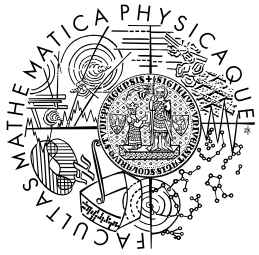

FACULTY
OF MATHEMATICS
AND PHYSICS
Charles University

MASTER THESIS

Nalin Kumar

Neural models for multilingual natural language understanding and generation

Institute of Formal and Applied Linguistics, Charles University

Supervisor of the master thesis: Mgr. Ondřej Dušek, Ph.D.
Study programme: study programme
Study branch: study branch

Prague 2024

I declare that I carried out this master thesis independently, and only with the cited sources, literature and other professional sources. It has not been used to obtain another or the same degree.

I understand that my work relates to the rights and obligations under the Act No. 121/2000 Sb., the Copyright Act, as amended, in particular the fact that the Charles University has the right to conclude a license agreement on the use of this work as a school work pursuant to Section 60 subsection 1 of the Copyright Act.

In ..... date ....

Author's signature

Dedication.

Title: Neural models for multilingual natural language understanding and generation

Author: Nalin Kumar

Department: Institute of Formal and Applied Linguistics, Charles University

Supervisor: Mgr. Ondřej Dušek, Ph.D., UFAL

Abstract: There has been significant progress in the field of natural language understanding and generation with the aid of pretrained language models (PLM). Notably, multi-task approaches have demonstrated superior performance when combined with PLM. The 2020 WebNLG shared task has served as a pivotal benchmark, providing a dataset comprising RDF triples and corresponding natural language descriptions. However, despite advancements, challenges persist, particularly in terms of accuracy in comprehension and generation. This thesis primarily investigates multi-task methodologies for language understanding, specifically RDF triple parsing and natural language generation. The study explores auxiliary training objectives, such as named entity recognition and denoising corrupt inputs, to regularize trained models. The research also delves into domain adaptability and knowledge transfer between RDF triple parsing and NLG tasks. Findings highlight the significance of data cleaning as a straightforward yet highly effective approach for generating accurate triples and natural text. Additionally, the thesis addresses low-resource scenarios and assesses the applicability of multilingual models to ensure the generalizability of the proposed methodologies beyond the limitations of the WebNLG benchmark. For extremely low-resource languages, where training data proved excessively noisy, cleaned training data is released as one of the solutions to this issue.

Keywords: Semantic parsing Natural language processing Natural language generation Natural language understanding

Contents

1. Introduction

In recent years, the fields of natural language understanding and generation have seen major advancements due to the development of pretrained language models (PLMs) both in terms of accuracy and adaptability. They are often applied with various other methods like multi-task/lingual approaches, improving their performance across various tasks and languages. Text generation models like BART, T5, and GPT-2 [Lewis et al., 2020, Raffel et al., 2020, Radford et al., 2019] have provided performance improvements on many natural language generation (NLG) tasks [Zhang et al., 2020b, Wei et al., 2021, Kale and Rastogi, 2020]. One of the most prevalent NLG tasks is data-to-text generation [Wiseman et al., 2017]. It involves converting structured data to fluent text in a given language. This task has found its application in various real-life domains, like sports journalism, voice assistants, etc. Another relevant and significant task is text-to-data — parsing fluent text to structured data. With PLMs getting popular, both tasks are often treated as sequence-to-sequence, working with a linearized representation of the data. To facilitate improvement in both tasks, the WebNLG 2017 challenge was introduced [Gardent et al., 2017a]. While the initial edition only focused on text generation in the English language, the following edition in 2020 Castro Ferreira et al. [2020] included both tasks along with Russian data. The latest edition in 2023 Cripwell et al. [2023] included texts in four low-resource languages — Irish, Maltese, Welsh, and Breton.

In this thesis, we explore the use of PLMs for these tasks in multilingual and monolingual settings. Specifically, for text-to-data generation, we experiment with several auxiliary tasks to improve the performance of text-to-data models. Additionally, we experiment with several methods in a multilingual setting. For the data-to-text generation task, we explore various training and decoding strategies for the data-to-text generation task to enhance multilingual text generation. Furthermore, we experimented with a parameter-efficient fine-tuning strategy in a simulated low-resource setting.

The main contributions of this thesis are:

- We propose various joint strategies to enhance data-to-text generation task.
- We release a multilingual data-to-text model capable of parsing English, Russian, Irish, Maltese and Welsh languages' texts.
- We propose and explore various training and decoding methods for improving data-to-text generation, both in English and low-resource languages.
- We study and compare the effectiveness of two different fine-tuning strategies in a low-resource setting.

In this thesis, we first discuss relevant concepts, methods, and tasks in Chapter 2. We then describe the used datasets in this thesis in Chapter 3. A detailed study on various approaches both in multilingual and monolingual settings is given in Chapter 4. Chapter 5 discusses various training, finetuning, and decoding strategies for text generation, again in both multilingual and monolingual settings. We finally conclude the thesis in Chapter 6.

2. Background

In this chapter, we explore the fundamental concepts in natural language processing, mostly in the context of deep learning-based methods. We cover the core subjects and discuss the preprocessing and using language as a sequence for subsequent tasks in Section 2.1. We discuss some related methods we use in our later experiments in Section 2.2. We briefly introduce the tasks of semantic parsing and data-to-text generation (Section 2.3&2.4). We give an overview of their history and existing works, along with their applications and current challenges. We also discuss other relevant tasks to the thesis in Section 2.5.

2.1 Prerequisites

In this section, we discuss the preliminary topics related to deep learning-based language processing.

2.1.1 Artificial Neural Network (ANN)

An artificial neural network (ANN) is a model, consisting of interconnected nodes organized into multiple layers with weight matrices. The primary goal of an ANN is to generalize a specific task by learning from observed examples and adjusting its weight values through a process called training. During training, the network iteratively modifies the weights based on the difference between predicted and actual outputs to approximate the desired function.

2.1.2 Tokenization

Tokenization converts raw text into smaller units called tokens, creating a vocabulary from the given corpus and mapping each sentence to a sequence of elements from this vocabulary.

The simplest form, *Word-based tokenization*, splits text based on whitespace and punctuations. While easy to implement, it results in a large vocabulary size and issues with out-of-vocabulary (OOV) words because it treats words with the same root differently.

Character tokenization addresses some of these issues by splitting text into individual characters, reducing the vocabulary size, and handling OOV words more efficientlyKalchbrenner et al. [2016], Lee et al. [2017]. However, this approach can be inefficient in learning relationships among tokens due to the broad range of characters involved.

Subword tokenization strikes a balance by splitting text into meaningful subword units through the merging of frequently co-occurring characters Wu et al. [2016], Sennrich et al. [2016], Devlin et al. [2019], Kudo and Richardson [2018]. This technique effectively handles unknown words without resorting to single-character tokens, creating a more manageable and useful vocabulary. One common algorithm for subword tokenization is Byte-Pair Encoding (BPE), which iteratively merges the most frequent character pairs to form subwords. Subword tokenization has proven particularly effective in applications like Neural Machine

Translation (cf. Section 2.5.2), where it efficiently balances vocabulary size and the handling of unknown words.

2.1.3 Word Embeddings

Word embeddings are vector representations of words mapped into an n-dimensional real space. These vectors should ideally reflect the properties of the words, such that similar words have closer embeddings. This helps models represent word meanings and relationships, aiding in contextual information during training. Larger dimensions capture more information but may require substantial data for effective learning. Various methods, including neural networks, statistical methods, and graph-based methods, are used to learn these embeddings.

2.1.4 Sequential Learning

Sequential learning in NLP refers to the process of teaching models to understand and generate sequences of text or speech progressively over time. This approach is well suited for modeling any tasks where context and order play a significant role, such as language modeling, machine translation, speech recognition, and dialogue systems.

Popular methods and mechanisms used in Sequential Learning

Recurrent Neural Networks Recurrent Neural Networks (RNNs) are specialized neural networks that process sequences by repeatedly applying the same network layer (cell) to each token in the sequence, using previous outputs as inputs along with the current input at each time step. Initially proposed in the 1980s Rumelhart and McClelland [1987], RNNs gained significant traction in the 2010s due to their ability to handle context-based tasks such as natural language generation and machine translation. Unlike traditional feed-forward networks, RNNs maintain hidden states, making them suitable for tasks where the relationship between elements in a sequence is important for prediction. However, RNNs often struggle with handling longer contexts due to vanishing or exploding gradients.

Long Short-Term Memory Long Short-Term Memory (LSTM) networks Hochreiter and Schmidhuber [1997], a type of RNN variant, address the issue of capturing long-term dependencies by incorporating a gated mechanism. LSTMs have proven effective in tasks requiring the model to learn and remember dependencies over longer sequences, surpassing the limitations of standard RNNs.

Attention-based models The attention mechanism allows models to focus selectively on important information rather than processing all elements uniformly.

In its operation, the attention mechanism computes alignment scores, $f(q, k)$ between a query vector q and key vectors k. These scores are then normalized into weights using the softmax function, which is applied to value vectors v to compute a weighted sum. This sum updates a representation vector c_i at each

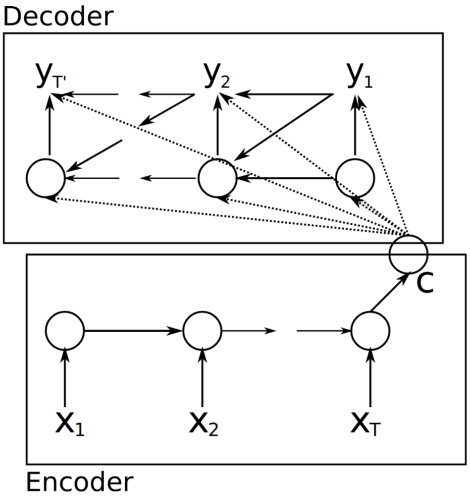

Figure 2.1: An example of sequence to sequence learning

position i, reflecting the importance of each element in the sequence, i.e.,

$$
\begin{aligned}
e_{ij} &= f(q_i, k_j) \\
\alpha_{ij} &= \text{softmax}(e_{ij}) \\
c_i &= \sum_{j=1}^{n} \alpha_{ij} v_j
\end{aligned}
$$

Various approaches exist for calculating alignment scores. Bahdanau et al. [2014] proposed an additive attention mechanism, while Luong et al. [2015] introduced multiple methods, including location-based scoring and generalized dot product functions. These advancements have significantly enhanced models' ability to process and generate sequences by focusing on contextually relevant information. Vaswani et al. [2017] introduced a scaled dot product-based alignment score calculation. This approach simplified the calculation of alignment scores.

RNN-based Sequence-to-Sequence

Cho et al. [2014] introduced an encoder-decoder architecture based on Recurrent Neural Networks (RNNs) aimed at handling variable-length inputs for learning phrase translations in Statistical Machine Translation (SMT). This model, depicted in Figure 2.1, employs an RNN encoder to convert input sequences into a fixed-size continuous representation known as a hidden state. This hidden state, combined with previously translated words, is decoded to produce corresponding words in the target language.

Building on this, Sutskever et al. [2014] expanded the concept into end-to-end sequence-to-sequence Neural Machine Translation (NMT) models. They adopted an encoder-decoder structure based on LSTM networks, enabling translation of entire sentence pairs rather than just phrases. Their model incorporated stacked LSTM layers in both the encoder and decoder.

