. In the case of D1 the performance of the pointer-generator networks progressively increases as more triples are given to the system as input. This shows the ability of the model to successfully "select" the relevant triples and address them in the generated summary. D3 is a much more challenging corpus due to the size of its source and target dictionaries (cf. Table 4.9 and 4.8) and the low average number of triples' entities that are identified in the non-biographical articles ($\sim$ 1.8 entities per summary). The latter is of great

importance since the ability to directly copy in the text information from the triples is essentially what separates the two architectures. In a scenario in which no information in the text is directly taken from the triples, we would expect both models to score almost identical performance[5]. However, even in this scenario, the systems based on the pointer mechanism consistently outperform the competition.

We also noted a drop in performance when our systems are provided with oversized triple sets. This is more noticeable in the case of D3, and is mainly due to the upper bounds with respect to the number of allocated triples that we set per summary (Eq. 7.13). Based on these upper bounds, we apply a simple approach of eliminating redundant triples (described in detail in Section 5.2.2), but in case their number still exceeds the threshold, we use 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 are recognised as realisations of entities or years is relatively low, it is likely the $E_{\max}$ triples that we retain from a very large set might not be at all reflected in their corresponding 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.

In order to further investigate how well our systems generalise across different categories, we group the Wikipedia summaries from the D3 dataset according to the instance type of their main entity (e.g. `dbo:Village` and `dbo:SoccerPlayer`). Figure 7.3 shows the performance of our systems against the baselines across the 99^{th} percentile of the ~ 225 included instance types. The lowest performance bounds of our systems are similar to the ones of Triples2GRU 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 models significantly outperform the competition. For example, while Triples2GRU summaries about `dbo:Village` and `dbo:IceHockeyPlayer` are one of the highest scored instance types[6] for both the Triples2GRU and Triples2LSTM systems, with respective BLEU 4 scores of 34.98 and 32.68 in the case of the Triples2GRU and 32.54 and 31.12 in the case of the Triples2LSTM, they are outperformed by both our systems (i.e. 37.10 and 36.01 in the case of w/ Surf. Form Tuples, and 40.38 and 36.94 in the case of w/o Surf. Form Tuples).

[5] In theory, the pointer-generator network should still have some advantage due its attention mechanism.

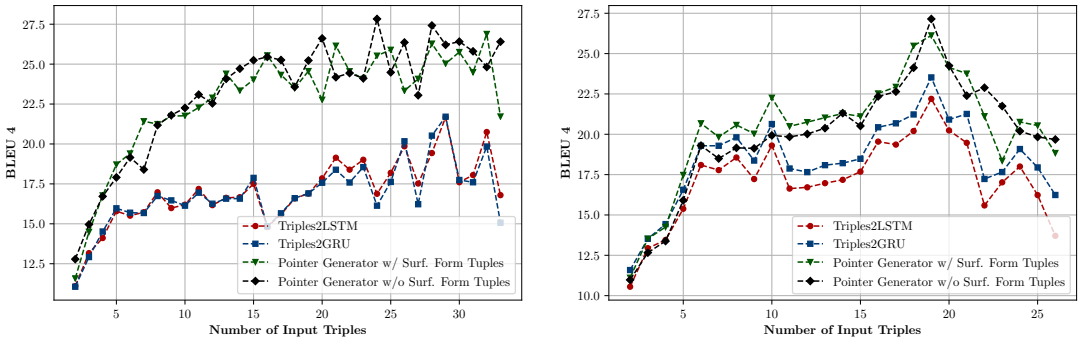
[6] Only instance types with more than 100 generated summaries in the test sets were considered.

TABLE 7.3: Automatic evaluation of our architectures against all other baselines using BLEU $1-4$, ROUGE_L and METEOR on the validation and test set of the D1 and D3 datasets. The average performance of the Random+ baseline along with its standard deviation is reported after sampling 10 times.

Model	**BLEU 1**		**BLEU 2**		**BLEU 3**		**BLEU 4**		**ROUGE_L**		**METEOR**	
	Valid.	Test	Valid.	Test	Valid.	Test	Valid.	Test	Valid.	Test	Valid.	Test
Random+	28.26 (±.02)	27.96 (±.02)	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+	22.04	21.79	15.16	14.97	11.00	10.85	8.04	7.94	36.74	36.47	31.46	31.32
IR+	36.51	36.37	24.60	24.44	18.16	17.96	13.66	13.45	36.20	36.08	18.12	18.02
Triples2LSTM	32.94	32.61	25.27	24.99	20.33	20.08	16.28	16.08	48.48	48.23	33.96	33.84
Triples2GRU	33.53	33.11	25.54	25.18	20.46	20.14	16.35	16.07	48.20	47.87	33.38	33.19
Pointer Generator on D1 w/o Surf. Form Tuples	**37.12**	**36.93**	**29.40**	**29.27**	**24.79**	**24.67**	**21.39**	21.26	49.11	49.12	33.92	33.98
Pointer Generator on D1 w/ Surf. Form Tuples	35.77	35.85	28.68	28.81	24.37	24.54	21.15	**21.34**	**49.65**	**49.90**	**34.96**	**35.30**
Random+	25.16 (±.01)	25.23 (±.01)	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+	19.13	19.15	11.98	11.99	8.07	8.06	5.55	5.53	28.49	28.50	16.50	16.50
IR+	**38.92**	**38.85**	27.24	27.09	20.88	20.68	16.66	16.45	38.62	38.46	17.77	17.59
Triples2LSTM w/ Surf. Form Tuples	34.34	34.40	25.89	25.85	20.56	20.48	17.87	17.73	46.20	46.23	27.08	27.03
Triples2GRU w/ Surf. Form Tuples	35.72	35.70	27.21	27.13	21.84	21.73	17.87	17.73	**47.33**	**47.36**	27.21	27.23
Pointer Generator on D3 w/o Surf. Form Tuples	34.87	34.80	26.81	26.75	21.78	21.71	18.14	18.08	46.59	46.58	**30.13**	**30.19**
Pointer Generator on D3 w/ Surf. Form Tuples	36.55	36.58	**28.07**	**28.11**	**22.81**	**22.86**	**19.04**	**19.09**	47.22	47.20	28.20	28.31

TABLE 7.4: Automatic evaluation of our architectures against all other baselines using BLEU $1-4$, ROUGE_L and METEOR on the triple sets of validation and test set of the D3 dataset that belong to the D1 one.

