POLITECNICO DI TORINO

Master's Degree in Mathematical Engineering

Master's Degree Thesis

Hybrid Neural Knowledge Graph-to-Text and Text-to-Text Generation

Academic supervisor
prof. Tatiana Tommasi
External supervisor
prof. Leo Wanner

Candidate
Marco Saponara

A.Y. 2021-2022

Summary

In this document, we propose a novel Natural Language Generation (NLG) task, namely the hybrid knowledge graph-to-text and text-to-text generation. It aims at enriching the text obtained from a knowledge graph (KG) encoded in the format of the Resource Description Framework (RDF) with relevant information extrapolated from a complementary textual context. This task is particularly useful when dealing with small-sized ontologies on topics for which richer textual resources are available.

In order to solve this task, we present a neural system based on a three-step pipeline: pure KG-to-text generation, content selection from the context, and, finally, the combination of the KG's verbalization and the additional information into a fluent and cohesive textual output. Each step is based on the Transformer (Vaswani et al. [2017]) architecture, with the first and the third steps employing a suitably fine-tuned T5 model (Raffel et al. [2020]), and the second based on BERT (Devlin et al. [2019]).

The KG-to-text generation model is fine-tuned on the WebNLG corpus (Gardent et al. [2017a]); the others are trained on two custom datasets derived from the latter.

The generated texts are then evaluated through the syntactic log-odds ratio (SLOR, Kann et al. [2018]), a referenceless model-dependent metric for fluency evaluation, and a questionnaire-based human evaluation on four dimensions, namely coherence, grammaticality, faithfulness, and informativeness. The generated texts overall reach good levels of grammatical correctness and informativeness, but there is room for improvement with regard to textual coherence and faithfulness.

Acknowledgements

I would like to thank my supervisors, Tatiana Tommasi, for her willingness to supervise my thesis, and Leo Wanner, for the constant attention I received throughout my work. I would also like to thank Simon, Alex and Alba for the support (both technical and moral), and Universitat Pompeu Fabra in Barcelona for welcoming me and providing me an outstanding work environment.

Thank you to my beautiful Spanish flatmates, Montse, Carla, and Carmen, who took care of me. I miss our quality time in the house. Thank you to Dani, Alexia, Timi and Vincent for lightening my stay in Barcelona. I miss you as well.

A special thanks goes to Luca, *a buddy who is more than a brother and a little less than a wife*, and Emma, simply the most important person in my life.

Finally, I would like to express my gratitude to my family for the support I received throughout my life.

Contents

List of Tables

List of Figures

Si un hombre nunca se contradice,
sera porque nunca dice nada.

[Miguel De Unamuno]

Chapter 1

Introduction

In recent years, Natural Language Generation (NLG) has drawn more and more attention, to the point that an increasingly vast amount of research is now exploring complex software systems able to produce texts that are almost indistinguishable from human-written texts on several tasks, as well as in varied domains. In fact, nowadays neural-network-based models allow to overcome the limitations of both text preprocessing, by operating on raw forms of textual input, and shallow learning methods, which typically rely only on a limited set of fixed rules and basic statistical inference.

The present thesis focuses on the task of hybrid neural generation from knowledge graphs and complementary textual material. The concept of knowledge graph (KG, Ehrlinger and Wöß [2016]), also known as semantic network, has emerged as a compelling abstraction for organizing world's structured knowledge, as well as a way to integrate information extracted from multiple data sources. In fact, it represents a network of real-world entities (i.e. objects, events, situations, or concepts) and illustrates the relationship between them. A knowledge graph is made up of three main components: nodes, edges, and labels. Any person, place or object can be a node, while the relationship between two nodes is defined by a specific edge. This network can be managed through the means of the Resource Description Framework (RDF) data model (Section 2.1) and, in particular, through its atomic data entity, the semantic triple (or RDF triple). As its name indicates, a triple is a set of three entities that codifies a statement about semantic data in the form of subject–predicate–object expressions. Each triple represents a single fact, so that a knowledge graph is basically equivalent to a set of RDF triples.

At the basis of our project, then, there is the idea to enrich and expand the core text generated from an input knowledge graph by performing a targeted content selection over an additional *context*, which is composed by at least one textual document related to the subjects involved in the knowledge graph. Our main contributions are the proposal and the implementation of a pipeline to address the task previously stated, along with the building of two novel datasets useful for the training and evaluation phases.

The rest of this chapter is organized as follows. In Section 1.1, we present an overview of the evolution of the approaches developed in order to deal with NLG. Secondly, in Section 1.2 we provide a formal and detailed description of the object of the present work, along with the motivation at its basis and the major issues involved (Sections 1.3 and 1.4,

respectively). Finally, we dedicate the last Section, 1.5, to present the structure of the thesis document.

1.1 Brief review of Natural Language Generation

NLG, a subfield of the broader area Natural Language Processing (NLP), is a software process that produces natural language output (e.g., explanations, narratives, summaries, etc.). It can be compared to the process humans execute when they turn ideas into writing or speech. As a consequence, NLG systems have been used for many real-world applications, such as generating weather forecasts, conducting interactive conversations with humans in spoken dialog systems (chatbots), captioning images or visual scenes, translating text from one language to another, and generating stories and news articles.

The first approaches of automatic NLG systems relied on rule-based pipeline methods (Reiter and Dale [2000]). The generation process is typically decomposed to stages. First, there is the *document planning*, which determines the content and its order and generates a text plan outlining the structure of messages. Then, we have the *micro-planning* stage, when referring expressions that identify objects like entities or places are generated, along with the choice of words to be used and how they are aggregated. Collating similar sentences to improve readability with a natural flow also occurs in this stage. Finally, we have the *realization*, in which the actual text is generated, using linguistic knowledge about morphology, syntax, semantics, etc.

There is also a large body of early work regarding template-based models. These methods, unlike the previous ones, map their non-linguistic input directly (i.e., without intermediate representations) to the linguistic surface structure. Crucially, this linguistic structure may contain gaps; well-formed output results when the gaps are filled or, more precisely, when all the gaps have been replaced by linguistic structures that do not contain gaps.

The major drawbacks of the latter systems mainly lie in depending heavily on the domain and on the application they are designed for and, consequently, in the limited range of sentences they can produce. Nevertheless, these earlier text generation approaches and their extensions played an important role in the evolution of NLG research. Following these works, an important direction that several NLG researchers have focused on is data-driven representation learning, which has gained popularity with the availability of more data sources. Availability of large datasets, knowledge bases, corpora of referring expressions, as well as shared tasks have been beneficial in the progress of several NLG tasks today. In fact, the last decade has witnessed a paradigm shift towards learning representations from large textual corpora in an unsupervised manner, by training deep neural network (DNN) models on very large corpora of human-written texts.

The paradigm shift started with the introduction of models relying on machine learning. Then, it continued with the advent of deep learning, particularly with the use of recurrent neural networks (RNNs, Graves [2013]), e.g. long short-term memory networks (LSTMs, Hochreiter and Schmidhuber [1997]) and gated recurrent units (GRUs, Cho et al. [2014]), for learning language representations, and later with sequence-to-sequence learning, which opened up a new chapter characterised by the wide application of the encoder-decoder

architecture.

Although sequence-to-sequence models were originally developed for machine translation, they were soon shown to improve performance across many NLG tasks. Among these architectures, the Transformer (Vaswani et al. [2017]), which incorporates an encoder and a decoder, both implemented using the self-attention mechanism, provided new state-of-the-art performances. In recent years, a large body of research has dealt with the study and the improvement of Transformer-based models, e.g., by increasing their complexity or using better sampling methods to reduce degeneration in decoding.

Let us now conclude this section by listing some of the most relevant NLG tasks treated with neural models:

- summarization: it can be divided in several subtasks, e.g., single or multi-document and generic or query-focused summarization;
- machine translation: it aims at translating text or speech from a source language to a target one;
- dialog response generation: it aims at simulating a conversation between humans and the interactions can be divided in goal-oriented and *chit-chat* (i.e., generic);
- caption generation: input are usually images or video frames;
- data-to-text generation: e.g., table description and KG-to-text generation.

1.2 Problem overview

Our task consists of an extension of the relatively-recent KG-to-text generation problem (Gardent et al. [2017b]) and it could be defined as a *hybrid KG-to-text and text-to-text generation*. More specifically, while the former only focuses on microplanning (that subtask of NLG consisting in mapping a given content to a text verbalizing it), we would like not only to generate the corresponding descriptive text of an input knowledge graph, but also to enrich it with relevant information extracted from a related textual context.

To the best of our knowledge, this task has not been explored yet and the literature regarding this specific topic is absent. Therefore, before going any further, it is crucial to provide a formal definition of the problem. Let us start from the usual notation.

A set of semantic triples is represented as follows:

$$S = \{t_1, t_2, \dots t_{n_S}\}, \quad \text{with} \quad t_i = \langle s_i \mid p_i \mid o_i \rangle ,$$

where s_i, p_i, o_i denote the subject, predicate and object of the i-th triple respectively. Note that the total number of triples may vary, so we denote it with n_S.