Additionally, Bahdanau et al. [2014] introduced the attention mechanism within the encoder-decoder framework. This mechanism enhanced the model's

ability to selectively focus on relevant parts of the input sequence during decoding. Specifically, the encoder generates hidden states h_i which are concatenated and used to inform the decoder's prediction of the target word at each time step, improving translation accuracy and fluency.

Transformers and beyond

Vaswani et al. [2017] introduced the Transformer, an attention-based encoder-decoder framework. After preparing the input embeddings, positional embeddings are calculated and added to the input to preserve the sequential order and maintain the contextual information. Unlike sequence-to-sequence models, the Transformer's encoder processes the entire input sentence through multiple layers of self-attention, to learn the relations among the input tokens independent of the distance. It is then passed through feed-forward networks to create a vector representation (refer to Figure 2.2). This allows to parallelize training and facilitates model scaling. The encoded representation, along with the output embeddings, is then used by each decoder step to calculate cross-attention scores, using information from both previous decoder outputs and encoder outputs. The output embeddings are masked at each step to prevent the model from attending to the tokens from the later time step. With further transformations on the scores, the next word is predicted using the final distribution on the output embedding.

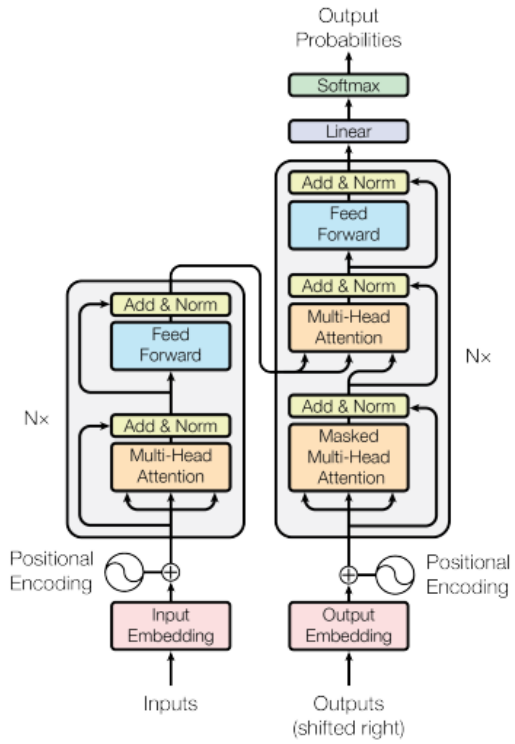

Figure 2.2: Transformers architecture Vaswani et al. [2017]

Pretrained Language Models

Pretrained language models (PLMs) represent a significant advancement in NLP, where models are first trained on large, diverse datasets using self-supervised

learning techniques before being fine-tuned on specific tasks. These models are usually transformer-based and can have different shapes. Encoder-only models like BERT Devlin et al. [2019](Bidirectional Encoder Representations from Transformers), decoder-only models like GPT (Generative Pre-trained Transformer)Radford et al. [2019], and their variants, learn rich representations of language in a self-supervised fashion, by predicting missing words in sentences or generating text based on context.

Sequence-to-Sequence PLMs With a similar architecture to Transformer, encoder-decoder models like T5 (Raffel et al., 2020) adopt a text-to-text framework for various NLP tasks, simplifying them into a unified format using task-specific prefixes. For instance, tasks like translation are indicated with prefixes such as "Translate from English to French:". T5 is pretrained in a self-supervised fashion that involves random sampling and removing 15% of the tokens to generate training examples. It is then further fine-tuned on supervised training on datasets like GLUE and SuperGLUE.

BART Lewis et al. [2019] utilizes a text-infilling corruption technique during pretraining, where segments of text are replaced with mask tokens for the decoder to predict. Other works use more sophisticated self-supervision, such as masking and predicting entire/partial sequences Zhang et al. [2020b].

These models demonstrate diverse approaches to leveraging Transformer architectures for specific NLP applications, each optimizing pretraining objectives and strategies to achieve state-of-the-art performance in their respective domains.

Recent Advancements and Large Language Models Recent advancements in text understanding and generation have been propelled by the emergence of extremely large language models such as GPT Brown et al. [2020], Falcon Almazrouei et al. [2023], Gemini Team et al. [2023], and LLaMA Touvron et al. [2023]. These models, characterized by their massive scale and billions of parameters, with better training strategies, have significantly improved their capabilities in understanding and generating human-like text. The models are even sometimes capable of solving a general-purpose task in a zero-shot prompting, i.e., the model generates relevant outputs using just the instruction (prompt) along with the input.

2.1.5 Finetuning methods

Fine-tuning involves adapting the parameters of a pre-trained large language model to suit a particular domain or task. The model is further trained on the downstream task requiring lesser training data. fine-tuning exposes the model to domain/task-specific data, enhancing its accuracy and applicability for targeted domains/tasks.

Adapters

With the growing scale of LMs in recent times, fine-tuning whole models can sometimes be very compute-intensive, and training a separate model for each downstream task can become quite impractical. To address this issue, adapter

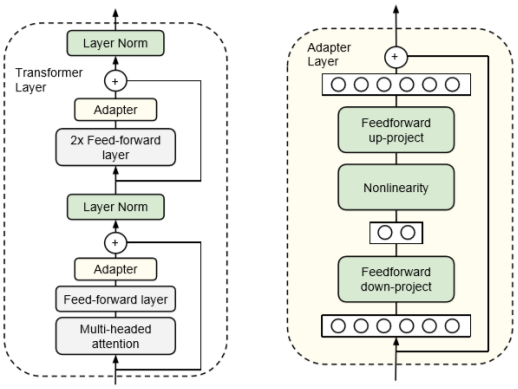

Figure 2.3: An example of adapter module Houlsby et al. [2019]

modules are proposed Houlsby et al. [2019], where only a small number of trainable parameters are added and tuned per task (cf. Fig 2.3). This approach allows models to efficiently generalize to new tasks and domains.

LoRA A popular strategy for finetuning large models more efficiently involves incorporating smaller trainable matrices, typically within the attention blocks Hu et al. [2021]. LoRA adds two low-ranked matrices decomposed from the delta weight matrix. The low-ranked matrices will further be trained during finetuning. These matrices are updated while the original weight matrix of the pretrained model remains frozen. This approach significantly reduces the number of trainable parameters, leading to reduced memory usage and shorter training times.

Prefix Tuning Prefix tuning Li and Liang [2021] offers a lightweight alternative to finetuning large pretrained language models for natural language generation tasks. The method involves prepending a small continuous vector, prefix, to each transformer layer. This prepended vector is then learned during the training. This method allows subsequent tokens to use the prefix as "virtual tokens" inspired by prompting techniques.

Prompt Tuning Proposed as a simplification of prefix tuning, prompt tuning Lester et al. [2021] adds task-specific prompts to the input and is trained independently of the frozen pretrained model parameters. Unlike discrete text prompts used in GPT-3-like models, the approach learns "soft prompts" through backpropagation, adapting them to incorporate information from the labeled examples. We will experiment with this method in the context of data-to-text generation in subsequent chapters (Section 5.2).

2.1.6 Decoding Strategies

We discuss various decoding strategies for auto-regressive language generation:

- **Greedy Search** It is the most straightforward decoding strategy. It selects the token with the highest conditional probability at each time step t (cf.

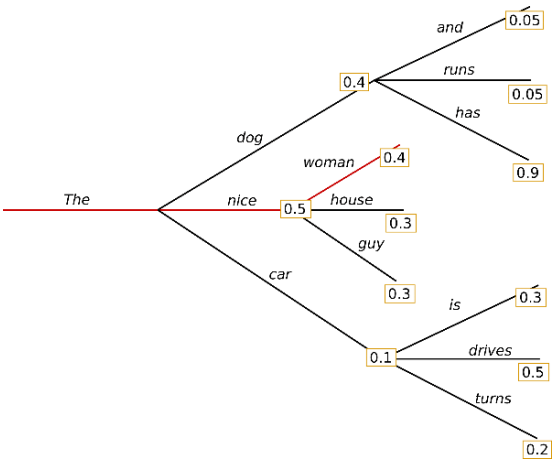

Figure 2.4: An example of greedy search decoding [1]

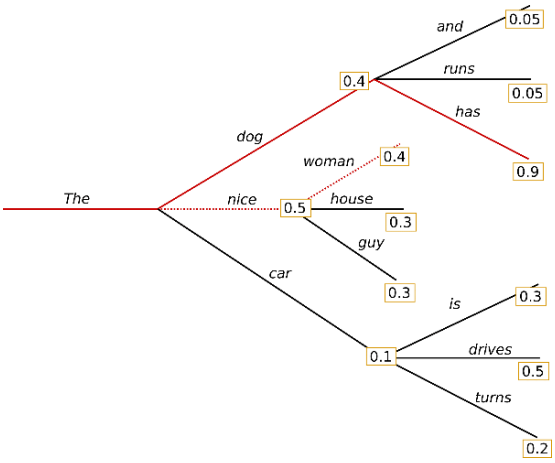

Figure 2.5: An example of beam search decoding [2]

Fig 2.4).

- **Beam Search** The previous method might miss the overall best sequence, which would be decoded by initially opting for a less probable token. To address this, at each step, each of the hypotheses is expanded up to *num_beams* times, and the overall best *num_beams* beams are retained (cf. Fig 2.5).
- **Sampling** In this decoding strategy, the tokens are sampled randomly based on the conditional probability distribution at each time step t.

Another widely used utility for text generation is *repetition penalty*. It is used to lower the likelihood of producing tokens that have recently been used in the generated text. This technique promotes the generation of more diverse and less repetitive text Keskar et al. [2019].

[1] `https://huggingface.co/blog/assets/02_how-to-generate/greedy_search.png`
[2] `https://huggingface.co/blog/assets/02_how-to-generate/beam_search.png`

2.2 Related Methods

2.2.1 Multi-task learning

Multitask learning (MTL) involves training models on various tasks concurrently, enhancing their ability to generalize across different tasks and domains. This approach has become standard with models like T5, BERT and GPT-3. MTL, including its variant transfer learning, where models are first trained on large datasets and then fine-tuned on specific tasks, is highly effective due to its ability to enhance model robustness and performance across diverse domains. Liu et al. [2019] introduce MT-DNN, a Multi-Task Deep Neural Network for learning representations across multiple natural language understanding (NLU) tasks. MT-DNN builds upon BERT by incorporating a pre-trained bidirectional transformer model, achieving state-of-the-art results on ten NLU tasks. They show that the model can achieve effective domain adaptation with reduced in-domain labels compared to pre-trained BERT representations. Raffel et al. [2020] introduce a unified text-to-text framework and systematically compares various pre-training objectives, architectures, datasets, transfer methods, and other factors across numerous language understanding tasks. Their method achieves state-of-the-art results in summarization, question answering, text classification, and beyond. Sanh et al. [2022] explores enhancement in zero-shot behavior of the models with explicit multitask learning. Their approach benchmarks a fine-tuned encoder-decoder model on a broad spectrum of tasks, achieving robust zero-shot performance on standard datasets and surpassing larger models in some cases. To generate robust outputs, several works have proposed adding an auxiliary task for classifying consistency in the generated outputs Peng et al. [2021], Kulhánek et al. [2021]. Generally, the outputs are corrupted with some probability p, and the decoder's last step logits are then used for the prediction of corruption. The loss derived from this task is then added to the overall loss.