Model	**BLEU 1**		**BLEU 2**		**BLEU 3**		**BLEU 4**		**ROUGE_L**		**METEOR**	
	Valid.	Test	Valid.	Test	Valid.	Test	Valid.	Test	Valid.	Test	Valid.	Test
KN+	16.91	16.97	10.13	10.16	6.47	6.46	3.64	3.61	25.31	25.33	15.84	15.81
IR+	**36.41**	**36.65**	24.23	24.37	17.75	17.82	13.24	13.27	35.81	35.91	17.58	17.48
Triples2LSTM w/ Surf. Form Tuples	32.48	32.87	24.41	24.64	19.32	19.47	15.37	15.46	47.06	47.41	30.73	30.71
Triples2GRU w/ Surf. Form Tuples	32.88	33.09	24.95	25.06	19.92	19.99	15.88	15.90	47.71	47.99	32.56	32.52
Pointer-Generator w/o Surf. Form Tuples	33.58	33.67	26.79	26.94	22.71	22.87	19.65	19.82	48.58	48.82	34.81	35.05
Pointer-Generator w/ Surf. Form Tuples	34.77	34.67	**27.76**	**27.64**	**23.49**	**23.37**	**20.30**	**20.17**	**49.47**	**49.51**	**34.19**	**34.42**



(A) D1 (B) D3

FIGURE 7.2: Performance of our models with the BLEU 4 metric across the different sizes of triple sets from the test set of the D1 (a) and D3 (b) dataset. Please note that sets of triples that consist of more than 26 and 21 triples in the case of D1 and D3 corpus respectively are inputted to the systems after they are stripped of their additional triples according to the methodology described in Section 7.2.

7.3.3 Human Evaluation

The above text similarity metrics have limitations in tasks with loose correlation between the input and the expected output (Reiter, 2010). Consequently, we also evaluate the performance of our approach in two separate user studies on the Figure Eight platform[7]. In the first, we explore the performance of our networks against the Triples2GRU (the most competitive system according to the results of Chapter 5) to generate open-domain summaries when trained on the D3 corpus. The goal of the second study is to investigate whether training our systems on the Full dataset with the same hyper-parameters (except the size of the input and output vocabularies) results in comparable performance against systems trained on the D1 corpus. Our experiments showed that in our dataset, triple sets with fewer triples usually lack enough information for our systems to generate a summary (see Figure 7.2). Hence, in both studies we include only sets that consist of at least 6 triples. For the first case study, we identified the triple sets that occur in the test sets of all the three neural-network-based systems. From those, we sampled 25 sets of triples according to the instance types distribution of the main discussed entities in the D3 dataset. For the second experiment, we compiled a list of the sets of triples that are part of the test sets of our four investigated systems. From this list, we randomly selected 25 sets of triples. For each study, we collected the summaries that have been generated by each one of the investigated systems given the selected input sets of triples.

Each contributor was asked to evaluate three, in the case of the first study, and four, in the case of the second, generated summaries against three criteria: (i) fluency, (ii) coverage (triples whose information is mentioned either implicitly or explicitly in the text), (iii) contradiction (information that exists in the sentence but it conflicts with one or

[7] https://www.figure-eight.com

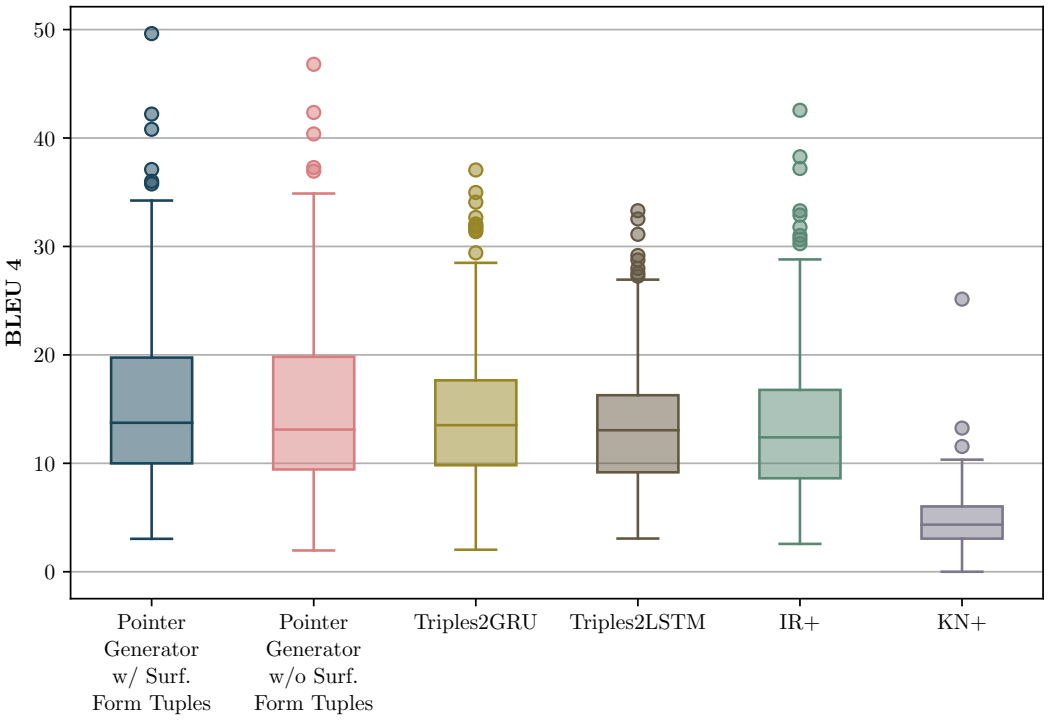


FIGURE 7.3: Performance across 225 domains.

more of triples from the input set). For fluency, the contributors were presented with all the generated summaries, and they had to score each one of them on a scale from 1 to 6, with 1 indicating an incomprehensible summary and 6 a coherent and grammatically correct one. For coverage and contradiction, the participants were provided with the input triples, and for each one of the generated summaries, they had to identify the relation of each triple to the summary as either "Absent" or "Present" and as "Direct Contradiction" and "Not a Contradiction" respectively. For coverage, we computed first the total number of triples that have been covered in a generated summary after performing majority voting over all the contributions of each fact with respect to this summary. For each summary, the number of covered triples is normalised by the total number of triples of the respective input set. These criteria are described in detail in Section 3.1.2. Each generated summary by each system is evaluated by 10 workers. The average scores are obtained after averaging the respective fluency, coverage and contradiction scores of all the 25 summaries that have been sampled from each system.