Then, we represent the context C, which is a textual document, as a sequence of n words:

$$C = \langle w_1, w_2, \dots, w_n \rangle .$$

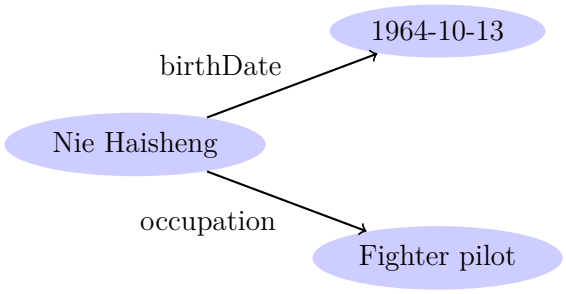

Figure 1.1: Graphical representation of the knowledge graph
{⟨Nie Haisheng | birthDate | 1964-10-13⟩ , ⟨Nie Haisheng | occupation | Fighter pilot⟩}.

Therefore, given a pair (S, C), we aim at generating the target text Y through the means of a software system, f, which depends on a configuration of parameters θ:

$$Y = f(S, C; \theta)$$

where $Y = \langle w_1, w_2, \ldots, w_m \rangle$.

As a simple example, let us consider the following set of RDF triples:

$$\{\langle \text{Nie Haisheng} \mid \text{birthDate} \mid \text{1964-10-13} \rangle , \langle \text{Nie Haisheng} \mid \text{occupation} \mid \text{Fighter pilot} \rangle\} .$$

The corresponding knowledge graph is shown in Figure 1.1. It can be mapped to several equally valid verbalizations. Let us report three of them:

> Nie Haisheng, born on October 13, 1964, worked as a fighter pilot.
> Nie Haisheng is a former fighter pilot who was born on October 13, 1964.
> Nie Haisheng born on 10/13/1964 is a fighter pilot.

Therefore, our work aims at providing an output which should expand and enrich the sentences above through relevant information involving the biography of Nie Haisheng and/or his occupation. A suitable candidate could be the following:

> Nie Haisheng was born on October 13, 1964 in Yangdang Town of Zaoyang County, Xiangyang City, Hubei Province. After graduating from high school he joined the People's Liberation Army Air Force in June 1983, and became a fighter pilot. He trained at the PLAAF's No. 7 Flying School and graduated in 1987. Nie flew on Shenzhou 6 and served as commander on both the Shenzhou 10 and Shenzhou 12 missions, the latter of which became the first crew to visit the Tiangong space station.

We address this task through a pipeline composed of three main steps, namely pure KG-to-text generation, relevant information extraction from the textual context and finally the combination of the intermediate outputs from the previous steps into one cohesive text.

1.3 Motivation

As machine learning advances, it becomes much effective in processing and *understanding* data available on the web. For this reason, the need to complement the current web structure with an extension in which the semantics of information is well defined becomes more and more urgent. The Semantic Web (Berners-Lee et al. [2001]) aims at fulfilling exactly this need. It can be defined as a framework (i.e., a set of formats and languages) that allows data to be shared and reused across application, enterprise, and community boundaries, with the goal to make data on the Internet machine-readable.

Nevertheless, to have complete and rich enough ontologies is always source of problems. In many concrete applications, the available RDF representations cover only a limited amount of facts so that it is desirable to extend the output generated from these representations through further information, which is available in textual format, on some of the most relevant aspects of these representations. This operation would facilitate the generation of informative texts in sufficient detail without the restrictive requirement of representing all the content in terms of RDF triples - which would need a very significant effort to either manually construct the corresponding ontologies or attempt to do it automatically.

One real-world application is, e.g., the generation of personalized recommendations concerning specific administrative procedures, such as, e.g., asylum application for migrants: this procedure depends on several factors, related to the personal situation of migrants, e.g. their age, gender and family status. Having this personal information encoded in RDF, it would be possible to generate a report for an NGO employee that contains not only these personal data, but also recommendations on which procedure to apply and how to proceed (summarized from available text material). A further example would be the generation of stock market reports. In fact, in this particular framework, the RDF representations are often related to real data on some specific stocks but lack background information, e.g. news involving the whole stock market or specific trades between two companies, thus resulting in non-informative and poorly-written reports. This issue can be solved, e.g., by integrating numerical data with relevant information extrapolated from daily news.

1.4 Challenges

Several issues come along with this task. First, one of the major challenges is inherent to the task itself and regards the combination of input at different levels of linguistic abstraction. In fact, the task of mapping sentences to RDF triples and vice versa are delicate problems and complex neural architectures are required to reach state-of-the-art performance (e.g., Gao et al. [2020],Li et al. [2020]).

Secondly, the success of the neural framework heavily relies on the availability of large-scale datasets. Unfortunately, given the novelty of this work, training data is not readily available. This implies that the canonical supervised learning approaches should be replaced with a strategy based on unsupervised or semi-supervised learning. Moreover, the lack of a suitable target reference highlights a serious arbitrariness in this task, in the sense

that the *best possible* result derived from the RDF-driven content selection is not unique: on the contrary, it might be strongly influenced by both subjective and contextual details, leading to a large variety of possible equally-valid outcomes. As a consequence, managing to compare and rank two or more output candidates is definitely challenging, leading to the necessity of employing human evaluation methods instead of the more practical automatic evaluation metrics.

Finally, another contribution to the difficulty of assessing a judgement on the results reached by our work is provided by the lack of documentation on this task. In particular, we are not aware of both baseline and state-of-the-art results.

1.5 Structure of the thesis

The rest of the document is organized as follows:

1. *Background* (Chapter 2), where we present the topics and tools involved throughout our work;
2. *System description* (Chapter 3), where we present our system;
3. *Dataset* (Chapter 4), where we describe the three datasets employed throughout our work;
4. Results (Chapter 5), where we present and discuss the obtained results, as well as their evaluation;
5. *Conclusions* (Chapter 6), where we draw our conclusions, discussing the contributions of this thesis and presenting some points we aim to work on in the future.

Chapter 2

Background

In this chapter, we provide an exhaustive description of the topics and tools involved throughout our work. We start from the Resource Description Framework (Section 2.1), which is crucial to understand the structure of our data. Then, we present the Transformer architecture (Section 2.2), along with two state-of-the-art transformer-based models, i.e. BERT (2.2.1) and T5 (2.2.2), which are at the basis of our system. Finally, in Section 2.3 we discuss the problem of evaluating the quality of a generated text and we provide four examples of evaluation metrics, namely BLEU (2.3.1), ROUGE (2.3.2), BLEURT (2.3.3), and SLOR (2.3.4).

2.1 Resource Description Framework

The Resource Description Framework (RDF) is the basic machine-interpretable information representation format of the Semantic Web. RDF provides a common (graph-based) data format and an identifier scheme that can serve as foundation to unify data from a large number of sources. Data and facts are specified as RDF graph statements with atomic constructs composed of a subject, an object, and a predicate (i.e., the connection between the two), also referred to as triples. Each of the three parts of the statement can be identified by the so-called URI[1]. The main advantage of this simple, flexible data model is the expressive power to represent complex situations, relationships, and other things of interest, while also being appropriately abstract. RDF should be considered for use in situations where:

- multiple source data integration is required without the overhead of a large development effort;
- data will be made available for re-use by stakeholders;

[1]Universal Resource Identifiers (URIs) unambiguously identify arbitrary resources on the Web. HTTP URIs are based on the Webs Domain Name System (DNS) which allows organisations and individuals to register global domain names, which themselves can be used to construct entity identifiers.

- data is available in a decentralised manner, that is, no single stakeholder has responsibility for the entirety of data;
- enhanced use of large amounts of structured data is required (browse, query, match, extract, input).

Data locked in organisational data silos can also benefit from being exposed as RDF without modification, allowing existing information architecture and original data representation remain unaltered. Once the data is exposed, several frameworks can be used to establish and discover relationships between and with other Web accessible Linked Data. Legacy databases can then be queried using the Semantic Web query language, SPARQL. The key point is that RDF use is for establishing a common representation of information contained in other formats to assist combined pre-processing rather than as a replacement of the originating format. The Semantic Web community usually refers to the activity of converting Open Data to RDF as RDFication or RDFizing.

It is important to remark that the RDF data model is not the only option to represent structured data: a valid alternative is provided, for instance, by the Abstract Meaning Representation (AMR, Banarescu et al. [2013]). Similarly to RDF, AMR is a semantic representation language where sentences are represented as rooted, directed, edge-labeled, leaf-labeled graphs (DAGs). They are intended to abstract away from syntactic representations, so that sentences similar in meaning should be assigned the same AMR, even if they are not identically worded. Nevertheless, it is worth mentioning that RDF is an ontology language, i.e., in principle, language independent, while AMR captures linguistic semantics, thus being, strictly speaking, language dependent. Finally, RDF was designed several years before AMR, therefore it has established itself as the most common format.

2.2 Transformer

The Transformer model was introduced in the crucial paper *Attention is all you need* (Vaswani et al. [2017]) and it aims at overcoming the limitations of previous Sequence-to-sequence (Seq2seq) architectures, e.g. recurrent neural networs (RNNs). In fact, RNNs typically factor computations along the symbol positions of the input and output sequences. To process the n-th token, the model combines the state representing the sentence up to the $n-1$-th token with the information of the new token to create a new state, representing the sentence up to token n. This inherently sequential nature precludes parallelization within training examples, which becomes critical at longer sequence lengths, as memory constraints limit batching across examples. Moreover, even if the information from one token could theoretically propagate arbitrarily far down the sequence, several practical issues, e.g. the so-called vanishing gradient problem, do not allow to treat long sequences properly.