2.2.2 Multilingual models

Multilingual models have advanced significantly, aiming to handle diverse languages with a single unified architecture. Models like mBARTLiu et al. [2020b], XLM-R Conneau et al. [2020], and mT5 Xue et al. [2021] are prominent examples, trained on multilingual datasets to perform NLP tasks across various languages. Recent research emphasizes expanding model capabilities beyond major languages to include under-represented languages, addressing challenges such as data scarcity and linguistic diversity. In this thesis (Chapter 4&5), we will explore using such models for data-to-text and text-to-data tasks for low-resource languages like Breton, Maltese, Irish and Welsh.

2.3 Semantic Parsing

Semantic parsing is the task of mapping natural language utterances into structured representations. These representations vary from SQL queries to smartphone instructions. Semantic parsing is often employed in virtual assistants and chatbots to interpret user commands and generate appropriate responses Gur

et al. [2018], Gupta et al. [2018]. The task can also be applied to data analysis and exploration by constructing queries from natural language Tang and Mooney [2001], Zhong et al. [2017], Yu et al. [2018]. Other applications include generating programming codes Yin and Neubig [2017], Lin et al. [2018].

While having many applications in real life, parsing can sometimes be not trivial due to ambiguity in natural language. Also, the lack of datasets in varied domain, makes the task more challenging.

We start by giving a formal definition and brief overview of its types (Section 2.3.1). Furthermore, we discuss its history and how its study has evolved after the inception of deep neural networks (Section 2.3.2). We give an overview of neural-based semantic parsing techniques and the existing datasets. We discuss the evaluation metrics in Section 2.3.3.

2.3.1 Introduction

Figure 2.6: A semantic parsing example [3]

Semantic parsing refers to the process of transforming natural language expressions into meaning representations known as logical forms, such that the logical forms are easily comprehensible by machines (cf. Figure 2.6). These meaning representations can have various interpretations for different tasks.

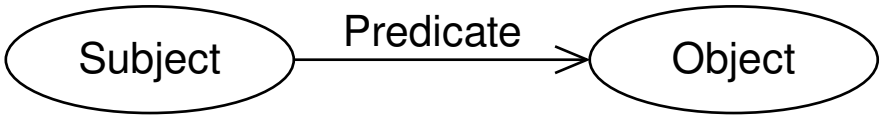

Figure 2.7: A triple in a RDF graph [4]

One of the popular tasks in semantic parsing is converting natural language to RDF triples. RDF, or Resource Description Framework, is used to represent information on the web. It offers capabilities that enable the merging of data, even when the underlying schemas vary. Additionally, RDF provides dedicated support for the gradual evolution of schemas without necessitating modifications to all data consumers. Within this framework, the information is stored in the form of a graph, with each graph consisting of a set of triples. Each triple contains two nodes, subject and object, and a directed arc (predicate) connecting them (Figure 2.7).

In this work, we will mostly focus on RDF-based semantic parsing using the WebNLG data. We will discuss this further in Chapter 2

[3] https://guide.allennlp.org/part3/semantic-parsing/alexa-as-semantic-parser.svg
[4] https://www.w3.org/TR/rdf12-concepts/rdf-graph.svg

2.3.2 History and Related Works

Semantic parsing, a field with roots dating back to the 1970s, initially focused on understanding natural language through rule-based and syntax-based systems, like SHRDLU Winograd [1971] and LUNAR Woods [1973], which struggled with scalability and natural language variations. Early efforts were domain-specific and labor-intensive, leading to limited adaptability Warren and Pereira [1982], Johnson [1984]. Significant advancements occurred in the 1990s with corpus-based methods, notably by Zelle and Mooney, who introduced models converting sentences to database queries for evaluation Zelle and Mooney [1996]. Subsequent improvements included learning lexicons paired with meaning representations Thompson [2003] and employing statistical techniquesZelle and Mooney [1996], Zettlemoyer and Collins [2012], Kwiatkowski et al. [2011], such as Probabilistic Combinatory Categorial Grammar (PCCG), to rank parses probabilistically. Despite enhancements like dynamic programming and factored lexicons Manning and Schutze [1999], challenges remained with variable word order and colloquial language, prompting exploration of alternative supervision methods to alleviate manual annotation difficulties.

After their introduction in machine translation and other tasks, end-to-end sequence-to-sequence approaches became applicable in the task of semantic parsing as well. Dong and Lapata [2016] propose a decoder that considers the tree structure of logical expressions to capture hierarchical information and parentheses. The decoder uses parent feeding connections and soft attention, performing comparably to previous feature-based systems. One challenge is learning parameters for rare words, which can be addressed by anonymizing entities or using Pointer Networks for word copying. Jia and Liang [2016] introduce attention-based copying, combining word copying and softmax selection. They also propose a generative model for data augmentation, sampling from a synchronous context-free grammar (SCFG) to generate more complex training examples. This boosts performance by improving generalization.

Gardent et al. [2017a] introduced an RDF-based semantic parsing dataset, along with organizing the WebNLG challenge. There have been two more iterations of the WebNLG challenge since then. We discuss the challenges in detail in Chapter 3. While the initial challenge focused only on the verbalization of RDF triples, the following challenge had tasks in both directions, semantic parsing and verbalization. There have been various interesting works since its release. Zeng et al. [2018] proposed an end-to-end seq2seq model with a copying mechanism to extract the relations from the given natural language. In another end-to-end relation extraction work, Fu et al. [2019] use graph convolutional networks (GCNs) to parse the input text to relational triples. Liu et al. [2020a] address the issue of overlapping relation in the triples. They use multi-head self-attention for relation detection and entity extraction. Some other works include those of Sun et al. [2020], Wang et al. [2020], Ye et al. [2021], Chen et al. [2021]. Sun et al. [2020] introduce a multi-task learning-based model for extracting and classifying relations. Wang et al. [2020] perceived extraction task as a token pair linking problem, and proposed TPLinker, an extraction model capable of extracting multiple relations and overlapping entities. Ye et al. [2021] proposed a contrastive training objective to capture long-term dependencies better. Chen et al. [2021] used pointer networks to extract overlapping triples corresponding to each word.

Ref Class	Ref String	Pred Class	Pred String	Type	Partial	Exact	Strict
SUB	john			MIS	MIS	MIS	MIS
		SUB	john	SPU	SPU	SPU	SPU
SUB	john	SUB	john jr	COR	PAR	INC	INC
SUB	john	OBJ	john	INC	COR	COR	INC
SUB	john	SUB	john	COR	COR	COR	COR
SUB	john	OBJ	john jr	INC	PAR	INC	INC

Table 2.1: An example of metric types and error categories.

2.3.3 Evaluation

Evaluating the structured data is relatively easier than evaluating a fluent sentence. One of the popular metrics to evaluate semantic parsing tasks is *exact match*, where the candidate is simply matched against the reference and is given the score 1 if there is an exact match and 0 otherwise. Building upon that, WebNLG 2020 organizers used metrics based on Named Entity Recognition task (Section 2.5.1). Given a reference and candidate's RDF triple entities (and their classes), they evaluate them in four different types:

- **Exact Match** The candidate triple element should exactly match the reference triple element, but the element type (subject, predicate, object) does not need to match.
- **Strict Match** The candidate triple element should exactly match the reference triple element, including the type (subject, predicate, object).
- **Partial Match** The candidate triple element should at least partially match the reference triple element, with the element type (subject, predicate, object) being irrelevant.
- **Type Match** The candidate triple element must at least partially match the reference triple element, and the element type (subject, predicate, object) should also match.

For each evaluation type, F1 scores are calculated. In order to calculate the F1 scores, the candidate triple entities are first pre-processed and lower-cased. The reference and candidate triples are then linearized and converted to strings. Furthermore, their information on start-end indices is stored. Separately, the entities are paired with their corresponding classes (subject, object, verb). Consequently, counts in five different categories are calculated:

- **Correct (COR)** both entities and classes are same as the reference
- **Incorrect (INC)** no match between reference and candidate
- **Partial (PAR)** some similarity between reference and candidate (partial word overlap)
- **Missing (MIS)** some part of the reference entity is missing in the candidate

- **Spurious (SPU)** some part of the candidate entity is missing in the reference

Table 2.1 shows an example of how such error categories are identified for each metric type. After getting the scores for each category in a metric type, two more quantities are calculated:

$$\text{ACTUAL}(ACT) = COR + INC + PAR + SPU$$

$$\text{POSSIBLE}(POS) = COR + INC + PAR + MIS$$

Using these quantities, precision, recall and F1 are calculated for each metric type. More specifically, precision and recall for *Strict* and *Exact* matching are calculated as:

$$\text{Precision} = \frac{COR}{ACT}$$

$$\text{Recall} = \frac{COR}{POS}$$

. On the other hand, for *Type* and *Partial* matching, they are calculated as:

$$\text{Precision} = \frac{COR + 0.5 \times PAR}{ACT}$$

$$\text{Recall} = \frac{COR + 0.5 \times PAR}{POS}$$

.

2.4 Data-to-Text Generation

Data-to-text (D2T) generation refers to the task of automatically generating text from structured data. This includes applications like generating sports articles, weather reports, verbalization of dialogue acts, etc. D2T generation has wide applicability in the area of journalism as well. Theune et al. [2001], Chen and Mooney [2008] generated soccer reports using D2T generation methods. Similarly, news articles and weather reports can be generated given the data Leppänen et al. [2017], Goldberg et al. [1994], Reiter et al. [2005]. Another real-world application of D2T is summarizing patient information in a clinical report Hüske-Kraus [2003], Banaee et al. [2013], Gatt et al. [2009]. It also involves adapting the text style to the user's needs Mahamood and Reiter [2011].

With recent advances in LLMs, the language generation task has reached human-level performance in some aspects, especially fluency. Even with an appropriate amount of data, the models tend to hallucinate and generate irrelevant text confidently Ji et al. [2023]. Also, most of the existing datasets for D2T generation task are in the English language, restricting the applicability of such models in other widely spoken languages. Furthermore, there is also a problem of the lack of diversity in the domains of existing datasets, and thus, D2T systems tend to be limited to only certain domains.

In this section, we will be discussing further about the task (Section 2.4.1). We will introduce the task and briefly describe the popular NLG pipeline by Reiter and Dale [1997]. We then go through the history and existing works in D2T generation (Section 2.4.2). We discuss the evaluation metrics in Section 2.4.3.

2.4.1 Introduction

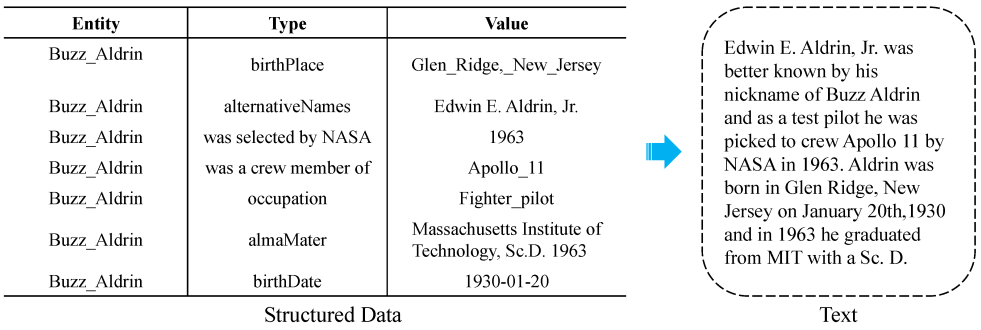

Figure 2.8: An example of D2T generation Gong et al. [2023]

Consider the example in Figure 2.8. On the left, we have structured data as a set of triples. The task of converting (verbalizing) such data to natural language is referred to as D2T generation. At higher levels, this verbalization can also have a wide variety according to the user's needs. The generated text can either be concise or more verbal and use either generalized or in-domain terms. This customizability makes this task even more challenging. However, the sole objective is to produce fluent text with relevant content to the given input.