In order to further explore whether our crowdworkers annotate the generated summaries according to the designated standards, we also run a subsequent pilot study with two researchers from the University of Southampton who are experts in the field of Semantic Web and have full professional proficiency in English. The experts were tasked to repeat the first case study by evaluating the summaries that had been generated by the Triples2GRU system.

TABLE 7.5: Average rating of the investigated systems against the human evaluation criteria. **Top:** Scores of the systems on the D3 dataset. **Bottom:** Scores of the systems evaluated on biographies. The "... on D1" systems have been trained on biographies only.

	Model	Fluency	Coverage	Contradiction
D3	Triples2GRU	4.86	26.76	31.96
	Pointer Generator w/o Surf. Form Tuples	**5.14**	31.60	30.36
	Pointer Generator w/ Surf. Form Tuples	**5.06**	**34.24**	32.64
D1	w/o Surf. Form Tuples on D1	4.71	40.36	14.36
	w/ Surf. Form Tuples on D1	4.81	39.32	17.24
	w/o Surf. Form Tuples on D3	4.44	30.20	15.76
	w/ Surf. Form Tuples on D3	5.10	35.92	20.16

The results of the two studies are in alignment with the results of the automatic evaluation (cf. Table 7.5). In the first study, our architectures achieve greater fluency and coverage scores compared to the Triples2GRU. The coverage of "w/ Surf. Form Tuples" is also significantly better ($p < .05$) than the Triples2GRU. While the system 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 re-produce the same quality of summaries when trained with the D3 dataset. Nonetheless, the inclusion of the surface form tuples makes the model more flexible at addressing larger vocabulary sizes without sacrificing significant performance on single domains. When trained on the full corpus (i.e. D3) and tested on biographies (i.e. D1), the system w/ Surf. Form Tuples is significantly more fluent than w/o Surf. Form Tuples ($p < .01$). Its fluency scores were also slightly improved compared to when it was trained only on D1.

We should also note that in the training portion of the D3 dataset, the triples entities are realised using the first realisation, 86% of the time, and the second and third, 11.5% and 2.5%, respectively, of the time. During testing, entities from the triples are realised using the first, second, and third realisation with respective percentages of 85, 11, and 4. We believe that this, along with the high fluency scores of our approach, highlights its ability to verbalise entities from the triples in a different number of ways in the text.

A Note on the Human Evaluation Metrics. Despite our greater efforts, we faced extreme difficulty to communicate with the crowdworkers the purpose of contradictions. In a preliminary version of our experiments, each triple was to be marked as either "Contradiction" or "Not Contradiction". However, this proved inadequate since workers were marking triples that were not covered in the summary as contradicting, resulting in an average of $\sim 50\%$ of triples whose information is contradicted in the summaries

across all the investigated systems. The same evaluation when performed by the two experts on the summaries that had been generated by the Triples2GRU system resulted in an average percentage 0.7% triples that are contradicted in the summaries (i.e. a result which is also much more consistent with the contradiction ratings on the system in Section 5.3.3). In order to minimise the effect of contradictions, besides changing the available labels for each triple to "Direct Contradiction" and "Not a Contradiction", in the relevant instructions, we explicitly noted that contradictions should be rare, and that we expected many summaries without any of them. By comparing the 31.96% contradiction as it was reported by the crowdworkers to the 0.7 of the experts, we conclude that workers tended ($p < 0.05$) to significantly overestimate the presence of triples that are contradicted in the generated summaries.

Such differences were observed in the way that both the experts and the crowdworkers evaluate fluency and coverage. The average fluency with which the experts evaluated the summaries by the Triples2GRU was 5.28. The ANOVA test computed on the two fluency score series produced $p < 0.05$. Consequently, we can claim that compared to the experts, crowdworkers tend to systematically underestimate the summaries' fluency by 0.5 out of 6. According to the experts, 39.71% of the input triples is covered in the summaries that are generated by the Triples2GRU system. As a result, workers tended to undercount the presence of facts in the generated summaries (confirmed by ANOVA test $p < 0.05$). Furthermore, a positive significant correlation (Pearson = 0.64) pointed out that workers evaluate coverage in a consistent manner with the experts.

7.4 Summary and Discussion

In this chapter, we have presented a pointer mechanism that enhances our original encoder-decoder architecture that was presented in Chapter 5. Our network 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. This mechanism sidesteps one of the limitations of the systems presented in Chapters 5 and 6 which was their inability to realise numbers from the triples in the generated text. We test this novel system on single- and open-domain Wikipedia summaries generation using the D1 and D3 corpora. Results from both automatic and human evaluation highlight the superiority of our approach compared to our original encoder-decoder architecture (in Section 5) and a set of baseline of different natures.

While in our experiment we observed only few repetitions of textual content, it might be one of the challenges in a multi-sentence generation scenario. A natural extension of this work is the implementation of a *coverage* mechanism on top of our attention mechanism that would discourage it from attending the same triples during the summary generation procedure (Tu et al., 2016; Mi et al., 2016).

Our experiments have also showed that there are differences between the scores that are provided by experts and crowdworkers. These differences are more notable in the case of the less trivial evaluation criteria (i.e. contradiction). However, using explicitly experts for the evaluation of the various NLG system results to non-scalable evaluation methodologies. For further details regarding this finding, we urge interested readers to refer to the corresponding published paper (Vougiouklis et al., 2018b). A part of our future work will focus on the methods with which the crowdworkers should be trained in order to perform more accurately on similar tasks.

Chapter 8

Conclusion and Future Work

The thesis has demonstrated to the reader a number of novel data-driven systems that are able to generate multilingual text that summarises a set of input knowledge base triples. The generated text can be used to effectively increase the accessibility of people who are unfamiliar either with the underlying technologies or with the more popular (and, thus, better covered) languages to the information that is stored in a structured knowledge base. This chapter summarises the results and contributions of this thesis, and highlights future work with respect to all of the previous contribution chapters.