In order to deal with the modeling of dependencies without regard to the distance of the sequences, the so-called attention mechanisms has been developed. This technique allows the model to capture the relationships between each part of a sequence by weighing differently each part of the input data according to their relative importance. Nevertheless, in most of the cases the attention mechanism has been integrated inside recurrent models,

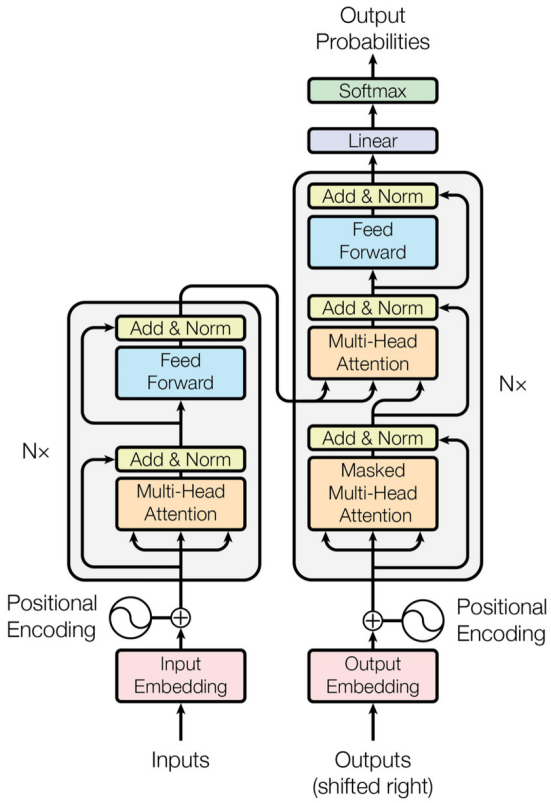

Figure 2.1: The Transformer - model architecture (original picture from Vaswani et al. [2017]).

thus not solving the issues related to the lack of parallelization. The Transformer eschews recurrence and relies entirely on the attention mechanism to draw global dependencies between input and output instead, allowing for significantly more parallelization and, as a consequence, for feeding the model with larger training data.

The Transformer is an Encoder-Decoder model. As the name suggests, this architecture has two main components, showed in the left and right halves of Figure 2.1: the first one, the *encoder*, takes a variable-length sequence $(x_1, \ldots, x_n)$ as the input and transforms it into a continuous representation (i.e. a numerical array) $z = (z_1, \ldots, z_n)$; the *decoder*, instead, maps the encoded state to a variable-length sequence $(y_1, \ldots, y_m)$ of symbols one element at a time. Note that, at each step, the model is *auto-regressive*, i.e. it consumes the previously generated symbols as additional input when generating the next one.

Both the encoder and the decoder are composed of a stack of N identical layers. The function of each encoder layer is to generate encodings that contain information about which parts of the inputs are relevant to each other. The encodings are then passed as inputs to the next encoder layer. Vice versa, each decoder layer takes all the encodings and uses their incorporated contextual information to generate an output sequence. To achieve this, each encoder and decoder layer makes use of a particular attention mechanism, called

Scaled Dot-Product Attention (Figure 2.2).

To have a glimpse on how it works, let us suppose feeding an input sentence to the model. After splitting the sentence into tokens, each token is mapped to a word embedding, i.e. a fixed-length numerical array. The word embedding of the i-th token, x_i, is multiplied with each of the three weight matrices characterizing the attention block (whose entries are learned during the training phase) to produce a query vector, $q_i = x_i W_Q$, a key vector, $k_i = x_i W_K$, and a value vector, $v_i = x_i W_V$. Attention weights are then calculated using the query and key vectors: the attention weight a_{ij} from token i to token j is the dot product between q_i and k_j. The attention weights are divided by the square root of the dimension of the key vectors, $\sqrt{d_k}$, which stabilizes gradients during training, and passed through a softmax function[2], which normalizes the weights. If each embedding vector is stacked into a single matrix X, we can consequently obtain a query matrix Q, a key matrix K, and a value matrix V, so that the whole process is described by the following simple equation:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

Obviously, the gain is not only related to a matter of simpler notation: dot-product attention is much faster and more space-efficient in practice than other methods because it can be implemented using highly optimized matrix multiplication code.

In the real implementation of the Transformer, the queries, keys and values are projected h times with different, learned linear projections to d_k, d_k, and d_v dimensions,

[2] $\text{softmax}(\mathbf{x})_i = \exp(x_i)/\sum_j \exp(x_j)$, where $\mathbf{x} = (x_1, x_2, \ldots, x_d)$.

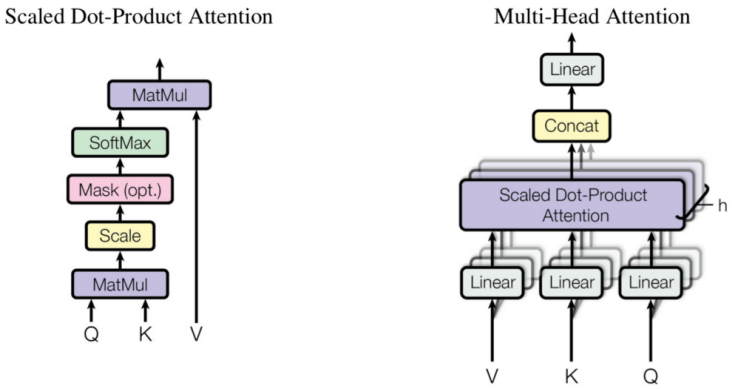

Figure 2.2: (left) Scaled Dot-Product Attention. (right) Multi-Head Attention consists of several attention layers running in parallel (original pictures from Vaswani et al. [2017]).

respectively. On each of these projected versions of queries, keys and values we then perform the attention function in parallel, yielding d_v-dimensional output values. These are concatenated and once again projected, resulting in the final values, as depicted in the right part of Figure 2.2. The authors refer to this mechanism as *Multi-Head Attention*.

In addition to attention sub-layers, each of the layers in the encoder and decoder contains a fully connected feed-forward network, which is applied to each position separately and identically. This consists of two linear transformations with a ReLU activation in between:

$$\mathrm{FFN}(x) = \max(0, xW_1 + b_1)W_2 + b_2.$$

For further details, we suggest to refer to the original paper and follow-up detailed guides[3].

2.2.1 BERT

BERT (Devlin et al. [2019]), which stands for *Bidirectional Encoder Representations from Transformers*, is surely one of the most popular transformer-based architectures and within a few years after its release it became a ubiquitous baseline in NLP experiments.

BERT's model architecture is a multi-layer bidirectional Transformer encoder based on the original implementation (the authors in the paper refer to it as "almost identical" to the original).

BERT is pre-trained using two unsupervised tasks:

1. Masked language modeling (MLM): in order to train a deep bidirectional representation, 15% of the input tokens are randomly masked and the model is trained to predict those masked tokens from the context.

2. Next sentence prediction (NSP): in order to train a model that understands sentence relationships, BERT is pre-trained to predict if a chosen next sentence was probable or not given the first sentence (specifically, when choosing the sentences A and B for building each pre-training example, 50% of the time B is the actual next sentence that follows A, and 50% of the time it is a random sentence from the corpus).

As a result of the training process, BERT learns contextual embeddings for words.

Following the so-called *transfer learning* paradigm, where a model is first pre-trained on a data-rich task before being fine-tuned on a downstream task, after the computationally expensive pre-training BERT can be fine-tuned with less resources on smaller datasets to optimize its performance on specific tasks, often providing state-of-the-art results.

2.2.2 T5

As previously mentioned, since the pre-training ideally causes the model to develop general-purpose abilities and knowledge that can then be transferred to downstream tasks,

[3]The Annotated Transformer: http://nlp.seas.harvard.edu/2018/04/03/attention.html
The Illustrated Transformer: http://jalammar.github.io/illustrated-transformer/

transfer learning has become increasingly common in the NLP area. However, the rapid rate of progress and diversity of techniques in this thriving field can make it difficult to compare different algorithms, tease apart the effects of new contributions, and better understand the existing methods for transfer learning.

The need for more rigorous understanding is at the basis of the Text-to-Text Transfer Transformer (T5, Raffel et al. [2020]) model. This work proposes to treat every text processing task as a "text-to-text" problem, i.e. taking text as input and producing new text as output. This framework allows us to directly apply the same model, objective, training procedure, and decoding process to every task we consider, including machine translation, question answering, summarization, and text classification. Specifically, the model is trained with a maximum likelihood objective regardless of the task, and, to specify which task the model should perform, a task-specific textual prefix is added to the original input sequence before feeding it to the model.

As an example, to ask the model to translate the sentence "That is good." from English to German, the model would be fed the sequence "translate English to German: That is good." and would be trained to output "Das ist gut." For text classification tasks, instead, the model simply predicts a single word corresponding to the target label.

The T5 architecture is roughly equivalent to the original Transformer, except for minor changes, e.g. placing the layer normalization outside the residual path, and using a different position embedding scheme. The pre-trained T5 model is available in five different sizes:

- `t5-small`, with 6×10^7 parameters;
- `t5-base`, with 2.2×10^8 parameters;
- `t5-large`, with 7.7×10^8 parameters;
- `t5-3b`, with 3×10^9 parameters;
- `t5-11b`, with 1.1×10^10 parameters.