Reiter and Dale [1997] introduced a generalized NLG pipeline. It became quite popular in the field. The pipeline constitutes tasks at various levels, with the initial ones being content-oriented and the final ones being more linguistically oriented. It goes from content determination, where the relevant information is extracted, to structuring extracted information. Sentences are then formed using structured information, and suitable words and phrases are used. It is then made more specific to the required domain, and finally, the sentences are combined together to construct meaningful sentences.

With the advent of seq2seq models, the task of NLG has also been made end-to-end. However, such systems are data-hungry. Also, because of being end-to-end, unlike the pipeline approach, there is no submodule within the framework and, thus, cannot be reused independently.

2.4.2 History and Related Work

Earlier modular approaches to text generation were mostly rule-based, but recent methods have become more statistical, utilizing techniques like clustering for content selection Duboue and McKeown [2003], Barzilay and Lapata [2005] and machine learning for text structuring Barzilay and Elhadad [2002], Bollegala et al. [2010]. Template-based generation for linguistic realization was proposed by Reiter [1995], McRoy et al. [2003], with recent advancements learning templates automatically Angeli and Manning [2014], Kondadadi et al. [2013]. Planning-based approaches, viewing text generation as an AI planning task, have also been proposed, using Reinforcement Learning for stochastic planning Lemon [2008], Rieser and Lemon [2009].

Since the introduction of RNN seq2seq architecture, there have been several works in end-to-end architectures for the D2T task. Structured data can also be considered as a sequence of information; thus, using it in a seq2seq setting makes

sense. Wen et al. [2015] used a semantically conditioned LSTM-based model for generating dialogue acts and realizations in the context of a dialogue system. Goyal et al. [2016] improved upon this work by using the network on the character level instead of the word. Castro Ferreira et al. [2017] framed the AMR(Abstract Meaning Representation)-to-Text task as a seq2seq task, and tested different MT systems on it. Puduppully et al. [2019] used the seq2seq model along with preserving the pipeline. They propose an end-to-end architecture to first highlight and generate the relevant content with appropriate ordering. Using the generated information, the model then generates the final text. The existing seq2seq methods ignore the structure of the input data and instead assume the data to be linear. Rebuffel et al. [2019] propose a hierarchical model that addresses this issue and represents the data on both structure and element level. Kale and Rastogi [2020] showed that even a simple end-to-end model with pretraining, like T5, can outperform the traditional pipeline-based methods to D2T generation task.

Since the introduction of the WebNLG dataset, numerous methods for generating text from RDF triples have emerged. Trisedya et al. [2018] developed a graph-based triple encoder within an encoder-decoder framework, while Shimorina and Gardent [2018] showed combining delexicalization and copying improved handling of rare terms. Jagfeld et al. [2018] compared word and character-based seq2seq models for the task. Zhao et al. [2020] introduced DualEnc for better text quality by incorporating both the graph structure and the linear structure of the output text. Kasner and Dušek [2020a] utilized an iterative text editing approach for the task. Chang et al. [2021] proposed a few-shot approach using cycle consistency, and Wang et al. [2023b] employed cycle training for low-resource settings. Recent methods leverage large language models, such as Xiang et al. [2022]'s two-step triple disambiguation and sentence fusion using GPT-3 and T5, and Lorandi and Belz [2024]'s effective use of LLMs for low-resource languages, showing significant improvements over previous methods.

2.4.3 Evaluation

In this section we discuss the common metrics to evaluate the quality of the generated texts. We describe two types for metrics: model-free and model-based.

Model-free metrics

These metrics are usually based on rule or statistics. Here are some commonly used model-free metrics:

- **BLEU** Introduced by Papineni et al. [2002], BLEU is a widely accepted metric for evaluating language generation quality, given the gold references. It calculates the weighted modified precision of word overlap in candidate and reference texts. It is calculated as the product of BP and geometric mean p_n, i.e.,

$$BLEU = BP \cdot exp(\sum_{n=1}^{N} w_n log p_n)$$

  where, BP is the Brevity Penalty, used to penalise short sentences. It takes

reference length r and candidate length c and is calculated as

$$BP = exp(min(0, 1 - \frac{r}{c}))$$

We calculate p_n using

$$p_n = \frac{\sum_{C \in \{candidates\}} \sum_{n-gram \in C} count_{clip}(n - gram)}{\sum_{C' \in \{candidates\}} \sum_{n-gram' \in C'} count(n - gram')}$$

where $count_{clip}(n - gram)$ is used to address unwanted repetitions in the candidate text and is calculated as

$$count_{clip}(n - gram) = min(\text{count, max\_ref\_count})$$

While used extensively, BLEU fails to identify grammatical mistakes and is focused only on surface-level matching.

- **ROUGE** Lin [2004] proposed a recall-based metric to evaluate the quality of the generated texts. They proposed various variants of the metric. ROUGE-N measures recall of n-gram overlap between reference and candidate texts. ROUGE-L calculates the score based on the longest common subsequence between the reference and candidate texts. More specifically, ROUGE-L is calculated as an F measure F_{lcs},

$$F_{lcs} = \frac{(1 + \beta)^2 R_{lcs} P_{lcs}}{R_{lcs} + \beta^2 P_{lcs}}$$

where

$$R_{lcs} = \frac{LCS(X, Y)}{m}$$

and

$$P_{lcs} = \frac{LCS(X, Y)}{n}$$

m and n are the lengths of the reference and candidate sentences X and Y respectively. β can be used to control the importance of precision and recall in the formula.

- **METEOR** Proposed in Banerjee and Lavie [2005], METEOR combines the both precision and recall into a single metric. It takes unigram precision (p) and recall (r) and calculates the harmonic mean F_{mean} as

$$F_{mean} = \frac{p \cdot r}{\alpha p + (1 - \alpha) r}$$

where the weight α is usually higher to make METEOR biased towards recall. A chunk penalty *pen* is added to penalise the candidates with more number of chunks (set of consecutive words). It is calculated as

$$\text{pen} = 0.5(\frac{\#\text{chunks}}{\#\text{unigram\_matched}})^3$$

Finally, the score is calculated as

$$M = F_{mean}(1 - \text{pen})$$

- **chrF++** It evaluates n-grams at the character level rather than at the token level Popović [2015]. This characteristic allows chrF++ to function independently of language and tokenization. It is calculated as,

$$chrF = \frac{(1+\beta)^2 R_{chr} P_{chr}}{R_{chr} + \beta^2 P_{chr}}$$

where R_{chr} and P_{chr} are recall and precision on character-level n-gram overlaps.

- **Perplexity** Perplexity in text generation quantifies the uncertainty a model experiences when predicting a new sequence of words. It evaluates the probability of word sequences, comparing the model's predicted distribution of words with the actual sequence. A lower perplexity score indicates that the model is better at generating text with diverse inputs because it is less confused. However, perplexity solely measures the model's confidence and does not necessarily reflect the quality of the generated text.

- **Translation Error Rate (TER)** TER Snover et al. [2006] is the minimum number of edits taken to convert the hypothesis to match one of the references. Explicitly,

$$TER = \frac{\#\text{edits}}{\text{Avg } \# \text{ reference words}}$$

where, edits can be: (1) deletion – the number of words that are present in the reference sentence but not in the generated sentence, (2) insertion – the number of words present in the generated sentence but missing from reference, (3) substitution – the number of words altered between reference and candidate.

Model-based Metrics

With LMs' increasing capability of understanding languages, they are being used to evaluate the texts. Such metrics are often capable and used to measure the semantic similarity between the texts. Following are some of the popular model-based metrics:

- **BERTScore** It is a metric that evaluates text quality by analyzing contextual token similarity Zhang et al. [2019]. It uses pairwise cosine similarity of word embeddings to compare a generated text with a reference. By incorporating precision, recall, and F1 measures, BERTScore is versatile for different text generation tasks and aligns well with human evaluations. This metric is highly adaptable and can be adjusted for specific aspects like fluency, style, and coherence, although its reliability may differ across these aspects.

- **BLEURT** It is a metric trained on human judgment of text quality, known for its adaptability and ability to be fine-tuned for various tasks Sellam et al. [2020]. Despite its comprehensive training, hybrid metrics like BERTScore can perform better with limited training data and do not require similar

distributions between training and test data. BLEURT aims to measure text adequacy, but due to training practices and added synthetic noise, it often correlates closely with fluency, blurring the distinction between these two aspects.

2.5 Other Relevant Tasks

In this section we will discuss tasks useful to our methods. We discuss the task of named entity recognition (useful for the semantic parsing task) in Section 2.5.1 and neural machine translation (useful for multilingual text generation) in Section 2.5.2.

2.5.1 NER

Named Entity Recognition (NER) is a crucial task in natural language processing (NLP) that involves identifying and classifying entities in text into predefined categories such as names of people, organizations, locations, dates, and more. Thus, it involves two steps, identifying the entity and classifying it. For instance, given the input text, "John lives in Prague" the corresponding tagged text would be "$[\text{John}]_{\text{PER}}$ lives in $[\text{Prague}]_{\text{LOC}}$."

The task has undergone significant advancements with pretrained models like BERT, GPT-3, and others Ashok and Lipton [2023], Wang et al. [2023a], Zhang et al. [2024]. By leveraging the embeddings generated by LLMs, NER systems can identify and classify entities with higher accuracy and robustness compared to traditional methods. This is particularly beneficial in handling ambiguous or context-dependent entities, which are challenging for rule-based or shallow learning approaches.

As generating RDF triples requires identifying named entities, we use the task in several experiments.

2.5.2 Neural Machine Translation

Neural Machine Translation (NMT) focuses on the automatic translation of text or speech from one language to another using neural sequential models. Unlike traditional methods that rely on phrase-based or statistical models, NMT uses encoder-decoder architectures with attention mechanisms, to provide more accurate and fluent translations. NMT models have shown remarkable performance improvements, often surpassing other translation models, and even have superhuman performance in some cases [Bojar et al., 2016] [Bentivogli et al., 2016] [Barrault et al., 2020]. However, they require large bilingual datasets for training, which poses a challenge for low-resource languages. To mitigate this, techniques such as transfer learning, back-translation, and the use of bilingual dictionaries have been employed to enhance the performance of NMT in resource-scarce scenarios. Models like NLLB Team et al. [2022] have a support of up to 200 languages for translation, capable of zero-shot translation.

We use translation systems and their zero-shot capabilities in our experiments to generate verbalizations in low-resource languages.

3. WebNLG Dataset

The WebNLG challenge involves generating natural language text from RDF data (and vice versa), with training pairs consisting of DBpedia triples and their corresponding verbalizations (cf. Fig.3.1). The challenge is a way of promoting the development of RDF verbalizers and microplanners for generating syntactically diverse and natural text and meaning representations. In this chapter, we describe the various iterations of the WebNLG challenges, the datasets, and the submitted systems. We use WebNLG 2020 and 2023 datasets for our experiments (Chapters 4&5).