8.1 Summary and Conclusions

Training data for NLG is not always readily available; this applies to Semantic Web scenarios as well. In the fourth chapter, we proposed a fully-automatic, cross-lingual, approach for building large corpora of loosely aligned Wikipedia snippets with triples from DBpedia and Wikidata. Using this methodology, we built five different corpora that enabled us to test our system against a variety of generative scenarios, including single- and open-domain summaries generation in three different languages (i.e English, Arabic and Esperanto).

In Chapter 5, an end-to-end trainable architecture based on the general encoder-decoder framework is presented. The system encodes the information from a set of triples into a vector of fixed dimensionality using a novel triple encoder and generates a summary by conditioning the output on the encoded vector. We train and evaluate the performance of our approach on English biographies generation. Our hypothesis was that biographies offer a good trade-off between linguistic variability and regular structure allowing us to explore the strengths and limitations of our purely data-driven approach. Each summary consists from regular words and realisation of entities from the triples. Consequently, we explored a set of different approaches that enable our models to verbalise entities

from the input set of triples in the generated text. 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 fluency of the generated summaries compared to a deterministic replacement of the generated entities' URIs. Results using methods for both automatic and human evaluation (i.e. using experts) generate fluent summaries that address around 50% of the input triples in the generated text. The very low percentage of contradiction (around 0.05%) between the generated summary and its corresponding triples reflect that our system properly conditions the generated content to the given input.

Given the promising results of our systems in the generation of biographies, in Chapter 6, we wished to explore the applicability of our approach to languages that do not offer the same training data abundance as English. We leverage the cross-lingual nature of Wikidata, and we adapt the Triples2GRU w/ Surf. Form Tuples system in order to generate textual summaries in two underserved Wikipedia languages, Arabic and Esperanto. We introduce the mechanism of property placeholders that enables the system to verbalise rare entities in the text. Each property placeholder that is generated is mapped to the triple with which it shares the same property, and is subsequently replaced with the textual label of the entity. Results based on automatic evaluation show that our approach was able to outperform and generalise across domains better than strong baselines of different natures, including MT, a template-based IR baseline and a templated Kneser-Ney language model. Our original goal was to evaluate whether the generated content could be used to enhance the coverage of the impoverished Wikipedias and to provide their corresponding editors with a "starting point" to write their article. Consequently, besides evaluating our approach with automatic evaluation metrics, we conducted two separate community studies, one for the readers and one for the editors of the involved Wikipedias. The readers of the targeted communities ranked our generated text close to the expected standards of Wikipedia, and were also likely to assume the generated text is part of Wikipedia. In the second study, we asked Wikipedia editors to write a short paragraph about an entity and we provided them with its corresponding triples and the automatically generated summary from our system. We found that the editors were likely to reuse a large portion of the generated summaries, thus, emphasizing the usefulness of our approach to the involved communities.

Finally, in Chapter 7, we enhance the Triples2GRU with a pointer mechanism that enables it 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. The suggested mechanism essentially tackles two limitations associated with the original system: (i) the degree of stochasticity in the case that multiple relations from the input meet the requirements of the predicted placeholders, since the model itself makes the final prediction about the triple that should be realised, and (ii) the inability to handle different numbers, since it can directly copy them in the text from the triples. We test the new

system on both biographies and open-domain Wikipedia summaries generation, and we compare its performance to the Triples2LSTM and Triples2GRU from Chapter 5, both equipped with the surface form tuples mechanism. We also investigate two alterations of the proposed pointer-generator network one which is equipped with surface form tuples and one without. Using methods for both automatic and human evaluation (using crowdsourcing), we showed that the proposed pointer mechanism results in performance improvements in all scenarios. We found that the inclusion of surface form tuples makes the model more flexible at addressing the large vocabulary sizes of open-domain corpora.

In the last section of Chapter 7, we also explored whether results of human evaluation using both experts and crowdsourcing are comparable to each other. We tested a sample of the generated summaries that had been evaluated using crowdsourcing, and we found differences between the scores that are provided by experts and crowdworkers. These differences are more notable in the case of the less trivial evaluation criteria (i.e. contradiction). We believe that this observation gives ground to future work regarding identifying the type of properties that influence negatively the workers' judgement and suggesting the methods with which the crowdworkers should be trained in order to perform more accurately on similar tasks.

8.2 Current Limitations and Future Work

In this section we present some of the existing limitation of our approach. We focus on the general problem of generating multi-sentence summaries and we discuss techniques that could enable our approach to scale to this domain. Finally, we propose future work that could build on top of the research that has been presented in this thesis.

8.2.1 Generation of Multi-Sentence Summaries

The performance of our systems is dependent on the number of input triples (see Figure 5.3 and 7.2). In all scenarios, 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 oversized triple sets. This is mainly due to the original requirement that the space complexity of the task should be contained in a single GPU. Consequently, we set upper bounds with respect to the number of allocated triples per summary (Eq. 5.10 and 7.13). Based on these upper bounds, we apply a simple approach of eliminating redundant triples (described in detail in Section 5.2.2), but in case their number still exceeds the threshold, we agnostically use an $E_{\max}$ number of them. This introduces a degree of stochasticity with respect to the quality of the training data with which the model is provided since the $E_{\max}$ triples that we retain from a very large set of triples might not be reflected in their corresponding summary. This is observed even

in the case of the experiments presented in Chapter 7, where we prioritise triples whose subjects or objects have been mentioned in the text, since the average number of tokens in the summaries that are recognised as realisations of entities or years from the input is relatively low.

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 should be provided with all the information that it requires in order to generate a summary that matches the expectation of a dataset; (ii) the decoder would have to retain the information from the input at very distant timesteps.

A simple but relatively expensive method of addressing the first challenge is the parallelisation of the model across multiple GPUs[1] allowing us to increase the maximum number of expected input triples further. A much more sophisticated approach would be to allow the model to "select" from an oversized set the most appropriate triples. Selection of triples is already actively performed by all the proposed systems, but only on the basis of the $E_{\max}$ provided inputs. In the case of the encoder-decoder architecture without attention, this selection is carried out in a single step, and it can only be observed empirically in the generated summaries. However, the pointer-generator network generates 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 might be 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 eventually stops when no further improvement in the automatic evaluation metrics is observed.

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 (Tu et al., 2016; Mi et al., 2016). 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. 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 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 future timesteps. Consequently the existing attention mechanism with which the pointer-generator network is equipped allow us to explore in future work to what extent monitoring the

[1] https://github.com/torch/cutorch

coverage of the generated text is required in a triples-to-multi-sentence-summaries scenario.