Clearly, larger models provide better results but note that they also require higher computational power and the training phase can be impractical even when recurring to cloud computing systems. In our system, we employ a fine-tuned `t5-large` model for the pure KG-to-text generation step, and a fine-tuned `t5-base` model for the last step, i.e., combining the KGs' verbalizations and the relevant textual information into one cohesive text

2.3 Evaluation of Text Generation

The recent advances in deep learning have yielded tremendous improvements in many Natural Language Generation tasks. This, in turn, highlights the important role assumed by the evaluation of these complex models.

Assessing the quality of NLG model output is challenging mainly because the majority of NLG tasks are open, in the sense that the target for a given input might not be unique.

For instance, a dialog system can generate multiple plausible responses for the same user input or a document can be summarized in different ways. Therefore, human evaluation remains the gold standard for almost all NLG tasks. This procedure, however, is expensive, and researchers often resort to automatic metrics for quantifying their progress and for performing automatic system optimization.

The most common human evaluation method is often referred as *intrinsic evaluation* and consists in asking people to evaluate the quality of the generated text, either overall or along some specific dimension (e.g., fluency, coherence, correctness, etc.). This is typically done by generating several samples of text from a model and asking human evaluators to score their quality. The simplest way to get this type of evaluation is to let them judge the quality of each text example individually. They could be asked to vote whether the text is good or bad, or to make more fine-grained decisions by marking the quality along a sliding scale. Moreover, to compare a model's output against baselines, model variants, or human generated text, intrinsic evaluations can also be performed by letting people choose which of two generated texts they prefer, or more generally, rank a set of generated texts. Nevertheless, as previously stated, human evaluation requires the employment of possibly expensive resources and is usually time-consuming, both when designing and running it, and, more importantly, the results are not always repeatable.

In order to cope with these issues, researchers often employ automatic evaluation metrics as an alternative in both developing new models and comparing them against baselines and state-of-the-art approaches. In the following paragraphs we report four examples of popular metrics that we will employ throughout the evaluation of our work in addition to the human evaluation.

2.3.1 BLEU

The Bilingual Evaluation Understudy (BLEU, Papineni et al. [2002]) is one of the first metrics used to measure the similarity between two sentences. Originally proposed for machine translation, it compares a candidate translation of text to one or more reference translations by measuring the overlap between n-grams[4].

BLEU is a weighted geometric mean of n-gram precision scores multiplied by a brevity penalty. Its simplest form is computed as follows:

$$\text{BLEU} = \min(1, e^{1-\frac{r}{c}}) \left(\prod_{i=1}^{4} \text{precision}_i \right)^{\frac{1}{4}}$$

where r and c are the counts of words in the reference and candidate texts respectively, while precision_i is the ratio between the count of the number of candidate's n-grams which occur in any reference divided by the total number of n-grams in the candidate text, for every n-gram of length $i = 1, \ldots, 4$.

BLEU is fast and cheap to compute, and enables a benchmark comparison with other models on the same task. Nevertheless, BLEU has been shown to correlate poorly with human judgments on tasks other than machine translation, where contextual understanding

[4] An n-gram is a contiguous sequence of n tokens from a given sample of text or speech.

and reasoning might be the key (e.g., story generation or long-form question answering), since it considers neither semantic meaning nor sentence structure.

2.3.2 ROUGE

The Recall-Oriented Understudy for Gisting Evaluation (ROUGE, Lin [2004]) is a set of metrics for evaluating automatic summarization of long texts consisting of multiple sentences or paragraphs. Although mainly designed for evaluating single- or multi-document summarization, it has also been used for evaluating short text generation, in the context of machine translation, image captioning, and question generation.

ROUGE includes a large number of distinct variants, including eight different n-gram counting methods to measure n-gram overlap between the generated and the ground-truth texts:

- ROUGE-{1/2/3/4} measures the overlap of unigrams/bigrams/trigrams/four-grams between the reference and hypothesis text;
- ROUGE-L measures the longest matching sequence of words using longest common sub-sequence (LCS);
- ROUGE-S is a less common variant that measures skip-bigram[5]-based co-occurrence statistics;
- ROUGE-SU is a less common variant that measures skip-bigram and unigram-based co-occurrence statistics.

Therefore, similarly to BLEU, ROUGE is based on measuring tokens' overlap, but the latter focuses on recall rather than precision.

2.3.3 BLEURT

BLEU and ROUGE fall into the category of *untrained automatic evaluation metrics* and share the drawback of correlating poorly with human evaluators on NLG tasks that permit significant diversity and allow multiple plausible outputs for a given input (e.g., a social chatbot). One solution to this problem is to train models on human judgment data to mimic human judges in order to measure many quality metrics of output, such as factual correctness, naturalness, fluency, coherence, etc.

A recent example is provided by BLEURT (Sellam et al. [2020]), a BERT-based machine-learned evaluation metric. It aims at capturing non-trivial semantic similarities between sentences and it is then suitable to properly evaluate various NLG systems. The evaluation model is trained as follows: A checkpoint from BERT is fine-tuned on synthetically generated sentence pairs using automatic evaluation scores, and then further fine-tuned on system-generated outputs and human-written references using human

[5] A skip-gram is a type of n-gram in which tokens (e.g., words) do not need to be consecutive but in order in the sentence, where there can be gaps between the tokens that are skipped over.

ratings and automatic metrics as labels. The fine-tuning of BLEURT on synthetic pairs is an important step because it improves the robustness to quality drifts of generation systems.

2.3.4 SLOR

A further model-dependant metric is the syntactic log-odds ratio (SLOR, Kann et al. [2018]), which is proposed as a score for referenceless fluency[6] evaluation of NLG output at the sentence level.

Fluency evaluation of NLG systems constitutes a hard challenge because it is not guaranteed that a fluent and readable output will match any of the given references. This results in difficulties for current reference-based evaluation, especially of fluency, causing word-overlap metrics like BLEU and ROUGE to correlate only weakly with human judgments.

SLOR assigns to a sentence S a score which consists of its log-probability under a given language model (LM), normalized by unigram log-probability and length:

$$\mathrm{SLOR}(S) = \frac{1}{|S|}(\log(p_M(S)) - \log(p_u(S)))$$

where $p_M(S)$ is the probability assigned to the sentence under the LM, which can be expressed in the following product using Bayes rule:

$$p_M(S) = p(\langle t_1, t_2, \ldots, t_{|S|}\rangle) = p(t_1)\prod_{i=2}^{|S|} p(t_i|t_1, \ldots, t_{i-1})$$

and $p_u(S)$ is the unigram probability of the sentence S computed as follows:

$$p_u(S) = \prod_{t\in S} p(t)$$

The intuition behind subtracting unigram log-probabilities is that a token which is rare on its own (in contrast to being rare at a given position in the sentence) should not bring down the sentence's rating. The normalization by sentence length is necessary in order to not prefer shorter sentences over equally fluent longer ones. Finally, note that the sentence log-probability normalized by sentence length corresponds to the negative cross-entropy of that sentence according to the language model employed during the evaluation.

[6] the property of a sentence to be perceived as natural by a human addressee. Alternative names include *naturalness*, *grammaticality*, or *readibility*.

Chapter 3

System description

The multiple issues stated in the introductory chapter, in particular the lack of a large customized dataset and the need to combine information from two different sources lead to the exclusion of the idea of using a single model to accomplish our task. Therefore, we decompose the latter into three necessary subtasks and propose a system based on the following three steps:

1. pure RDF-to-Text generation through a suitably fine-tuned T5 model;
2. content selection from the context through MARGE, a BERT-based ROUGE regression model;
3. combination of the two intermediate outputs through a T5-based fusion block.

The pipeline is shown through a schematic representation in Figure 3.1.

3.1 RDF-to-Text generation

In order to implement the pure RDF-to-Text generation phase, we follow the approach proposed in Li et al. [2020], because the obtained results ranked among the best in English KG-to-Text generation task of the WebNLG-2020 challenge (Castro Ferreira et al. [2020]),

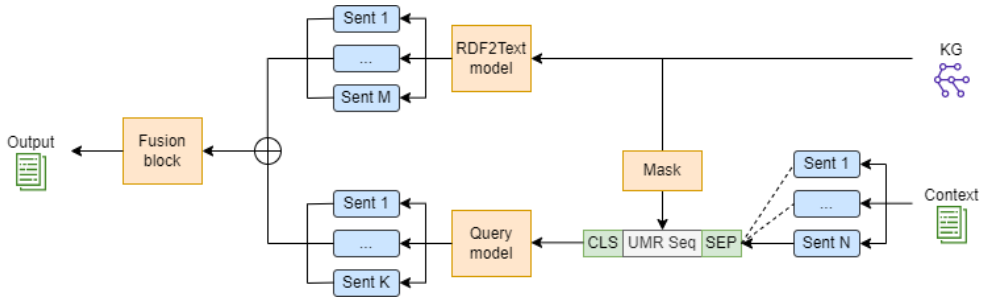

Figure 3.1: Overview of the proposed approach.

RDF:	`Aarhus_Airport	cityServed	\"Aarhus, Denmark\"`
MR:	`__subject__ Aarhus Airport __predicate__ cityServed __object__ Aarhus, Denmark`
Lex:	`The Aarhus is the airport of Aarhus, Denmark.`

Table 3.1: The format of original and preprocessed data. RDF is the original triple from DBPedia. MR is our preprocessed meaning representation. Lex is the reference realisation of the RDF triple (original example from Li et al. [2020]).

showing optimal results on the test set and nearly perfect performance on the validation set of the WebNLG-2020 English dataset (which is described in detail in Section 4.1). Nevertheless, note that any other off-the-shelf model could have been used instead.