RDF Triple:	⟨John_Donald, birthPlace, Prague⟩
Verbalisation:	John Donald was born in Prague.

Figure 3.1: An example of RDF Triple and its verbalization. The input can have a maximum of 7 triples

3.1 WebNLG Challenge 2017

The first iteration of WebNLG Challenge Gardent et al. [2017b] involves the task of mapping data to text. The edition was only for the text generation task. The training set consists of parallel data consisting of RDF triples extracted from DBPedia and their corresponding verbalizations. Each instance in the WebNLG dataset is a tree with different shapes and sizes. An example of such a tree is given below:

```
<entry category="Artist" eid="Id5" shape="(X (X) (X))" shape_type="sibling" size
    ="2">
      <originaltripleset>
        <otriple>Abradab | birthPlace | Katowice</otriple>
        <otriple>Abradab | placeOfBirth | Poland</otriple>
      </originaltripleset>
      <originaltripleset>
        <otriple>Abradab | birthPlace | Poland</otriple>
        <otriple>Abradab | placeOfBirth | Katowice</otriple>
      </originaltripleset>
      <modifiedtripleset>
        <mtriple>Abradab | birthPlace | Katowice</mtriple>
        <mtriple>Abradab | birthPlace | Poland</mtriple>
      </modifiedtripleset>
      <lex comment="good" lid="Id1">Abradab was born in Katowice, Poland.</lex>
      <lex comment="good" lid="Id2">The birth place of Abradab is Poland, in
    Katowice.</lex>
      <lex comment="good" lid="Id3">Abradab was born is Katowice, Poland.</lex>
</entry>
```

`shape="(X (X) (X))"` and `shape_type="sibling"` are string representations of the tree shape and type, respectively. The tree shape can be of three types: (1) chain (object in triple1 is a subject of triple2), (2) sibling (all triples share the same subject), (3) mixed (both types are present). `category="Artist "` represents the triple category (or domain) from which the RDF triple is extracted. "originaltripleset" is a triple set directly extracted from DBPedia. "modifiedtripleset" refers to the triple set after several modifications. The modifications include merging triples with semantically similar properties, clarification of subjects and objects, merging the

same triples in different formats, etc. "lex" describes the verbalization of the given RDF triple. The verbalizations are obtained using crowdsourcing, using a four-step process: (1) removing out triples with no possible lexicalization, (2) generating three lexicalizations for a single triple, (3) merging sentences into fluent sentences for the case of more than one triple (4) verifying the quality of the verbalizations.

3.1.1 Dataset

The WebNLG 2017 Challenge dataset consists of a total of 21,866 training and dev instances combined, of RDF triples-verbalization pairs, in 9 distinct DBpedia categories — Astronaut, University, Monument, Building, ComicsCharacter, Food, Airport, SportsTeam, and WrittenWork. The triple set size varies from 1 to 7. The test set consists of around 1,700 pairs with an almost equal split between seen and unseen categories.

For evaluation, both automatic and manual analyses are done. BLEU, TER, and METEOR are used as automatic evaluation metrics. The human evaluation is a likert scale and crowdsourced.

3.1.2 Baseline and Submitted Systems

The 2017 baseline system Gardent et al. [2017a] consists of three steps: (1) Preprocessing data (2) Training model for verbalization (3) Relexicalisation. For the data preprocessing step, the triples are first linearized. They are then tokenized and delexicalized using exact match (cf. Fig 3.2). An LSTM-based seq2seq model (cf. Section2.1.4) is then trained using the OpenNMT toolkit with default parameters. Finally, the verbalizations are relexicalized.

RDF Triple:	⟨John_Donald, birthPlace, Czechia⟩ ⟨John_Donald, play, football⟩
Linearize:	John_Donald birthPlace Prague John_Donald play football
Delexicalization:	ATHLETE birthPlace COUNTRY ATHLETE play SPORT

Figure 3.2: An example of data preprocessing step of WebNLG 2017 baseline system.

The first edition of the WebNLG challenge received various submissions, including template or grammar-based pipeline framework, statistical machine translation, and NMT framework. While all the systems performed well on the whole dataset, with each type of framework being in the top four, the machine learning-based systems outperform the rule-based ones on the seen data. A similar trend is observed for unseen data as well, except for the NMT system that fails to handle rare words

3.2 WebNLG Challenge 2020

The second edition of the WebNLG Challenge Castro Ferreira et al. [2020] adds a new language, Russian, and a new task of text-to-RDF semantic parsing. The

data instances still have the same format, except that extra information on language is added in the lexicalization tags.

3.2.1 Dataset

The English training set consists of a total of 35,426 lexicalization-triples pairs with 13,211 different triple sets. Unlike 2017, the WebNLG 2020 dataset now contains entries in 16 different categories — 10 seen categories from the WebNLG 2017 dataset, 5 unseen categories from the same edition (Athlete, Artist, CelestialBody, MeanOfTransportation, Politician), and a new category, Company. The development set contains 4,464 lexicalizations in 1,667 entries. This edition of the challenge split the test set into three types — (1) seen categories, (2) seen categories with unseen entities, and (3) unseen categories. The test set for the verbalization task consists of 1,779 text-triple pairs, with 490, 393, and 896 instances in seen categories, unseen entities, and unseen categories, respectively. It consists of a maximum of 5 references for each test item. For the RDF parsing task, the test set consists of a total of 2,155 pairs with 606, 457, and 1092 pairs in seen categories, unseen entities, and unseen categories, respectively.

The Russian WebNLG dataset was created in 3 steps: (1) translation of English lexicalizations into Russian using Sennrich et al. [2017]'s system, (2) post-processing using crowdsourcing, and (3) sanity checks and spell checking by Russian native speakers. The dataset consists of 14,630 and 2,065 data-text pairs in training and development sets with entries in 9 distinct categories (Airport, Astronaut, Building, CelestialBody, ComicsCharacter, Food, Monument, SportsTeam, and University). The test set consists of 2,780 texts for the verbalization task and 1,206 texts for the RDF parsing task.

BLEU, METEOR, chrF++, TER, and BERTScore are used as automatic metrics to evaluate the verbalization task. For the RDF parsing task, the four metric types, Strict, Exact, Type, and Partial (described in Section 2.3.3) are used.

3.2.2 Baseline and Submitted Systems

Mille et al. [2019a]'s FORGe, a grammar and template-based generator, is used as a baseline for verbalization in both the languages, English and Russian. Another baseline Mille et al. [2019c], a submission in WebNLG 2017 challenge, is used as well. Unlike the second baseline, the first has improved grammar. Since such a grammar-based framework has no hallucinations, the baselines built on the same framework performed decently well in the human evaluation. Among the other English submissions in the RDF Verbalization task, Guo et al. [2020a] had one of the best performances. They proposed a plan-and-pretrain approach, with a relational graph convolutional network as the planner and a T5 model as the surface realization model. Li et al. [2020] used T5-large for English and mBART for Russian and fine-tuned the models on the WebNLG dataset. Yang et al. [2020] used BART [Lewis et al., 2019] for pretraining and they investigate various methods of encoding RDF graphs in relation to natural language text. They compare different linearization strategies and analyze the impact of integrating generation with document planning. Additionally, they apply a second phase of pre-training

using DocRED Yao et al. [2019] and curate the lexicalization of RDF properties from WebNLG+ and DocRED datasets. Kasner and Dušek [2020b] used mBART model for generating text in both the languages, English and Russian. Similar to Li et al. [2020], Pasricha et al. [2020] used pretrained T5 architecture and fine-tuned it on the WebNLG dataset. Agarwal et al. [2020] used single T5 for both the languages on both tasks, D2T and RDF-based semantic parsing. They additionally add machine translation as an auxiliary task in finetuning stage.

As a baseline for the RDF parsing task, Stanford CoreNLP's Open Information Extraction module [Manning et al., 2014] was used. The second edition of the WebNLG challenge received several submissions for the RDF-based semantic parsing task. Guo et al. [2020a] employed a two-step approach. They first identified the entities using entity linking from the text of DBpedia ontology. Ultimately, they queried the entities on the DBpedia database to extract the relation. Since this work involves jointly using two methods, independent from the training data, the performance on the unseen RDF categories was relatively good. Agarwal et al. [2020] trained a joint model capable of performing both tasks, parsing and text generation, in both languages. They also leverage machine translation for augmented data to get better performance. Guo et al. [2020c] proposed an unsupervised training method CycleGT, involving iterative back translation between text and triples.

3.3 WebNLG Challenge 2023

To address the multilingual challenges for data-to-text generation in the context of data scarcity for low-resource languages, WebNLG 2023 challenge was organized Cripwell et al. [2023]. It aimed to evaluate the latest advancements and stimulate progress in data-to-text NLG methods for low-resource languages by including five non-English languages. Four of these languages were new low-resource ones: Maltese (*Mt*), Welsh (*Cy*), Irish (*Ga*), and Breton (*Br*), alongside Russian (*Ru*) from the 2020 challenge.

3.3.1 Dataset

The training set for Breton, Irish, Maltese, and Welsh consists of 13,211 parallel entries. The training sets were created using the machine translation system Edinburgh ZeroZhang et al. [2020a], capable of zero-shot translations. Edinburgh Zero system is used to translate each lexicalization in the WebNLG 2020 English dataset. The development and test sets for the low-resource languages were created manually. The dev set consists of 1,399 instances for Breton and 1,665 for the other three languages, while the test set for all four languages contains 1,778 items.

The Russian dataset remains the same as the previous edition.

Automatic evaluation is based on BLEU, METEOR, chrF++, TER, and BERTScore. Additionally, human evaluation involving native speakers is also used to assess grammatical correctness, fluency, appropriateness, etc.

3.3.2 Baseline and Submitted Systems

The organizer-provided baselines for Russian include Kasner and Dušek [2020]'s finetuned mBART and the 2020 Challenge baseline, the FORGE system Mille et al. [2019b] coupled with Google Translate. The baseline for Breton, Irish, Maltese, and Welsh is the 2020 Challenge English system of Guo et al. [2020b], coupled with Edinburgh Zero MT Zhang et al. [2020a].

This edition of WebNLG received submissions from five teams Cripwell et al. [2023]. Kazakov et al. [2023] utilizes FRED-T5[1](1.7B parameters) to finetune on the WebNLG 2020 Russian dataset and has the best scores on automatic evaluation metrics. Mille et al. [2023]'s FORGe system uses a rule-based pipeline to generate Irish verbalization from the given RDF triples and has the second-best performance. Hari et al. [2023] finetune T5 to generate English verbalizations given the RDF triples. They use NLLB to translate English verbalizations into Irish, Maltese, Russian, and Welsh. Lorandi and Belz [2023] use GPT-3.5 and 4 in (1) direct generation and (2) verbalization and translation configurations. They submit their Irish, Maltese, and Welsh systems and perform the best in all those languages. We submitted a system in this edition as well. We describe our methods in detail in Section 5.1 Kumar et al. [2023].

[1] `https://huggingface.co/ai-forever/FRED-T5-1.7B`

4. RDF Triple-based Semantic Parsing

In this chapter, we discuss various techniques to enhance RDF triple-based semantic parsing within a seq2seq framework. The task of RDF triple-based semantic parsing involves generating a set of RDF triples given a fluent text (cf. Chapter 2). We utilize a standard seq2seq model (T5) and assess the effectiveness of different data-centric, loss-based, and auxiliary task-based methods (Section 4.1). Additionally, we investigate multilingual semantic parsing using the WebNLG 2023 dataset (Section 4.2) by experimenting with several different training strategies adapted for multilingual settings. We conclude the whole chapter in Section 4.3.