8.2.2 The Main Entity of Interest

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.

8.2.3 Using the S3 Corpus

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. In Chapter 4, we briefly explored an approach for generating high quality data-to-text corpora using crowdsourcing. While such an approach offers much better guarantees for the alignment of triples with the corresponding text, it is costly to apply on large domains. An extension of this work is to collect enough data to train a system that would perform the same task efficiently, reducing the required manual interventions to only minor grammar checking. Since S3 is absolved from the notion of the main entity of interest, we believe that building such a corpus could be a reasonable first step for the generation of narratives. Nonetheless, a large, high quality corpus of knowledge base triples aligned with text would be of great value to other research domains alongside NLG, such as Relation Extraction and Question Answering.

Bibliography

Gabor Angeli, Percy Liang, and Dan Klein. 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, 2010. Association for Computational Linguistics.

Mercedes Arguello, Julio Des, Maria Jesus Fernandez-Prieto, Rogelio Perez, and Stavros Lekkas. 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*, pages 366–371, Nov 2011.

Ebru Arisoy, Tara N. Sainath, Brian Kingsbury, and Bhuvana Ramabhadran. Deep neural network language models. In *Proceedings of the NAACL-HLT 2012 Workshop: Will We Ever Really Replace the N-gram Model? On the Future of Language Modeling for HLT*, WLM '12, pages 20–28, Stroudsburg, PA, USA, 2012. Association for Computational Linguistics.

Ibrahim Badr, Rabih Zbib, and James Glass. Segmentation for english-to-arabic statistical machine translation. In *Proceedings of ACL-08: HLT, Short Papers*, pages 153–156. Association for Computational Linguistics, 2008.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473, 2014.

Regina Barzilay and Mirella Lapata. Collective content selection for concept-to-text generation. In *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing*, HLT '05, pages 331–338, Stroudsburg, PA, USA, 2005. Association for Computational Linguistics.

Regina Barzilay and Mirella Lapata. Aggregation via set partitioning for natural language generation. In *Proceedings of the Main Conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*, HLT-NAACL '06, pages 359–366, Stroudsburg, PA, USA, 2006. Association for Computational Linguistics.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. A neural probabilistic language model. *J. Mach. Learn. Res.*, 3:1137–1155, March 2003. ISSN 1532-4435.

Yoshua Bengio, Patrice Simard, and Paolo Frasconi. Learning long-term dependencies with gradient descent is difficult. *Trans. Neur. Netw.*, 5(2):157–166, March 1994. ISSN 1045-9227.

Steven Bird, Ewan Klein, and Edward Loper. *Natural Language Processing with Python*. O'Reilly Media, Inc., 1st edition, 2009. ISBN 0596516495, 9780596516499.

Christopher M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, Inc., New York, NY, USA, 1995. ISBN 0198538642.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3:993–1022, March 2003. ISSN 1532-4435.

Ondřej Bojar, Rajen Chatterjee, Christian Federmann, Yvette Graham, Barry Haddow, Shujian Huang, Matthias Huck, Philipp Koehn, Qun Liu, Varvara Logacheva, Christof Monz, Matteo Negri, Matt Post, Raphael Rubino, Lucia Specia, and Marco Turchi. Findings of the 2017 conference on machine translation (WMT17). In *Proceedings of the Second Conference on Machine Translation, Volume 2: Shared Task Papers*, pages 169–214, Copenhagen, Denmark, September 2017. Association for Computational Linguistics.

Ondej Bojar, Christian Federmann, Mark Fishel, Yvette Graham, Barry Haddow, Philipp Koehn, and Christof Monz. Findings of the 2018 conference on machine translation (WMT18). In *Proceedings of the Third Conference on Machine Translation: Shared Task Papers*, pages 272–303, Belgium, Brussels, October 2018. Association for Computational Linguistics.

Nadjet Bouayad-Agha, Gerard Casamayor, Simon Mille, and Leo Wanner. Perspective-oriented generation of football match summaries: Old tasks, new challenges. *ACM Trans. Speech Lang. Process.*, 9(2):3:1–3:31, August 2012. ISSN 1550-4875.

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

David L. Chen, Joohyun Kim, and Raymond J. Mooney. Training a multilingual sportscaster: Using perceptual context to learn language. *J. Artif. Int. Res.*, 37: 397–435, 2010. ISSN 1076-9757.

David L. Chen and Raymond J. Mooney. 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, 2008. ACM. ISBN 978-1-60558-205-4.

Qian Chen, Xiao-Dan Zhu, Zhen-Hua Ling, Si Wei, and Hui Jiang. Distraction-based neural networks for modeling document. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA, 9-15 July 2016*, pages 2754–2760, 2016.

Stanley F. Chen and Joshua Goodman. An empirical study of smoothing techniques for language modeling. In *Proceedings of the 34th Annual Meeting on Association for Computational Linguistics*, ACL '96, pages 310–318, Stroudsburg, PA, USA, 1996. Association for Computational Linguistics.

Andrew Chisholm, Will Radford, and Ben Hachey. 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, April 2017. Association for Computational Linguistics.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 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, October 2014. Association for Computational Linguistics.

Junyoung Chung, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *CoRR*, abs/1412.3555, 2014.

Paul Clough, Robert Gaizauskas, Scott S.L. Piao, and Yorick Wilks. Measuring text reuse. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*, pages 152–159, Philadelphia, Pennsylvania, USA, July 2002. Association for Computational Linguistics.

Deborah A. Dahl, Madeleine Bates, Michael Brown, William Fisher, Kate Hunicke-Smith, David Pallett, Christine Pao, Alexander Rudnicky, and Elizabeth Shriberg. Expanding the scope of the ATIS task: The ATIS-3 corpus. In *Proceedings of the Workshop on Human Language Technology*, HLT '94, pages 43–48, Stroudsburg, PA, USA, 1994. Association for Computational Linguistics. ISBN 1-55860-357-3.

Joachim Daiber, Max Jakob, Chris Hokamp, and Pablo N. Mendes. 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, 2013. ACM. ISBN 978-1-4503-1972-0.

Dana Dannélls, Mariana Damova, Ramona Enache, and Milen Chechev. 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, 2012. ACM. ISBN 978-1-4503-1230-1.

Li Dong and Mirella Lapata. 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, August 2016. Association for Computational Linguistics.