The approach consists in fine-tuning large pretrained models to convert RDF triples into natural language. Regarding the English task of the WebNLG 2020 challenge, a T5 model is employed, while mBART (Liu et al. [2020]) is used for the Russian part.

Since we work with a sequence-to-sequence model, each triple set is linearized into a sequence with three delimiters, which are `__subject__`, `__predicate__`, `__object__`. Additionally, the underscores (`_`), as well as quotes (`\"`) surrounding the subject and the object are removed in order to reduce noise in the representations.

Considering that the T5 model tokenizes the sentences into subwords and the triple delimiters (e.g. `__predicate__`) are supposed to be indivisible, the three special delimiters are added to the vocabulary of the pretrained model. The embeddings of these three special delimiters are randomly initialized. After extending the vocabularies, the total parameters to be fine-tuned are 737,643,008.

The T5 model is then fine-tuned using cross entropy loss without label smoothing. The learning rate is constantly 2×10^{-5} and the batch size is equal to eight samples. The optimizer is Adam[1] (Kingma and Ba [2017]), with $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 1 \times 10^{-8}$, and the weight decay is 0. The best checkpoint is selected by validation with patience of ten training epochs. With this setting, the best checkpoint[2] is at the end of the 7^{th} epoch.

For further details, we recommend the original paper.

3.2 Content selection with MARGE

Regarding the extraction of relevant information from the textual context, we follow a strategy almost equivalent to the one presented in Xu and Lapata [2021]. This paper deals with query-focused summarization (QFS) in a framework where training data in the form of queries, documents and summaries is not readily available. The QFS task was first introduced in DUC 2005 (Dang [2005]), it provides a set of queries paired with relevant

[1] Adam optimization is a stochastic gradient descent method that is based on adaptive estimation of first-order and second-order moments. It is currently recommended as the default algorithm to use, and often works slightly better than other methods.

[2] A model checkpoint capture the exact value of all parameters (the so-called *weights*) used by such model. These weights can be used to make predictions as is, or used as the basis for ongoing training.

document collections and the expected output consists of a short summary answering the query according to data in the documents.

The authors of the paper propose a weakly supervised system for abstractive QFS where no query-related resources are required. In fact, they decompose the task into two phases:

1. *query modeling*, i.e. finding supportive evidence within a set of documents for a query;
2. *conditional language modeling*, i.e. generating an abstractive summary based on found evidence.

Our interest is focused mostly on the former, which is treated with MARGE, a Masked ROUGE regression framework for evidence estimation and ranking.

The query model is trained with distant supervision derived from *generic* summarization data which is easier to obtain (e.g., from online sources) compared to QA datasets which must be annotated from scratch (e.g., for different types of questions and domains). The crucial hypothesis behind this approach lies in the fact that the summaries themselves could constitute a response to *latent* queries, although queries are not directly verbalized in generic summarization. Then, in order to derive queries from the summaries, both queries and summaries are rendered in a *Unified Masked Representation* (UMR)[3], which enables summaries to serve as proxy queries for model training. Given the further assumption to find the answer to these queries in sentences from the document collection, we can assume that a certain sentence contains an answer if it has a high ROUGE score against the reference summary. Therefore, ROUGE is used as a distant supervision signal to train a model that takes a query and document sentence as input and estimates their relevance. At inference time, the actual queries are also rendered in UMR and the trained model ranks all sentences in the document collection. Specifically, the UMR query and a candidate sequence are concatenated to the sequence "`[CLS]` $\mathcal{U}$ `[SEP]` $\mathcal{C}$ `[SEP]`", where $\mathcal{U}$ is a sequence of tokens within a UMR query and $\mathcal{C}$ is a sequence of tokens in a document sentence. The `[CLS]` vector serves as input to a single layer neural network which estimates whether the sentence contains sufficient evidence to answer the query. The loss is computed via mean-square error and the encoding parameters in BERT are updated via standard backpropagation:

$$\mathcal{L}(\theta) = \frac{1}{|\mathcal{D}|} \sum_{(S,C)\sim\mathcal{D}} \Big[(y - \hat{y}(S, C; \theta))^2 \Big]$$

where (S, C) is a summary-sentence pair sampled from the documents' collection $\mathcal{D}$ and y is the training signal. Note that the training signal y is defined as the F1 interpolation of ROUGE-2 and ROUGE-1:

$$y = R_2(S, C) + \lambda R_1(S, C)$$

[3]Masking is a useful technique in NLP, especially in an unsupervised framework, consisting of *hiding* some words in a sentence behind a special token, i.e. the mask.

RDF:	`Aarhus_Airport	cityServed	\"Aarhus, Denmark\"`
UMR:	`[SUBQUERY] Aarhus Airport [MASK] city served [MASK] Aarhus, Denmark`

Table 3.2: The format of original and preprocessed data. RDF is the original triple from DBPedia. UMR is our Unified Masked Representation.

with $\lambda \geq 0$.

In our particular framework, instead, the query is composed by a set of semantic triples and the text for evidence ranking corresponds to the context associated to that specific set.

To cope with the different level of linguistic abstraction of the RDF triples with respect to the textual queries treated in the original paper we propose a masking process which aims at rendering a set of triples in such a way to be more similar to a masked sentence: in particular, we introduce a `[SUBQUERY]` token at the beginning of each triple and a `[MASK]` token between subject and predicate, as well as between predicate and object. Moreover, similarly to the preprocessing step in the pure RDF-to-text generation phase, we remove the underscores (_) and quotes (\") surrounding subjects and objects. Finally, we split the predicate according to the present upper cases, as shown in example 3.2. This masking strategy allows the pipeline to be flexible with respect to the selected content, allowing to give more relevance to specific parts of the RDF query by masking other parts. Nevertheless we decide not to mask any component in order to be sure to select only the strictly relevant information with respect to the triples involved.

The setting for the training phase is the following: the learning rate is constantly 3×10^{-5}, the batch size is equal to 16 samples, the number of training epochs is two and finally $\lambda = 0.15$. The optimizer is Adam, with $\epsilon = 1 \times 10^{-8}$ and weight decay equal to 0.

At inference time, given a set of RDF triples and a textual context, the model evaluates the score for each query-sentence pair in the document. The sentences are then ranked in descending order according to their scores. Finally, the *top-K* sentences are selected and concatenated to compose the final output of the content selection phase. We fix $K = 10$ but note that this parameter can be adjusted depending on the specific requirements of the application.

An alternative approach could be to replace the RDF triples with their verbalizations, i.e. to use the output of the RDF-to-text generation block as the queries of the content selection step, thus to solve the issues related to the different levels of linguistic abstraction between RDF triples and textual documents. We exclude this strategy in order to avoid the propagation of errors, e.g. hallucinations due to the T5 model severely affecting the evidence ranking, but we are confident that this may constitute a further line of future research.

3.3 Fusion block

To conclude the pipeline, we need to organize the information contained in the two intermediate textual outputs into one single cohesive and fluent text. Note that a simple concatenation might contain several repetitions as well as disjoint consecutive sentences.

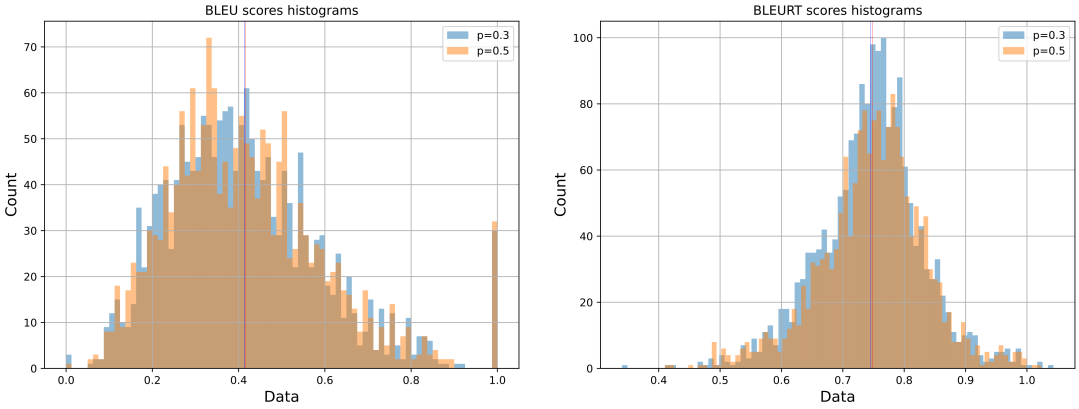

Figure 3.2: Histograms of BLEU and BLEURT scores (left and right, respectively) over the two test sets.

Hence, in order to rearrange the information in a more appealing way, we follow the approach presented in Section 3.1, i.e. fine-tuning a large pretrained sequence-to-sequence model on a custom dataset thought for this specific task.

As we stated above, the task mainly consists of removing repetitions and rearranging the sentences in the best possible order, similarly to a jigsaw puzzle problem. Therefore we fine-tune a T5 model on a dataset, which is described in detail in 4.3, where the input text is essentially a noisy version of the target, containing randomly shuffled sentences and random repetitions of the same information.