4.1 Improving English RDF Parsing

This section describes various strategies and methods to improve RDF triples-based semantic parsing in English (*En*). We begin by outlining the experimental setup in Section 4.1.1. The proposed methods are discussed in Section 4.1.2. Initially, we approach the RDF parsing task as a two-step task of span prediction (for subject/object) and predicate generation. For further experimentation, we treat it as a seq2seq task, establishing a baseline with an encoder-decoder model trained on the provided dataset. We then experiment with incorporating different auxiliary tasks and develop a data-centric method for better input-output alignment. The results of these approaches are compared in Section 4.1.3. We finally conclude the study in Section 4.1.4.

4.1.1 Experimental Settings

Data preprocessing

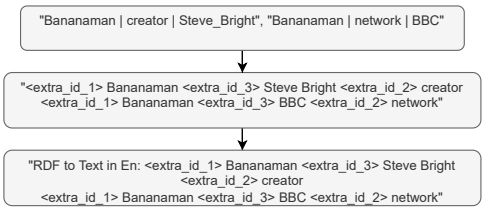

Figure 4.1: An instance of the preprocessing step

Except for *SpanGen* (cf. Section 4.1.2), we approach the RDF parsing task as a seq2seq one. We linearize the RDF triples and prepend the class token to each entity, i.e., we insert three new tokens to the model corresponding to the subject, object, and predicate. We sort the triples in subject-object-verb order (cf. Fig 4.1). Depending on the method, we add an additional prompt for T5, e.g. “Text to RDF” (in most cases).

Model Hyperparameters

For all experiments, unless specified otherwise, we use T5 Raffel et al. [2020] small to generate RDF triples. We train the models for 45 epochs (the model losses started plateauing at around 30 epochs), employing early stopping based on validation loss. We set batch sizes between 12 and 36 depending on the available GPU memory.

Evaluation

We evaluate the model on the four evaluation metric types described in Section 2.3.3. Furthermore, we evaluate the triples separately for seen and unseen categories.

4.1.2 Proposed Approaches

Span Prediction and Predicate Generation (SpanGen)

The task is to identify subjects and objects in a given text, along with their relation. In order to identify them, we can just predict the start-end tokens of the subjects and objects in the input text. Consequently, using an encoder-based system, we view it as a span prediction task to predict the subject and the object. We determine and create start-end spans of the indices for each subject and object entity. Since the dataset contains a maximum of 7 triples, we have 7 unique start-end target tuples separately for subjects and objects, resulting in a total of 28 classification heads. We then train a decoder-based system to generate the predicates, given input text and subject-object pair. We then train a decoder-based system to generate the predicates based on the input text and subject-object pairs. During inference, predicates are generated for all possible subject-object pairs. BERT-base (Devlin et al. [2019]) is used for span prediction, and GPT2 (Radford et al. [2019])is used for predicate generation.

However, this approach has a fundamental limitation. Mapping named entities from verbalizations to triples can be non-trivial. For instance, the example, "Input Text: American president Joe Biden was born on 20.11.1942. — Triples: ⟨Joe Biden, president, USA⟩ ...", has "American" in input mapped to "USA" in the output triple. This system is referred to as *SpanGen*.

Baseline

We consider the semantic parsing task as a seq2seq and use an encoder-decoder model to parse the *En* texts. We observed improved performance with the addition of the task prompt "Text to RDF". Consequently, we use the prompt in all subsequent experiments. This system is referred to as *baseline*.

Delexlicalized Semantic Parsing (Delex)

The training data contains a significant number of named entities. To make the model generalize better, we modify the training data by delexicalizing it using a NER system. We replace the named entities in each training instance with their corresponding class names (PER, ORG, LOC, MISC). For each class, we

Input Text	Nalin works at CUNI located in Prague. Deepak is the best friend of Nalin.
RDF Triples	⟨Nalin | workplace | CUNI⟩, ⟨CUNI | location | Prague⟩, ⟨Deepak | bestFriend | Nalin⟩

Using an NER system we replace the named entities as

Input Text	[PER1] works at [ORG1] located in [LOC1]. [PER2] is the best friend of [PER1].
RDF Triples	⟨[PER1] | workplace | [ORG1]⟩, ⟨[ORG1] | location | [LOC1]⟩, ⟨[PER2] | bestFriend | [PER1]⟩

Figure 4.2: An example of modification of training data in Delexicalized Semantic Parsing

Model	Seen	Unseen	Whole
NER-Enc	**78.33**	44.22	63.38
NER-DataConc	78.2	**46.37**	**64.3**
NER-JointTrain	77.98	45.28	64.01

(a) Strict

Model	Seen	Unseen	Whole
NER-Enc	**78.62**	46.47	64.53
NER-DataConc	78.38	**48.23**	**65.21**
NER-JointTrain	78.12	47.96	65.03

(b) Exact

Model	Seen	Unseen	Whole
NER-Enc	**82.18**	54.21	69.92
NER-DataConc	81.87	**55.49**	**70.35**
NER-JointTrain	81.56	55.09	70.05

(c) Partial

Model	Seen	Unseen	Whole
NER-Enc	**85.36**	58.80	73.72
NER-DataConc	85.08	**60.26**	**74.24**
NER-JointTrain	84.82	60.01	73.68

(d) Type

Table 4.1: Evaluation scores for model variants of NER + Semantic Parsing

add an additional 20 tokens (e.g. PER1, PER2,..., PER20) to uniquely label all the named entities in a given verbalization. We then train a standard encoder-decoder model on the delexicalized parallel data. During decoding, we fill the corresponding slots with their respective values.

Figure 4.2 shows an example of the delexicalization of the training instance.

We use FLAIR Akbik et al. [2019] system (with 4 classes — ORG, LOC, PER, MISC) for the NER task in this and further experiments. We denote the system as *Delex*.

NER + Semantic Parsing

Given the significance of named entities in RDF parsing, we add NER as an auxiliary task. We combine NER along with the seq2seq semantic parsing task. We experiment with three variations of this combined approach, using an encoder-decoder system for all methods:

1. **Pretraining encoder for NER (NER-Enc)** We pretrain the system's encoder for NER task. We further fine-tune the model parameters on the semantic parsing task.

2. **Data concatenation (NER-DataConc)** We synthesize additional data for text-to-text NER task using the FLAIR NER system on the WebNLG

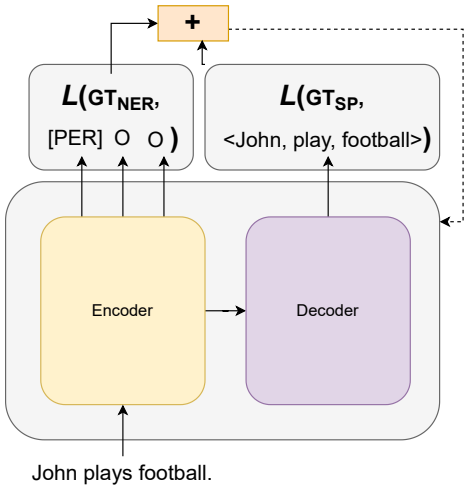

Figure 4.3: An example of joint training.

dataset. For each verbalization, we create a NER-tagged output using the NER system. We combine this dataset with the original WebNLG dataset and train the model on the shuffled training set. We use distinct prompts to distinguish between the tasks, namely "Text to RDF" and "Text to NER" for semantic parsing and NER respectively.

3. **Joint training for NER and Semantic Parsing (NER-JointTrain)** Since for each verbalization, we have corresponding NER tagged output (from the previous experiment) and a set of RDF triples, we simultaneously train the model for both the tasks. We use the model's encoder for NER tagging and the whole model for generating RDF triples (cf. Figure 4.3). We take the sum of both the losses during training.

Table 4.1 presents concise automatic evaluation scores given different model variants. *NER-Enc* performs slightly better in the seen categories. However, the performance deteriorates substantially for the unseen categories, making it worse than the other two on the whole dataset. Thus, we consider the *NER-DataConc* variant for our further comparison and manual evaluation.

Predicting Cardinality of RDF Triples set + Semantic Parsing

In our preliminary experiments, models struggle to parse longer inputs, specifically those with larger sets of target triples. To address this issue, we add an additional task of predicting the cardinality of the output triples set (i.e. no. of triples in output). We experiment with several variants of this configuration.

1. **Prepending cardinality to the outputs (Card-T2T)** We train the model to predict the cardinality right before predicting the RDF triples, in a text-to-text fashion. For each *En* verbalisation *[INPUT_TEXT]*, the corresponding output is formatted as "n | $RDFTRIPLE_1$ | ... | $RDFTRIPLE_n$", where n is the cardinality of set of output RDF triples.

Model	Seen	Unseen	Whole
Card-T2T	77.28	45.83	63.59
Card-DataConc	**78.09**	45.37	63.68
Card-JointEnc	77.96	46.60	**64.23**
Card-JointLog	77.13	**47.62**	64.22

(a) Strict

Model	Seen	Unseen	Whole
Card-T2T	77.58	47.91	64.67
Card-DataConc	**78.29**	47.32	64.64
Card-JointEnc	78.25	48.69	**65.30**
Card-JointLog	77.39	**49.51**	65.20

(b) Exact

Model	Seen	Unseen	Whole
Card-T2T	81.01	55.29	69.81
Card-DataConc	**81.81**	54.86	69.94
Card-JointEnc	**81.81**	56.04	**70.52**
Card-JointLog	80.97	**56.62**	70.32

(c) Partial

Model	Seen	Unseen	Whole
Card-T2T	84.08	59.73	73.48
Card-DataConc	**85.11**	59.75	73.94
Card-JointEnc	85.00	60.63	**74.33**
Card-JointLog	84.17	**61.07**	74.07

(d) Type

Table 4.2: Automatic evaluation scores for model variants of Cardinality Prediction + Semantic Parsing.

2. **Data Concatenation (Card-DataConc)** Similar to the previous NER-DataConc experiment, we train the model on a combined dataset for RDF parsing and cardinality prediction, treating cardinality prediction as a text-to-text task. For each training instance, we prepend either "Text to RDF" for the semantic parsing task or "Predict output cardinality" for the cardinality prediction task.

3. **Joint training with encoder (Card-JointEnc)** We use the model encoder to predict the number of triples while using the entire model for generating the triples. We use the encoder output corresponding to the ⟨BOS⟩ token to perform our classification task. We sum the losses from both sources and train the model accordingly.

4. **Joint training with logits (Card-JointLog)** Instead of using the encoder for classification, we add a classification layer to the decoder's last step logits for cardinality prediction. We calculate the corresponding loss and add it to the final loss.

The automatic evaluation scores for the variants are presented in Table 4.2. As *Card-JointEnc* demonstrates decent overall performance, we select this variant for further evaluation and comparison with other methods.

Triples Corruption + Semantic Parsing (TripCorr)

To improve the robustness of the model for unseen categories (cf. Section 3.2.1), we introduce additional noise to the output triples with a probability p_{corr}. Using the model's logits Peng et al. [2021], Kulhánek et al. [2021], we classify if the output contains noise. We mask the loss for corrupted outputs to prevent the model from learning to predict nonsensical triples.