Xinya Du, Junru Shao, and Claire Cardie. Learning to ask: Neural question generation for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1342–1352, Vancouver, Canada, July 2017. Association for Computational Linguistics.

Pablo Duboue and Kathleen McKeown. Content planner construction via evolutionary algorithms and a corpus-based fitness function. In *Proceedings of the International Natural Language Generation Conference*, pages 89–96. Association for Computational Linguistics, 2002.

Daniel Duma and Ewan Klein. Generating natural language from linked data: Unsupervised template extraction. In *Proceedings of the 10th International Conference on Computational Semantics (IWCS 2013) – Long Papers*, pages 83–94, Potsdam, Germany, March 2013. Association for Computational Linguistics.

Basil Ell and Andreas Harth. 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., June 2014. Association for Computational Linguistics.

Jeffrey L. Elman. Finding structure in time. *Cognitive Science*, 14(2):179–211, 1990.

Hady Elsahar, Pavlos Vougiouklis, Arslen Remaci, Christophe Gravier, Jonathon Hare, Frederique Laforest, and Elena Simperl. T-REx: A Large Scale Alignment of Natural Language with Knowledge Base Triples. In Nicoletta Calzolari (Conference chair), Khalid Choukri, Christopher Cieri, Thierry Declerck, Sara Goggi, Koiti Hasida, Hitoshi Isahara, Bente Maegaard, Joseph Mariani, Hlne Mazo, Asuncion Moreno, Jan Odijk, Stelios Piperidis, and Takenobu Tokunaga, editors, *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan, May 7-12, 2018 2018. European Language Resources Association (ELRA). ISBN 979-10-95546-00-9.

Dimitrios Galanis and Ion Androutsopoulos. Generating multilingual descriptions from linguistically annotated owl ontologies: the naturalowl system. In *Proceedings of the Eleventh European Workshop on Natural Language Generation (ENLG 07)*, 2007.

Claire Gardent, Anastasia Shimorina, Shashi Narayan, and Laura Perez-Beltrachini. Creating training corpora for NLG micro-planners. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 179–188, Vancouver, Canada, July 2017a. Association for Computational Linguistics.

Claire Gardent, Anastasia Shimorina, Shashi Narayan, and Laura Perez-Beltrachini. 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, 2017b.

Felix A. Gers, Nicol N. Schraudolph, and Jürgen Schmidhuber. Learning precise timing with LSTM recurrent networks. *J. Mach. Learn. Res.*, 3:115–143, March 2003. ISSN 1532-4435.

Alex Graves. Generating sequences with recurrent neural networks. *CoRR*, abs/1308.0850, 2013.

Alex Graves and Jürgen Schmidhuber. Offline handwriting recognition with multidimensional recurrent neural networks. In D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, editors, *Advances in Neural Information Processing Systems 21*, pages 545–552. Curran Associates, Inc., 2009.

Nancy Green. 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, 2006. Association for Computational Linguistics. ISBN 1-932432-72-8.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor O.K. Li. 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, August 2016. Association for Computational Linguistics.

Nizar Habash and Owen Rambow. Arabic tokenization, part-of-speech tagging and morphological disambiguation in one fell swoop. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 573–580. Association for Computational Linguistics, 2005.

Nathan Halko, Per-Gunnar Martinsson, and Joel A. Tropp. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM Rev.*, 53(2):217–288, May 2011. ISSN 0036-1445.

Kenneth Heafield, Ivan Pouzyrevsky, Jonathan H. Clark, and Philipp Koehn. 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, August 2013. Association for Computational Linguistics.

Brent Hecht and Darren Gergle. The tower of babel meets web 2.0: User-generated content and its applications in a multilingual context. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '10, pages 291–300, New York, NY, USA, 2010. ACM. ISBN 978-1-60558-929-9.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. Teaching machines to read and comprehend. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 1693–1701. Curran Associates, Inc., 2015.

Sepp Hochreiter, Yoshua Bengio, and Paolo Frasconi. Gradient flow in recurrent nets: the difficulty of learning long-term dependencies. In J. Kolen and S. Kremer, editors, *Field Guide to Dynamical Recurrent Networks*. IEEE Press, 2001.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, November 1997. ISSN 0899-7667.

Sepp Hochreiter and Jrgen Schmidhuber. Bridging long time lags by weight guessing and "long short term memory". In *Spatiotemporal Models in Biological and Artificial Systems*, pages 65–72. IOS Press, 1996.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In Francis Bach and David Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 448–456, Lille, France, 07–09 Jul 2015. PMLR.

Sabine Janzen and Wolfgang Maass. Ontology-based natural language processing for in-store shopping situations. In *Proceedings of the 2009 IEEE International Conference on Semantic Computing*, ICSC '09, pages 361–366, Washington, DC, USA, 2009. IEEE Computer Society. ISBN 978-0-7695-3800-6.

Thorsten Joachims. A probabilistic analysis of the rocchio algorithm with tfidf for text categorization. In *Proceedings of the Fourteenth International Conference on Machine Learning*, ICML '97, pages 143–151, San Francisco, CA, USA, 1997. Morgan Kaufmann Publishers Inc. ISBN 1-55860-486-3.

Lucie-Aimée Kaffee, Alessandro Piscopo, Pavlos Vougiouklis, Elena Simperl, Leslie Carr, and Lydia Pintscher. A glimpse into babel: An analysis of multilinguality in wikidata. In *Proceedings of the 13th International Symposium on Open Collaboration*, OpenSym '17, pages 14:1–14:5, New York, NY, USA, 2017. ACM. ISBN 978-1-4503-5187-4.

Andrej Karpathy and Li Fei-Fei. Deep visual-semantic alignments for generating image descriptions. *IEEE Trans. Pattern Anal. Mach. Intell.*, 39(4):664–676, April 2017. ISSN 0162-8828.

Andrej Karpathy, Justin Johnson, and Fei-Fei Li. Visualizing and understanding recurrent networks. *CoRR*, abs/1506.02078, 2015.

Joohyun Kim and Raymond J. Mooney. 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, 2010. Association for Computational Linguistics.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

Philipp Koehn and Rebecca Knowles. Six challenges for neural machine translation. In *Proceedings of the First Workshop on Neural Machine Translation*, pages 28–39, Vancouver, August 2017. Association for Computational Linguistics.