As described in 4.3, we build two versions of the dataset, experimenting with two different average numbers of repetitions. By consequence, we perform one separate fine-tuning for each dataset. We employ the same setting for both cases and the models are trained for ten epochs. The learning rate is constantly equal to 3×10^{-4} with batches composed of four samples. The optimizer is Adam, with $\epsilon = 1 \times 10^{-8}$ and the weight decay is 0. The best checkpoint is at the end of the 4^{th} epoch for both models.

The two models yield roughly equivalent performances on the test sets in terms of BLEU and BLEURT scores, as showed in Figure 3.2. Despite their similar behaviour, we empirically observe that the whole pipeline provides better results when equipped with the model fine-tuned on the dataset with the higher average number of repetitions. We think that this might be due to the fact that this model has learned to deal with sequences whose average length matches with the one of the real data.

Chapter 4

Dataset

As we mentioned in the introductory chapter, given the novelty of our task, we could not find a suitable dataset. Therefore, to gather custom data is a necessary preliminary step for our work. Ideally, each instance should contain at least one cluster of textual documents representing the context, a set of RDF triples, and at least one target text, but, unfortunately, only the first two components are readily available.

The proposed system copes with the lack of the actual target, but it requires three datasets, one for each of the three steps involved in the pipeline. Regarding the first, for the RDF-to-text generation step, we use an already existing dataset: the WebNLG corpus[1], which comprises of sets of triplets and their corresponding facts in form of natural language text. For the relevant information extraction from the context, we create a custom dataset by performing a connection between a collection of Wikipedia articles and the WebNLG corpus. Finally, for the fusion block, we propose a further dataset, based on the descriptive texts available in the WebNLG corpus, where the input text is essentially a noisy version of the target, containing randomly shuffled sentences and random facts' repetitions.

In the following sections, we describe in detail the structure of the three datasets presented above. We start describing the WebNLG corpus (Section 4.1). Then, in Section 4.2, we describe the process of building our first custom dataset and we analyze its features. Finally, in Section 4.3, we describe the second custom dataset.

4.1 WebNLG dataset

The WebNLG dataset (Gardent et al. [2017a]) has been developed for the homonymous challenge, which involves mapping data to text. The training data consists of data/text pairs where the data is a set of triples extracted from DBpedia and the text is a verbalization of these triples. Therefore, the WebNLG challenge involves specific NLG subtasks,

[1] https://gitlab.com/shimorina/webnlg-dataset

such as sentence segmentation, i.e. how to chunk the input data into sentences, lexicalization of the DBpedia properties, aggregation in order to avoid repetitions, and surface realization, i.e. how to build a syntactically correct and natural sounding text.

The dataset associated with the first edition of the challenge, i.e. WebNLG 2017, consists of 21855 data/text pairs with a total of 8372 distinct data input. The input originally described entities belonging to nine distinct DBpedia categories: *Astronaut, University, Monument, Building, ComicsCharacter, Food, Airport, SportsTeam* and *WrittenWork*. Then, this set was further expanded to include *CelestialBody, MeanOfTransportation, City, Athlete, Politician, Artist*.

Our work employs the dataset associated with the second edition of the challenge, WebNLG 2020 (Castro Ferreira et al. [2020]), which comprised two main tasks:

1. RDF-to-text generation, similarly to WebNLG 2017, but into two languages (English and Russian);
2. Text-to-RDF semantic parsing: converting a text into the corresponding set of RDF triples.

The English WebNLG 2020 dataset for training comprises data/text pairs for 16 distinct DBpedia categories:

- ten seen categories used in 2017: *Airport, Astronaut, Building, City, ComicsCharacter, Food, Monument, SportsTeam, University*, and *WrittenWork*;
- five unseen categories of 2017, which are now part of the seen data: *Athlete, Artist, CelestialBody, MeanOfTransportation, Politician*.
- one new category: *Company*.

Moreover, approximately 5600 texts were cleaned from misspellings and missing triple verbalizations were added to some texts.

In Table 4.1 we report some basic statistics regarding the whole dataset, while below we show an example in the original XML format.

```
<entry category="Food" eid="Id65" shape="(X (X) (X))" shape_type="sibling" size="2">
    <originaltripleset>
        <otriple>Arròs_negre | country | Spain</otriple>
        <otriple>Arròs_negre | ingredient | White_rice</otriple>
    </originaltripleset>
    <modifiedtripleset>
        <mtriple>Arròs_negre | country | Spain</mtriple>
        <mtriple>Arròs_negre | ingredient | White_rice</mtriple>
    </modifiedtripleset>
    <lex comment="good" lid="1">White rice is an ingredient of Arros negre which is
    a traditional dish from Spain.</lex>
    <lex comment="good" lid="2">White rice is used in Arros negre which is from
    Spain.</lex>
    <lex comment="good" lid="3">Arros negre contains white rice as an ingredient and
```

	Train	Dev
Entries	13211	1667
Lexicalizations	35426	4464
Distinct properties	372	290

Table 4.1: Basic statistics related to the WebNLG 2020 English dataset.

```
    it comes from Spain.</lex>
</entry>
```

4.2 Custom dataset #1

Let us first describe the source from where we extract data from Wikipedia. We employ the Wikipedia articles' collection[2] publicly available on the `datasets` library from Hugging Face, a large open-source community focused on the NLP domain that quickly became an enticing hub for pre-trained deep learning models. The Wikipedia dataset contains cleaned articles of different languages. The datasets are built from the Wikipedia dump with one split per language. Clearly, we focus our interest on the English split, `20200501.en`, which is composed of approximately $6 \cdot 10^6$ articles. Each example contains the content of one full Wikipedia article preprocessed in order to strip markdown and unwanted sections (references, etc.). The articles have been parsed using the `mwparserfromhell` tool.

A short example is reported below.

```
{'title': "Waleed Al Sayegh",
'text': "Waleed Ibrahim Al-Sayegh is the director-general of the Central
Finance Department of Sharjah (since at least 2014), chairman of Sharjah
Holding (a real-estate developer in Sharjah) and CEO of Sharjah Asset
Management ; he is on the board of Air Arabia the Sharjah-based low-cost
airline, and was part of the board imposed during the government of
Sharjah's takeover of Invest Bank.

References

Category:Chief executive officers
Category:Living people
Category:Year of birth missing (living people)"
}
```

In order to create a connection between the Wikipedia and the WebNLG datasets, we compare the title of each Wikipedia article with the list of subjects inside each triplet set: if the title of an article coincides with the subject of one of the triples in a specific

[2] https://huggingface.co/datasets/wikipedia

WebNLG record, then such article is selected to be part of the context associated to that WebNLG record. The selected articles are further preprocessed to eliminate the last part, mainly containing the categories to which the subjects belong.

The number of entries is reported in Table 4.2. Comparing the values with the ones inside Table 4.1, we could see that we have not been able to enstablish connections for 669 records in the training set and for 83 in the validation set.

We conclude this section by highlighting the fact that the test set of this dataset is employed during the evaluation phase of our system (Chapter 5).

4.3 Custom dataset #2

The second custom dataset is necessary for fine-tuning the language model inside the fusion block. Since this last step mainly involves repetitions' removal and rearranging the sentences in order to improve coherence and fluency, the input text needs to be essentially a noisy version of the target, where the sentences are randomly shuffled and the same fact might appear more than once.

To build this dataset, we exploit the fact that each entry inside the WebNLG corpus usually contains multiple verbalizations (between one and three, with an average of 2.7 verbalizations per entry). In particular, we apply the following criteria:

- if an entry contains a single verbalization, we discard it;
- if an entry contains two verbalizations, we randomly select one as the input text and the other as the target;
- if an entry contains three verbalizations, we randomly select one as the target text, while the other two are combined to form the input text, in such a way to let the information appear in a random order and contain repetitions.

By discarding the entries containing one verbalization, the size of this dataset is slightly smaller than the WebNLG corpus, having 12326. 1558 and 1753 entries for the training, validation and test sets, respectively.

Clearly, this dataset heavily depends on the choice of the average amount of repetitions: in fact, our algorithm selects each *additional* sentence according to the realization of a Bernoulli distribution of parameter p. Therefore, we build two datasets, for $p = 0.3$ and $p = 0.5$, in order to perform two different fine-tunings and see which one leads to better results.

Let us conclude this section by reporting the same example in the case where $p = 0.3$ and $p = 0.5$, respectively:

	Train	**Dev**	**Test**
Entries	12542	1584	1705

Table 4.2: Number of entries for each split of the first custom dataset.

{'inputs': "Elliot See, born in Dallas, which is in Collin County, Texas, graduated from the University of Texas in Austin. The University of Texas, Austin, in Dallas, Collin County, is an affiliate of the University of Texas system. Its sporting teams compete in the Big 12 Conference and their mascot is Hook'em. A notable alumni of the university is Dallas-born Elliot See.",
'targets': "Elliot See was born in Dallas, Collin County, Texas. He was a student at the University of Texas at Austin, which is affiliated to the University Of Texas system. The University of Texas at Austin is competing in the Big 12 Conference and their mascot is called Hook'em."}

{'inputs': "The Austin University with its mascot Hook'em is affiliated to the Texas University system and is competing in the Big 12 Conference. The University of Texas, Austin, in Dallas, Collin County, is an affiliate of the University of Texas system. Its sporting teams compete in the Big 12 Conference and their mascot is Hook'em. A notable alumni of the university is Dallas-born Elliot See.",
'targets': "Elliot See was born in Dallas, Collin County, Texas. He was a student at the University of Texas at Austin, which is affiliated to the University Of Texas system. The University of Texas at Austin is competing in the Big 12 Conference and their mascot is called Hook'em."}

As we can see from the examples above, the two datasets are very similar, but the expected length of the input texts of the second dataset is slightly greater than the one in the first dataset.