We experiment with various levels of corruption: entity, triple, and triples set. At the entity level, we replace a random entity within a triple with another entity from the same class in a different triple (with probability p_{ren}). At the triple level,

Input Text: John plays football and eats pasta.
RDF Triple: ⟨John, play, football⟩, ⟨John, eat, pasta⟩

Prob.	Corruption Example (for ⟨John, play, football⟩)
p_{rem}	-
p_{rep}	⟨John, eat, pasta⟩
p_{perm}	⟨play, football, John⟩
p_{ren}	⟨John, eat, football⟩
p_a	⟨Sam, workplace, CUNI⟩

Figure 4.4: Triples corruption types with example. Here "Prob." refers to corruption probability.

Model	Seen	Unseen	Whole
Verb-MTL	**75.71**	**46.32**	**63.19**
Verb-TF	73.78	38.66	58.39
Verb-BT	33.98	28.19	31.50

(a) Strict

Model	Seen	Unseen	Whole
Verb-MTL	**75.86**	**48.15**	**64.01**
Verb-TF	73.52	40.12	59.89
Verb-BT	35.74	31.99	34.13

(b) Exact

Model	Seen	Unseen	Whole
Verb-MTL	**79.53**	**55.28**	**69.23**
Verb-TF	77.71	45.57	63.72
Verb-BT	40.53	38.19	39.53

(c) Partial

Model	Seen	Unseen	Whole
Verb-MTL	**82.95**	**60.03**	**73.04**
Verb-TF	81.93	48.96	66.91
Verb-BT	43.00	39.17	41.36

(d) Type

Table 4.3: Automatic Evaluation Scores for method variants with RDF Verbalization

we either permute the entities in the given triple (p_{perm}), replace the triple with another from the same set (p_{rep}), or remove the triple completely (p_{rem}). With probability p_a, we replace the whole output triples set with the set from other training instances (cf. Fig. 4.4).

We performed experiments with values of $p_{corr} = \{0.3, 0.4, 0.45\}$ and found out that $p_{corr} = 0.3$ works best. We set other hyperparameters values as p_{rem}=0.1, p_{rep}=0.3, p_{perm}=0.4, p_{ren}=0.2, p_a=0.5.

From our experiments on dev data, we found out that $p_{corr} = 0.3$ works best. Thus, we consider this variant for our further comparisons and denote the method by *TripCorr*.

RDF Verbalization + Semantic Parsing

Following the WebNLG 2020 challenge, several works in semantic parsing using verbalization in a multitask setup are proposed Agarwal et al. [2020], Guo et al. [2020c]. They gained a slight improvement in performance when compared to the semantic parsing task alone. Building upon their work, we introduce three variants of the method.

- **Verb-MTL** For our first variant, we simply add the dataset for the task of generating verbalization to the semantic parsing dataset, similar to the

Model	Seen	Unseen	Whole
Align-Alpha	72.25	44.92	60.5
Align-ShuffTrip	76.71	47.14	63.62
Align-ShuffSent	72.23	45.07	63.24
Align-AttnSc	**77.79**	**49.64**	**65.51**

(a) Strict

Model	Seen	Unseen	Whole
Align-Alpha	73.04	46.31	61.87
Align-ShuffTrip	76.81	49.71	64.88
Align-ShuffSent	77.40	47.87	64.38
Align-AttnSc	**78.60**	**51.59**	**66.25**

(b) Exact

Model	Seen	Unseen	Whole
Align-Alpha	75.69	53.24	66.91
Align-ShuffTrip	80.64	56.66	70.05
Align-ShuffSent	81.25	54.89	67.57
Align-AttnSc	**82.33**	**58.37**	**72.15**

(c) Partial

Model	Seen	Unseen	Whole
Align-Alpha	79.73	56.34	71.41
Align-ShuffTrip	83.93	60.30	73.49
Align-ShuffSent	80.32	59.14	71.75
Align-AttnSc	**85.12**	**62.36**	**74.86**

(d) Type

Table 4.4: Automatic Evaluation Scores for methods using align-then-generate

multitask setup (cf. Chapter 2.2.1). We distinguish between the tasks by using distinct prompts — "RDF to Text" for verbalization and "Text to RDF" for the semantic parsing task.

- **Verb-TF** Furthermore, we experiment with round-trip loss, viz., we use a vanilla encoder-decoder model to generate verbalizations using teacher forcing. We use this intermediate output to generate back the corresponding triples using the same model. We calculate the loss based on the predicted triples and train the model accordingly.

- **Verb-BT** We also experiment with another variant of this method. Instead of using teacher forcing to get the intermediate output, we let the model generate the triples autoregressively.

Based on the scores in Table 4.3, we consider *Verb-MTL* for further evaluation.

Align-then-generate

In our preliminary experiments, we noticed that the majority of the triples were not aligned with the corresponding verbalizations, i.e., the order of triples was different from the order of the corresponding entities in the verbalization. To check the effect of the ordering, we experiment with various methods.

1. **Alphabetical Order (Align-Alpha)** For each training instance, we sort the triples based on the first letters of the subject, object, and predicate.

2. **Shuffle triples (Align-ShuffTrip)** In this variant, we disrupt the sequence of triples by randomizing their order.

3. **Shuffle sentences (Align-ShuffSent)** We also explore shuffling the order of sentences within each training instance. While introducing noisy inputs can potentially improve generalization, it may also negatively impact scores due to the use of pronouns during inference.

Model	Seen	Unseen	Whole
baseline	77.43	45.07	63.24
SpanGen	27.60	10.95	20.53
Delex	42.86	30.25	37.38
NER-DataConc	**78.20**	46.37	64.3
Card-JointEnc	77.96	46.60	64.23
TripCorr	77.45	47.37	64.27
Verb-MTL	75.71	46.32	63.19
Align-AttnSc	77.79	**49.64**	**65.51**

(a) Strict

Model	Seen	Unseen	Whole
baseline	77.60	46.83	64.14
SpanGen	28.26	13.64	22.06
Delex	43.77	31.82	38.58
NER-DataConc	78.38	48.23	65.21
Card-JointEnc	78.25	48.69	65.30
TripCorr	77.63	49.05	65.10
Verb-MTL	75.86	48.15	64.01
Align-AttnSc	**78.6**	**51.59**	**66.25**

(b) Exact

Model	Seen	Unseen	Whole
baseline	81.04	54.28	69.31
SpanGen	30.50	16.30	24.48
Delex	48.34	37.49	43.63
NER-DataConc	81.87	55.49	70.35
Card-JointEnc	81.81	56.04	70.52
TripCorr	81.12	55.99	70.10
Verb-MTL	79.53	55.28	69.23
Align-AttnSc	**82.33**	**58.37**	**72.15**

(c) Partial

Model	Seen	Unseen	Whole
baseline	84.13	59.19	73.20
SpanGen	31.62	15.31	24.70
Delex	51.34	40.68	46.71
NER-DataConc	85.08	60.26	74.24
Card-JointEnc	85.00	60.63	74.33
TripCorr	84.36	60.67	73.98
Verb-MTL	82.95	60.03	73.04
Align-AttnSc	**85.12**	**62.36**	**74.86**

(d) Type

Table 4.5: Comparison of best variants from each method

4. **Aligning using Attention Scores (Align-AttnSc)** We modify the training data by ordering the target triples set $T = \{T_1, ..., T_n\}$ based on the cross-attention scores between them and the input verbalizations I, for each training instance. To order the triples, we first create a tuple $Tup_i = (Idx_{sub_i}, Idx_{obj_i}, Idx_{pred_i})$ corresponding to each triple T_i in the triples set T, where $Idx_{\{sub_i, obj_i, pred_i\}}$ are the indices to the tokens corresponding to relevant entity/information in the given verbalization. In order to map each entity in a given triple to the indices, we use cross-attention scores. We sum over all cross-attention heads of the first layer in the decoder. We then take softmax over the last dimension (i.e., embeddings). We consequently get a 2-dimensional matrix M corresponding to the verbalizations and triples token indices. We calculate $Idx_{ent_i} = \arg\max_k M(k, ent_i)$, where $ent_i \in \{sub_i, obj_i, pred_i\}$.

Based on the scores in Table 4.4, we consider *Align-AttnSc* for further evaluation.

4.1.3 Comparison of Proposed Methods

Table 4.5 presents a comparison of results among the top-performing variants across different methods. *Align-AttnSc* shows the best performance across all categories, except for the seen category in the "strict" evaluation type. *NER-DataConc* performs decently well in the seen category, even beating the *Align-AttnSc* method in the Strict evaluation type. The corruption-based approach (*TripCorr*) performs 2nd best on unseen categories. Method cardinality prediction *Card-JointEnc*) exhibits a slight advantage over the baseline, although

the improvements are not substantial. The inclusion of task-generation tasks in *Verb-MTL* appears to have decreased performance. Both *SpanGen* and *Delex* show poor performance, with the lowest scores. Given that the baseline system is a direct generation method, it performs quite well when compared to other methods with additional losses and tasks.

Qualitative Analysis

We conducted a qualitative analysis of generated triples from each method by randomly selecting 50 test instances for evaluation.

baseline The baseline method produces decent generations. However, it struggles with longer input verbalizations, generating incomplete triples in some cases.

SpanGen The span prediction method did not perform well. In many cases, predicted spans were empty, particularly for longer verbalizations. The generation task also struggled with shorter outputs and sometimes produced irrelevant predicates.

Delex This method performed slightly better than SpanGen. However, it faced challenges due to the complex mapping between named entities in verbalizations and output triples. The model frequently hallucinated and incorrectly generated NER placeholders (often as ORG), making accurate slot filling difficult.

NER-DataConc The outputs generated by this method were generally decent. However, the model sometimes propagated errors from the NER task. Consider the example "Input: It's Great to Be Young is a film edited by Max Benedict. — Gold Triples: ⟨It's_Great_to_Be_Young_(1956_film), editing, Max_Benedict⟩". When checked for the NER task, the model fails to recognize "It's" as a part of the film name "It's Great to Be Young". Consequently, the model generates ⟨Great_to_Be_Young_(1956_film), editing, Max_Benedict⟩ as output.

Card-JointEnc The predicted cardinality and length of generated triples generally aligned well for shorter verbalizations. However, as verbalization length increases, synchronization between the predicted cardinality and actual triples set is less consistent, with a difference of up to two triples. Also, cardinality prediction is not accurate in a few cases.

TripCorr Outputs from this method were comparable in quality to the baseline. Similar to other models, it struggled with longer triple sets. Although it does not reflect on the automatic scores, the order of generated triples was frequently different from the order of named entities in the input.

Verb-MTL Adding the task of RDF verbalization does not help the semantic parsing task. The performance, both in terms of qualitative and quantitative evaluation, remains similar to the baseline. However, when evaluated on the verbalization task, there is an improvement over direct fine-tuning on the target task.

Align-AttnSc The model performs better on longer verbalizations. We suspect that this performance gain is due to the well-structured and ordered targets. Consequently, it gets easier for the model to learn to parse the input texts more efficiently.

4.1.4 Conclusion

There is a slight improvement in the performances over several proposed methods over the baseline, especially in the unseen categories. However, even with the improvement, the strict match ultimately is fairly bad. The models tend to miss some parts of the entities, especially in the unseen categories. Furthermore, the models struggle with generating larger output triple sets.