John Kolen and Stefan Kremer. *Gradient Flow in Recurrent Nets: The Difficulty of Learning Long-Term Dependencies*, pages 237–243. Wiley-IEEE Press, 2001. ISBN 9780470544037.

Ioannis Konstas and Mirella Lapata. 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, 2012a. Association for Computational Linguistics.

Ioannis Konstas and Mirella Lapata. 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, June 2012b. Association for Computational Linguistics.

Ioannis Konstas and Mirella Lapata. A global model for concept-to-text generation. *J. Artif. Int. Res.*, 48(1):305–346, October 2013. ISSN 1076-9757.

Alon Lavie and Abhaya Agarwal. 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, 2007. Association for Computational Linguistics.

Niklas Laxström, Pau Giner, and Santhosh Thottingal. Content translation: Computer assisted translation tool for wikipedia articles. In Îlknur Durgar El-Kahlout, Mehmed Özkan, Felipe Sánchez-Martínez, Gema Ramírez-Sánchez, Fred Hollowood, and Andy Way, editors, *Proceedings of the 18th Annual Conference of the European Association for Machine Translation*, pages 194–197, Antalya, Turkey, May 2015.

Rémi Lebret, David Grangier, and Michael Auli. 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, 2016.

James C. Lester and Bruce W. Porter. Developing and empirically evaluating robust explanation generators: The knight experiments. *Comput. Linguist.*, 23(1):65–101, March 1997. ISSN 0891-2017.

Vladimir I. Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics Doklady*, 10(8):707–710, February 1966.

David C.S. Li. Between English and Esperanto: what does it take to be a world language? *International Journal of the Sociology of Language*, 2003(164):33–63, 2006.

Percy Liang, Michael I. Jordan, and Dan Klein. 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, 2009. Association for Computational Linguistics. ISBN 978-1-932432-45-9.

Chin-Yew Lin. ROUGE: A package for automatic evaluation of summaries. In Stan Szpakowicz Marie-Francine Moens, editor, *Text Summarization Branches Out: Proceedings of the ACL-04 Workshop*, pages 74–81, Barcelona, Spain, July 2004. Association for Computational Linguistics.

Wei Lu, Hwee Tou Ng, Wee Sun Lee, and Luke S. Zettlemoyer. 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, October 2008. Association for Computational Linguistics.

Thang Luong, Hieu Pham, and Christopher D. Manning. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421, Lisbon, Portugal, September 2015a. Association for Computational Linguistics.

Thang Luong, Ilya Sutskever, Quoc Le, Oriol Vinyals, and Wojciech Zaremba. 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, July 2015b. Association for Computational Linguistics.

James Martens. Deep learning via hessian-free optimization. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10), June 21-24, 2010, Haifa, Israel*, pages 735–742, 2010.

James Martens and Ilya Sutskever. Learning recurrent neural networks with hessian-free optimization. In *Proceedings of the 28th International Conference on Machine Learning, ICML 2011, Bellevue, Washington, USA, June 28 - July 2, 2011*, pages 1033–1040, 2011.

Hermann Mayer, Faustino Gomez, Daan Wierstra, Istvan Nagy, Alois Knoll, and Jurgen Schmidhuber. A system for robotic heart surgery that learns to tie knots using recurrent neural networks. In *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, pages 543–548, Oct 2006.

Hongyuan Mei, Mohit Bansal, and Matthew R. Walter. 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, June 2016. Association for Computational Linguistics.

Chris Mellish and Robert Dale. Evaluation in the context of natural language generation. *Computer Speech & Language*, 12(4):349 – 373, 1998. ISSN 0885-2308.

Haitao Mi, Baskaran Sankaran, Zhiguo Wang, and Abe Ittycheriah. 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, November 2016. Association for Computational Linguistics.

Tomáš Mikolov. *Statistical Language Models Based on Neural Networks*. PhD thesis, 2012.

Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Cernockỳ, and Sanjeev Khudanpur. Recurrent neural network based language model. In *INTERSPEECH 2010, 11th Annual Conference of the International Speech Communication Association, Makuhari, Japan, September 26-30, 2010*, pages 1045–1048, 2010.

Tomáš Mikolov, Stefan Kombrink, Lukáš Burget, Jan Honza Černockỳ, and Sanjeev Khudanpur. Extensions of recurrent neural network language model. In *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on*, pages 5528–5531, May 2011.

Tomáš Mikolov, Ilya Sutskever, Anoop Deoras, Hai-Son Le, Stefan Kombrink, and Jan Cernocky. Subword language modeling with neural networks. Technical report, 2012. Unpublished Manuscript.

Mike Mintz, Steven Bills, Rion Snow, and Daniel Jurafsky. Distant supervision for relation extraction without labeled data. 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*, pages 1003–1011, Suntec, Singapore, August 2009. Association for Computational Linguistics.

Yassine Mrabet, Pavlos Vougiouklis, Halil Kilicoglu, Claire Gardent, Dina Demner-Fushman, Jonathon Hare, and Elena Simperl. 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), pages 29–36. Association for Computational Linguistics, 2016.

Ramesh Nallapati, Feifei Zhai, and Bowen Zhou. SummaRuNNer: A recurrent neural network based sequence model for extractive summarization of documents, 2017.

Ramesh Nallapati, Bowen Zhou, Cicero dos Santos, Caglar Gulcehre, and Bing Xiang. Abstractive text summarization using sequence-to-sequence rnns and beyond. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 280–290, Berlin, Germany, August 2016. Association for Computational Linguistics.

Axel-Cyrille Ngonga Ngomo, Lorenz Bühmann, Christina Unger, Jens Lehmann, and Daniel Gerber. Sorry, i don't speak SPARQL: Translating SPARQL queries into natural language. In *Proceedings of the 22Nd International Conference on World Wide Web*, WWW '13, pages 977–988, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-2035-1.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. BLEU: A method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, pages 311–318, Stroudsburg, PA, USA, 2002. Association for Computational Linguistics.

Razvan Pascanu, Tomáš Mikolov, and Yoshua Bengio. Understanding the exploding gradient problem. *CoRR*, abs/1211.5063, 2012.

Romain Paulus, Caiming Xiong, and Richard Socher. A deep reinforced model for abstractive summarization. In *International Conference on Learning Representations*, 2018.

Ehud Reiter. *Natural Language Generation*, chapter 20, pages 574–598. Wiley-Blackwell, 2010. ISBN 9781444324044.