Chapter 5

Results

This chapter is dedicated to present and analyze the results obtained by applying our system to the test set of our first custom dataset. We recall that our aim is to enrich the descriptive text of an input knowledge graph with relevant information extrapolated from a related textual context.

We remark the difficulty in assessing the quality of a generated textual output in a *referenceless* framework. For this reason, we first conduct an empirical analysis in Section 5.1 by reporting and describing both positive and negative examples. Afterwards, in Section 5.2, we describe the results according to two methods, i.e. intrinsic human evaluation and automatic fluency evaluation.

5.1 Examples

In order to have a glimpse on the real behaviour of an NLG system, it is good practice to examine some output examples and try to empirically extrapolate their characteristics.

Therefore, we now present a list of texts, starting from the positive examples and concluding with the failed ones. Note that the reference text for the pure KG-to-Text generation is showed as a baseline along with the output of our system for comparative purposes: ideally, the latter should be an enriched version of the former.

First example

Baseline:

> The Acharya Institute of Technology can be found in India in the state of Karnataka, is affiliated with Visvesvaraya Technological University, and the campus address is: Soldevanahalli, Acharya Dr. Sarvapalli Radhakrishnan Road, Hessarghatta Main Road, Bangalore - 560090.

Example:

> The Acharya Institute of Technology is located in the state of Karnataka in India. It is affiliated with the Visvesvaraya Technological University and has a

> Campus in the academic discipline of Mechanical engineering and is affiliated with the All India Council for Technical Education (AICTE). The institute was established in 2000 and is located at Soldevanahalli, Acharya Dr. Sarvapalli Radhakrishnan Road, Hessarghatta Main Road, Bangalore – 560090.

Description: As we can see, the output is cohesive and fluent, as well as grammatically correct. Moreover, not only it contains the facts presented in the baseline, but also enriches them with closely related information.

Second example

Baseline:

> Alan Shepard is an American who was born on November 18, 1932 in New Hampshire. He graduated from NWC with an M.A. in 1957. Shepard retired on August 1, 1974 and died in California.

Example:

> Rear Admiral Alan Shepard Jr was born on Nov 18th 1923 in Derry, New Hampshire and died on 1st July 1974 in California. He graduated from NWC with an MA in 1957 and served as a test pilot. He was a crew member on the Douglas F5D Skylancer and was an American. He was a member of the U.S. Navy and became a Commander in Chief of the Aircraft Office from July 1, 1974 to July 1, 1974.

Description: As we can see, the text is well structured, fluent, and grammatically correct. Despite the overall good quality of the text, there are some evident date-related mismatches between the baseline and our example.

Third example

Baseline:

> Adam West was born in the year 1928.

Example:

> Adam West (1980) played Batman in the 1960s ABC series of the same name. He later became a performer in the stefy show, "Chelsea", played for Quahog, Rhode Island. In addition, he is an American. In the film, Family Guy, he was born on the 19th September, 1928 and passed away on the 9th of June, 2017.

Description: As we can see, the output results fluent and grammatically correct, but lacks of coherence and faithfulness. Despite illustrating some further information on the biography of the subject Adam West, his year of birth is not mentioned clearly and correctly. Moreover, some of the stated facts are visibly inexact.

Fourth example

Baseline:

> The song Mermaid by Train is written in the reggae music genre and was followed by the band's cover of the John Lennon song Imagine. The original version of Imagine is in the pop music genre and was followed up by Lennon's hit Happy Xmas (War is Over).

Example:

> John Lennon is an English musician. He was originally a solo performer of the song 'Imagine' which was published in 1998. During the recording of the song, "Body Counts" he performed it with the guitar and was accompanied by Yoko Ono. The song was published in October 2010 on the B-side of the John Lennon Peace Monument (located in Chavasse Park, Liverpool).

Description: The text is grammatically correct and rather cohesive, but lacks of faithfulness and coherence. Another negative aspect of this output is the drastic shift of focus towards a relatively marginal subject (i.e., John Lennon) over the actual subjects presented in the baseline (i.e., the songs "Mermaid" and "Imagine").

Fifth example

Baseline:

> The Pontiac Rageous assembly line is in Detroit, which encompasses an area of 370.03 square kilometers.

Example:

> Detroit is part of the city of Detroit which is in Detroit and Southeast Michigan. The city is governed by the Home Rule Charter and boasts a population of 110,000. It is home to the largest city in the city of Detroit, where Detroit is located. Detroit has a total area of 370.03 square kilometres, and is home to the most violent tornadoes. Detroit in 2001, is led by a private firm called The Detroit City Code.

Description: This is an example of seriously negative result, where all dimensions of a possible evaluation would obtain poor scores.

5.2 Evaluation

Clearly, the qualitative analysis provided in the previous section is not informative enough and we certainly need a more reliable quantitative study. Therefore, we propose a two-fold approach:

1. referenceless automatic fluency evaluation through SLOR;

2. questionnaire-based intrinsic human evaluation.

We recall from Section 2.3.4 that SLOR assigns to a sentence a score which consists of its log-probability under a given language model normalized by unigram log-probability and length. Our SLOR implementation employs the pretrained XLNet (Yang et al. [2020]) language model mainly because its tokenizer easily provides unigram log-probabilities.

In order to understand the scores achieved by our system on the test set, we compare them with the SLOR values of the reference verbalizations of the corresponding entries in the WebNLG test set and the candidate verbalizations obtained by the model for pure RDF-to-text generation. The average and standard deviation of the obtained SLOR scores are reported in Table 5.1: as we can see, the average values are very close to each other (in fact, the three values lie inside an interval of length 0.46). This lead to the conclusion that the three sets of texts share approximately the same level of language fluency.

Let us now discuss the questionnaire-based human evaluation. This procedure involves the rating of the quality of a small sample of texts according to four dimensions:

- *Coherence*, whether the text makes sense and is coherently organised (three options: Yes/Somewhat/No);
- *Grammaticality*, whether the text is free of grammatical mistakes (three options: Yes/Somewhat/No);
- *Faithfulness*, whether the text contains at least all the information from a reference text (two options: Yes/No);
- *Informativeness*, whether the text provides interesting information that enriches the reference text (three options: Yes/Somewhat/No).

Clearly, the coherence and the grammaticality only depends on the text under analysis, while the remaining criteria need a comparison with a reference (also called *seed text*). Therefore, each judge is asked to rate a sample composed of 20 texts resulting from our system along with the associated output for the pure KG-to-text task. The latter is necessary to set a sort of baseline and make reasonable comparisons. The seed texts for both the evaluations correspond to the reference verbalizations extracted from the WebNLG test set.

A total of 13 judges completed the questionnaire. In the following lists we report the summary of the received feedback.

	AVG_{SLOR}	STD_{SLOR}
ref. verbalization	2.79	0.88
KG2Text output	2.83	0.90
our output	3.25	0.49

Table 5.1: Average and standard deviation of SLOR scores on the reference verbalizations of the corresponding entries in the WebNLG test set, the candidate verbalizations obtained by the KG-to-text model and the test output of our system.

Baseline:

%	Yes	No	Somewhat
Coherence	80.0	6.15	13.85
Grammaticality	82.7	8.08	9.23
Faithfulness	76.15	23.85	-
Informativeness	12.3	78.85	8.85

Our results:

%	Yes	No	Somewhat
Coherence	45.38	28.85	25.77
Grammaticality	65.0	18.46	16.54
Faithfulness	35.38	64.62	-
Informativeness	68.08	7.3	24.62

As we can see, our system is outperformed by the pure RDF-to-text generator, except for the informativeness' criterion, which received approximately 7% of negative votes in the former case and 79% in the latter. This is an expected behaviour given the higher complexity of our task, which also involves information enrichment. Nevertheless, both coherence and grammaticality achieves good ratings (approximately 45% and 65% of positive votes, respectively). The most serious issue is represented by the lack of faithfulness: in fact, almost 65% of our samples miss some information from the seed knowledge graphs. Overall, the generated texts are usually sensible and well-structured, with a good level of grammatical correctness, but their content might often be subject to omissions and hallucinations. These issues occur often in the deep learning-based NLG framework and are typically difficult to control (Ji et al. [2022]), leading to the generation of possibly unintended and non-reliable text.

Since human evaluation always includes some degree of subjectivity, judges might disagree in their ratings, and the level of disagreement can be a useful measure to researchers. In fact, high levels of inter-evaluator agreement generally mean that the task is well-defined and the differences in the generated text are consistently noticeable to evaluators, while low agreement can indicate a poorly defined task or that there are not reliable differences in the generated text. Roughly, human evaluations are broadly thought to be more valuable the higher the inter-annotator agreement. Nevertheless, the reality is slightly more complex, because natural language brings in itself a variability which cannot be reduced, except by weakening its expressive power. For instance, the perception of sentence idiomaticity can change from person to person because of styles, educational and regional difference (e.g., differences between British and American English for English sentences).