Ehud Reiter and Anja Belz. An investigation into the validity of some metrics for automatically evaluating natural language generation systems. *Comput. Linguist.*, 35 (4):529–558, December 2009. ISSN 0891-2017.

Ehud Reiter and Robert Dale. *Building Natural Language Generation Systems*. Cambridge University Press, New York, NY, USA, 2000. ISBN 0-521-62036-8.

Ehud Reiter, Roma Robertson, and Somayajulu Sripada. Acquiring correct knowledge for natural language generation. *J. Artif. Int. Res.*, 18:491–516, 2003. ISSN 1076-9757.

Ehud Reiter, Somayajulu Sripada, Jim Hunter, Jin Yu, and Ian Davy. Choosing words in computer-generated weather forecasts. *Artif. Intell.*, 167(1-2):137–169, September 2005. ISSN 0004-3702.

David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323:533 EP –, Oct 1986.

Alexander M. Rush, Sumit Chopra, and Jason Weston. 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, September 2015. Association for Computational Linguistics.

Christina Sauper and Regina Barzilay. Automatically generating wikipedia articles: A structure-aware approach. 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*, pages 208–216, Suntec, Singapore, August 2009. Association for Computational Linguistics.

Abigail See, Peter J. Liu, and Christopher D. Manning. 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, 2017.

Iulian Vlad Serban, Alberto García-Durán, Caglar Gulcehre, Sungjin Ahn, Sarath Chandar, Aaron Courville, and Yoshua Bengio. 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)*, pages 588–598, Berlin, Germany, August 2016. Association for Computational Linguistics.

Nigel Shadbolt, Tim Berners-Lee, and Wendy Hall. The semantic web revisited. *IEEE Intelligent Systems*, 21(3):96–101, May 2006. ISSN 1541-1672.

Lifeng Shang, Zhengdong Lu, and Hang Li. 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)*, pages 1577–1586, Beijing, China, July 2015. Association for Computational Linguistics.

Amin Sleimi and Claire Gardent. 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, 2016.

Richard Socher and Christopher Manning. Deep learning for natural language processing. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, Atlanta, Georgia, June 2013. Association for Computational Linguistics.

Alessandro Sordoni, Michel Galley, Michael Auli, Chris Brockett, Yangfeng Ji, Margaret Mitchell, Jian-Yun Nie, Jianfeng Gao, and Bill Dolan. A neural network approach to context-sensitive generation of conversational responses. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 196–205, Denver, Colorado, May–June 2015. Association for Computational Linguistics.

Somayajulu G. Sripada, Ehud Reiter, Jim Hunter, and Jin Yu. SUMTIME-METEO: Parallel Corpus of Naturally Occurring Forecast Texts and Weather Data. Technical Report AUCS/TR0201, Computing Science Department, University of Aberdeen, 2002.

Xiantang Sun and Chris Mellish. An experiment on "free generation" from single rdf triples. In *Proceedings of the Eleventh European Workshop on Natural Language Generation*, pages 105–108, Saarbrücken, Germany, June 2007. DFKI GmbH. Document D-07-01.

Martin Sundermeyer, Hermann Ney, and Ralf Schlüter. From feedforward to recurrent LSTM neural networks for language modeling. *Trans. Audio, Speech and Lang. Proc.*, 23(3):517–529, March 2015. ISSN 1063-6676.

Martin Sundermeyer, Ralf Schlüter, and Hermann Ney. LSTM neural networks for language modeling. In *INTERSPEECH 2012, 13th Annual Conference of the International Speech Communication Association, Portland, Oregon, USA, September 9-13, 2012*, pages 194–197, 2012.

Ilya Sutskever, James Martens, and Geoffrey Hinton. Generating text with recurrent neural networks. In Lise Getoor and Tobias Scheffer, editors, *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, ICML '11, pages 1017–1024, New York, NY, USA, June 2011. ACM. ISBN 978-1-4503-0619-5.

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

Tijmen Tieleman and Geoffrey E. Hinton. RMSProp: Divide the gradient by a running average of its recent magnitude, 2012.

Zhaopeng Tu, Zhengdong Lu, Yang Liu, Xiaohua Liu, and Hang Li. 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, August 2016. Association for Computational Linguistics.

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

O. Vinyals, A. Toshev, S. Bengio, and D. Erhan. Show and tell: A neural image caption generator. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3156–3164, June 2015a.

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

Oriol Vinyals, ukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey Hinton. Grammar as a foreign language. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 2773–2781. Curran Associates, Inc., 2015c.

Oriol Vinyals and Quoc V. Le. A neural conversational model. *CoRR*, abs/1506.05869, 2015.

Pavlos Vougiouklis, Hady Elsahar, Lucie-Aime Kaffee, Christophe Gravier, Frdrique Laforest, Jonathon Hare, and Elena Simperl. Neural wikipedian: Generating textual summaries from knowledge base triples. *Journal of Web Semantics*, 2018a. ISSN 1570-8268.

Pavlos Vougiouklis, Jonathon Hare, and Elena Simperl. A neural network approach for knowledge-driven response generation. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 3370–3380, Osaka, Japan, December 2016. The COLING 2016 Organizing Committee.

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

Tsung-Hsien Wen, Milica Gasic, Nikola Mrkšić, Pei-Hao Su, David Vandyke, and Steve Young. Semantically conditioned LSTM-based natural language generation for spoken dialogue systems. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1711–1721, Lisbon, Portugal, September 2015. Association for Computational Linguistics.

Tsung-Hsien Wen, Milica Gašić, Nikola Mrkšić, Lina M. Rojas-Barahona, Pei-Hao Su, David Vandyke, and Steve Young. 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*, pages 120–129, San Diego, California, June 2016. Association for Computational Linguistics.

Sandra Williams and Ehud Reiter. Generating basic skills reports for low-skilled readers. *Natural Language Engineering*, 14(4):495525, 2008.

Michael J. Wise. Yap3: Improved detection of similarities in computer program and other texts. In *Proceedings of the Twenty-seventh SIGCSE Technical Symposium on Computer Science Education*, SIGCSE '96, pages 130–134, New York, NY, USA, 1996. ACM. ISBN 0-89791-757-X.

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

Wojciech Zaremba and Ilya Sutskever. Learning to execute. *CoRR*, abs/1410.4615, 2014.