In order to study the inter-annotator agreement in our specific case, we propose a weighted version of the standard percent agreement. This metric measures the level of agreement between multiple annotators by looking at how often pairs of evaluators agree. Let us denote the set of texts to evaluate as X and the set of available scores as S. Then,

we can define, for each text $x_i \in X$, the agreement in the scores, a_i, as follows:

$$a_i = \frac{\sum_{s \in S} \#\text{ of evaluator pairs who score } x_i \text{ as } s}{\text{total } \#\text{ of evaluator pairs}}.$$

Then, the overall percent agreement for the task is:

$$P_a = \frac{\sum_{i=1}^{|X|} a_i}{|X|}.$$

However, this formula treats all evaluator disagreements as equally bad, but it might be argued that, in our case, the disagreements involving the "Somewhat" option should impact less than Yes-No pairs. To keep track of this aspect, we propose to add a further term, which could be referred as *percent partial agreement*, P_{pa}, multiplied by a relevance factor $\lambda \in [0,1]$:

$$P_{\text{mod}} = P_a + \lambda P_{pa}.$$

where

$$P_{pa} = \frac{\sum_{i=1}^{|X|} b_i}{|X|}$$

with

$$b_i = \frac{\#\text{ of evaluator pairs where one judge scores } x_i \text{ as } \textit{Somewhat}}{\text{total } \#\text{ of evaluator pairs}}.$$

After setting $\lambda = 0.5$, we obtain the inter-evaluator agreement scores reported in Table 5.2: as we can see, we obtain a good overall agreement across all the four dimensions, both for the baseline and our results, although the ones related to the former are slightly higher. Clearly, this is due to the fact that the reference and baseline texts are derived from the same task, so that the comparison is more straightforward.

Despite this expected behavior, it is worth noticing that the difference between the agreement score related to the informativeness of both the baseline and our results is negligible. Moreover, the high level of agreement on the informativeness dimension implies that our task is well-defined and the differences in the generated text are consistently noticeable to evaluate.

P_{mod}	Baseline	Our results
Coherence	0.80	0.67
Grammaticality	0.80	0.65
Faithfulness	0.74	0.65
Informativeness	0.77	0.74

Table 5.2: Values of P_{mod} according to the four dimensions and the two datasets under analysis.

Chapter 6

Conclusions

6.1 Discussion

In this thesis we discussed the novel hybrid knowledge graph-to-text and text-to-text neural generation problem. This task is particularly relevant in a framework where the available ontologies are poor but can be enriched with contextual information available in textual format.

In order to address this task, we proposed a system composed of three blocks, each of them aiming at solving one of the following sub-tasks:

1. pure KG-to-Text generation through a suitably fine-tuned T5 model;
2. content selection from the context through MARGE, a BERT-based ROUGE regression model;
3. combination of the two intermediate outputs through a T5-based fusion block.

Each model has been trained on a different dataset. Regarding the KG-to-Text phase, the WebNLG-2020 English dataset has been employed. Then, for the remaining two steps, we derived two further custom datasets from the WebNLG corpus. In particular, to train MARGE, we created a connection between a collection of Wikipedia articles and the WebNLG corpus. Finally, to train the T5 model employed in the last step, we propose a further dataset, based on the descriptive texts available in the WebNLG corpus, where the input text is a noisy version of the target, containing randomly shuffled sentences and random repetitions.

The obtained results have been then analyzed through the means of intrinsic human evaluation and the SLOR metric for fluency. The generated texts overall reached good levels of grammatical correctness and informativeness, but there is room for improvement with regard to textual coherence and faithfulness.

6.2 Future works

We are confident that this work can open the path to several future research directions.

First of all, as pointed out throughout the discussion, a large-scale task-specific dataset is a necessary tool for further developing this research: in fact, it would allow to address the task through a broader variety of neural models and it would drastically simplify the evaluation procedure.

For instance, a reasonable task-specific dataset could be built by cleverly exploiting the WebNLG dataset. Since its examples contain up to seven RDF triples, one might randomly remove triples from each record according to a certain preserving ratio. Then, by associating a text (e.g., Wikipedia articles) that contains the removed information, we finally obtain records composed of a KG source, a textual context and a target descriptive text of the KG, which is also enriched with further information available in the context. Moreover, WebNLG records with a small number of triples (e.g., KGs composed of two triples at most) could be employed to derive negative examples, by not removing any triple and associating an uninformative context.

With this kind of resources, we believe it would be possible to expand the knowledge-grounded[1] NLG area, often restricted to conversation models (e.g., Ghazvininejad et al. [2018], Qin et al. [2019], Dinan et al. [2019]), to a broader content-to-text generation framework. This would eventually lead to improve our current system in such a way to increase the faithfulness and the coherence of the generated output.

[1]Grounding is broadly referred as any linking of text to data or non-textual modality. In contrast, Cognitive Science more formally defines "grounding" as the process of establishing what mutual information is required for successful communication between two interlocutors.

Bibliography

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. Abstract Meaning Representation for sembanking. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*, pages 178–186, Sofia, Bulgaria, August 2013. Association for Computational Linguistics. URL https://aclanthology.org/W13-2322.

Tim Berners-Lee, James Hendler, and Ora Lassila. The semantic web: A new form of web content that is meaningful to computers will unleash a revolution of new possibilities. *ScientificAmerican.com*, 05 2001.

Thiago Castro Ferreira, Claire Gardent, Nikolai Ilinykh, Chris van der Lee, Simon Mille, Diego Moussallem, and Anastasia Shimorina. The 2020 bilingual, bi-directional WebNLG+ shared task: Overview and evaluation results (WebNLG+ 2020). In *Proceedings of the 3rd International Workshop on Natural Language Generation from the Semantic Web (WebNLG+)*, pages 55–76, Dublin, Ireland (Virtual), 12 2020. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/2020.webnlg-1.7.

Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *CoRR*, abs/1406.1078, 2014. URL http://arxiv.org/abs/1406.1078.

Hoa Trang Dang. Overview of duc 2005. In *In Proceedings of the Document Understanding Conf. Wksp. 2005 (DUC 2005) at the Human Language Technology Conf./Conf. on Empirical Methods in Natural Language Processing (HLT/EMNLP*, 2005.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019.

Emily Dinan, Stephen Roller, Kurt Shuster, Angela Fan, Michael Auli, and Jason Weston. Wizard of wikipedia: Knowledge-powered conversational agents, 2019.

Lisa Ehrlinger and Wolfram Wöß. Towards a definition of knowledge graphs. 09 2016.

Hanning Gao, Lingfei Wu, Po Hu, and Fangli Xu. Rdf-to-text generation with graph-augmented structural neural encoders. In Christian Bessiere, editor, *Proceedings of*

the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20, pages 3030–3036. International Joint Conferences on Artificial Intelligence Organization, 7 2020. doi: 10.24963/ijcai.2020/419. URL https://doi.org/10.24963/ijcai.2020/419. Main track.

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. doi: 10.18653/v1/P17-1017. URL https://aclanthology.org/P17-1017.

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, Santiago de Compostela, Spain, September 2017b. Association for Computational Linguistics. doi: 10.18653/v1/W17-3518. URL https://aclanthology.org/W17-3518.

Marjan Ghazvininejad, Chris Brockett, Ming-Wei Chang, Bill Dolan, Jianfeng Gao, Wen tau Yih, and Michel Galley. A knowledge-grounded neural conversation model, 2018.

Alex Graves. Generating sequences with recurrent neural networks. *CoRR*, abs/1308.0850, 2013. URL http://arxiv.org/abs/1308.0850.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9:1735–80, 12 1997. doi: 10.1162/neco.1997.9.8.1735.

Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Yejin Bang, Andrea Madotto, and Pascale Fung. Survey of hallucination in natural language generation. *CoRR*, abs/2202.03629, 2022. URL https://arxiv.org/abs/2202.03629.

Katharina Kann, Sascha Rothe, and Katja Filippova. Sentence-level fluency evaluation: References help, but can be spared!, 2018.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.

Xintong Li, Aleksandre Maskharashvili, Symon Jory Stevens-Guille, and Michael White. Leveraging large pretrained models for WebNLG 2020. In *Proceedings of the 3rd International Workshop on Natural Language Generation from the Semantic Web (WebNLG+)*, pages 117–124, Dublin, Ireland (Virtual), 12 2020. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/2020.webnlg-1.12.

Chin-Yew Lin. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain, July 2004. Association for Computational Linguistics. URL https://aclanthology.org/W04-1013.

Yinhan Liu, Jiatao Gu, Naman Goyal, Xian Li, Sergey Edunov, Marjan Ghazvininejad, Mike Lewis, and Luke Zettlemoyer. Multilingual denoising pre-training for neural machine translation, 2020.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *ACL*, 2002.

Lianhui Qin, Michel Galley, Chris Brockett, Xiaodong Liu, Xiang Gao, Bill Dolan, Yejin Choi, and Jianfeng Gao. Conversing by reading: Contentful neural conversation with on-demand machine reading, 2019.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer, 2020.

Ehud Reiter and Robert Dale. *Building Natural Language Generation Systems*. Studies in Natural Language Processing. Cambridge University Press, 2000. doi: 10.1017/CBO9780511519857.

Thibault Sellam, Dipanjan Das, and Ankur P. Parikh. Bleurt: Learning robust metrics for text generation, 2020.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *CoRR*, abs/1706.03762, 2017. URL http://arxiv.org/abs/1706.03762.

Yumo Xu and Mirella Lapata. Generating query focused summaries from query-free resources, 2021.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. Xlnet: Generalized autoregressive pretraining for language understanding, 2020.