4.2 Multilingual Semantic Parsing

In this section, we study and compare RDF triple-based semantic parsing across various languages, namely English (*En*), Russian (*Ru*), Maltese (*Mt*), Welsh (*Cy*), and Irish (*Ga*), in monolingual and multilingual settings. We describe the experimental configuration and the model variants in Section 4.2.1. We present and explain the results in Section 4.2.2, followed by conclusion in Section 4.2.3.

4.2.1 Experimental Settings

Data Preprocessing

Similar to our experiments in Section 4.1.1, we linearize the triples and prepend the class token (cf. Fig 4.1). We adjust the prompts relevant to each language. For instance, the prompt for RDF triple-based parsing for Welsh looks like "Text to RDF in the Welsh language".

Model Hyperparameters

We use mT5 base Xue et al. [2021] for our experiments. We train the model for 50 epochs with early stopping based on validation loss. We keep the batch sizes to 32 so that it fits the GPU memory and lr = 2e-05 (default). We select the best model based on the validation loss.

Model Variants

baseline We modify the WebNLG 2023 silver training data (Section 3.3.1) by retranslating the verbalizations using the SOTA translation system NLLB Costa-jussà et al. [2022]. We then fine-tune mT5 on the modified parallel data for each language.

Multilingual Modelling (MLM) We combine and shuffle the datasets for all the languages and fine-tune a single mT5-base.

Lang	Model	Seen	Unseen	Whole
En	Sec 4.1	**77.79**	**49.64**	**65.51**
	base	75.26	46.68	62.94
	MLM	66.42	29.56	50.40
Cy	base	**63.89**	**30.41**	**49.57**
	MLM	62.16	25.00	46.23
Ga	base	57.69	**24.92**	43.64
	MLM	**60.32**	23.70	**44.63**
Mt	base	**70.43**	**33.22**	**54.51**
	MLM	64.57	26.99	48.39
Ru	base	-	-	**85.85**
	MLM	-	-	75.09

(a) Strict

Lang	Model	Seen	Unseen	Whole
En	Sec 4.1	**78.6**	**51.59**	**66.25**
	base	75.37	47.62	63.40
	MLM	66.68	31.00	51.17
Cy	base	**64.25**	**32.15**	**50.52**
	MLM	62.39	26.62	47.06
Ga	base	58.02	**27.14**	44.78
	MLM	**60.64**	25.98	**45.79**
Mt	base	**70.66**	**35.26**	**55.52**
	MLM	64.70	28.77	49.23
Ru	base	-	-	**85.88**
	MLM	-	-	75.35

(b) Exact

Lang	Model	Seen	Unseen	Whole
En	Sec 4.1	**82.33**	**58.37**	**72.15**
	base	79.15	55.72	69.06
	MLM	71.62	38.56	57.25
Cy	base	**69.21**	**40.15**	**56.78**
	MLM	67.59	34.33	53.33
Ga	base	62.98	**35.49**	51.19
	MLM	**65.49**	33.20	**51.65**
Mt	base	**74.38**	**43.26**	**61.07**
	MLM	69.75	36.37	55.38
Ru	base	-	-	**87.90**
	MLM	-	-	78.81

(c) Partial

Lang	Model	Seen	Unseen	Whole
En	Sec 4.1	**85.12**	**62.36**	**74.86**
	base	82.79	62.50	74.05
	MLM	75.96	43.30	61.76
Cy	base	**73.36**	**44.88**	**61.18**
	MLM	72.05	38.62	57.72
Ga	base	67.08	**39.47**	55.24
	MLM	**69.53**	36.31	**55.3**
Mt	base	**77.66**	**47.66**	**64.83**
	MLM	74.30	40.72	59.84
Ru	base	-	-	**89.86**
	MLM	-	-	81.85

(d) Type

Table 4.6: Automatic evaluation scores of the methods across languages.

4.2.2 Results

We evaluate the *En* model on WebNLG 2020 dataset (Section 3.2.1), while other languages are evaluated on the latest WebNLG 2023 dataset (Section 3.3.1). Since the test data for *Ru* do not have any unseen category, we evaluate the *Ru* model only on the whole dataset. We also compare the *En* method with the best system from the previous section (*Align-AttnSc*). Even though *Align-Sc* is based on T5 small, it performs substantially better than the *En* baseline model (using mT5 base).

Table 4.6 presents the automatic evaluation scores for *En*, *Cy*, *Ga*, *Mt*, and *Ru*. For *Ga*, *MLM* performs substantially better than the baseline on seen categories. Furthermore, *MLM* has similar scores even for *Cy* on seen categories. However, *MLM* performs worse than the baseline for the other two languages (*En* and *Mt*). In fact, *MLM* has worse scores for unseen categories in every language we experimented with. Except for *Ga*, *base* has the best overall performance. We suspect this might be due to the poor quality of NLLB translations to *Ga*, which made the cross-lingual transfer from other languages more beneficial than

the standalone model.

4.2.3 Conclusion

The outputs are borderline usable based on automatic evaluation scores. However, based on the analysis using gold triples, there are cases where the models hallucinate with named entities. They often also struggle with numbers, along with incomplete triples in several cases. However, due to a limited understanding of languages other than *En*, we could not perform any qualitative analysis based on the input.

4.3 Takeaway

In this chapter, we proposed various approaches and methods for RDF triple-based semantic parsing. Initially, our experiments focused solely on the English language due to the availability of sufficient data. We employed several auxiliary tasks using different methodologies to aid the model in this task. Given that the parsing task involves identifying named entities, we added the NER task to complement the triple generation. While beneficial in some cases, incorrect NER predictions occasionally led to incomplete entity representations in output triples. A significant limitation across methods was incomplete parsing for longer texts. To address this, we introduced cardinality prediction as an additional task. While the model successfully predicted cardinality in many cases, it did not consistently improve the generation of the correct number of triples. We also experimented with corrupting outputs during training to enhance model robustness, yielding slight improvements for unseen categories. Furthermore, we explored incorporating the task of verbalization, however, the performance seemed to be similar to the baseline. The most effective method turned out to be a data-centric one, involving aligning output triples in the same order as the input text. Across all models, the improvements over the baseline were generally modest.

We also conducted experiments in a multilingual setting, training mT5 models individually for each language (base) and compared them with a model trained on all five languages collectively (MLM). Despite the storage efficiency of the combined model, its performance in generating triples was lower than the baseline in most cases. Including other languages in the training data, thus, did not enhance performance in any single target language other than *Ga*.

5. Data-to-Text Generation

In this chapter, we describe our various approaches proposed for RDF verbalization. Using the WebNLG 2023 dataset, we experiment with different methods in a multilingual setting (Section 5.1). As we took part in the WebNLG 2023 challenge, submitting systems for all the included languages, this chapter largely draws from the system description we submitted for the challenge Kumar et al. [2023][1]. Our approach leverages the mT5 model Xue et al. [2021], a multilingual transformer trained in a multi-task manner. Additionally, we examined and compared a recent parameter-efficient fine-tuning method, prompt tuning, with the traditional model fine-tuning (modifying all parameters) considering the training data size and the type of training strategy used (instruction tuned and pretrained; Section 5.2).

5.1 Multilingual Data-to-Text Generation

In this section, we propose various methods for the task of generating texts from RDF triples in the low-resource and multilingual setting (Section 5.1.1). To enhance performance for low-resource languages, we applied several additional fine-tuning and inference techniques: (1) We enhanced the WebNLG 2023 training data by re-translating it using a more robust machine translation (MT) model Costa-jussà et al. [2022]. (2) We fine-tuned either separate models for each language (single-task setting) or a unified model for all languages (multi-task setting). (3) We generated text by dividing the input RDF triples and decoding a subset of triples at a time. We describe the experimental settings (Section 5.1.2), present the evaluation results in Section ,5.1.3, and finally conclude the section in Section 5.1.4.

5.1.1 Approaches

We refine the training data for low-resource languages using better translation and filtering techniques. Furthermore, we investigate adding more tasks and languages during model training through multitask learning and address complex inputs by splitting them for decoding.

Better Translation & Data Filtering

To enhance the WebNLG 2023 baseline (Section 3.3.2) and demonstrate the impact of the quality of MT-processed training data on outputs in low-resource languages, we either replace or filter the baseline MT system outputs. These refined MT outputs are then used in an alternative generate-and-translate baseline and as improved training data for our direct NLG methods.

For *Mt*, *Ga*, and *Cy*, we substitute the Edinburgh Zero System Zhang et al. [2020a] with the SOTA NLLB system Costa-jussà et al. [2022]. Moreover, we revise our translation process. Originally, the training data was generated by

[1]My co-author, Saad Obaid Ul Islam, prepared the baselines, while I experimented with other methods and prepared the submitted systems

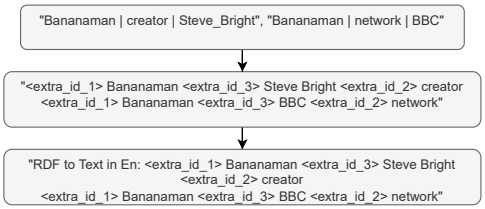

Figure 5.1: An instance of the preprocessing step

translating complete English (*En*) verbalizations. However, upon review, we discovered frequent length inconsistencies resulting from partial translations. To counter this, we now translate each *En* sentence separately using the NLLB system.

For *Br*, where NLLB is not available, we filter out inconsistent examples from the existing MT-processed training data. Each set of input triples in the training set usually corresponds to multiple verbalizations, so we calculate the length ratios of all *En* verbalizations to their corresponding *Br* translations. We then filter out the *Br* verbalizations with ratios (to their corresponding *En* originals) that are less than half the maximum ratio for the given input triple set.

Multitask Learning (MTL)

We utilize MTL (cf. Section 2.2.1) to enhance the model's comprehension of input triples. Besides generating text from data in the target language, we incorporate English translation and data-to-text generation in English as supplementary tasks. The model differentiates these tasks through distinct prompts.

Split and Generate (SaG)

Large input triple sets often present challenges in data quality, making it difficult for PLMs to produce coherent and accurate outputs. Previously, Narayan et al. [2017] proposed a split and rephrase method for simplifying sentences. Their approach involved creating parallel data by identifying verbalizations of a subset of input triples from the WebNLG dataset. Expanding on this idea and inspired by earlier work on sentence-by-sentence verbalization Moryossef et al. [2019], Ferreira et al. [2019], we adopt a simple decoding strategy: We partition the input triple sets into subsets based on their shared subject, generate outputs for each subset independently, and then concatenate the resulting texts. Additionally, we explore the effectiveness of limiting subset sizes by further partitioning subset S if its size exceeds a predefined value n.

5.1.2 Experimental Settings

Data preprocessing

In typical PLM-based RDF-to-text systems, we arrange the input triples in a linear format. To facilitate this, we introduce three new tokens within the model: one each for subjects, objects, and predicates (similar to Section 3.1.2). Additionally, we eliminate underscores from entities. The triples are concatenated in a subject-object-verb sequence, as illustrated in Figure 5.1. Following the

mT5 pretraining approach, we employ prompts to differentiate tasks. For RDF triple verbalization, our prompt format is structured as: "*RDF-to-text in* ⟨*lang*⟩*:* ⟨*input*⟩" where ⟨*lang*⟩ represents the target language and ⟨*input*⟩ denotes the input triples

Model Hyperparameters

We use the base variant of mT5 Xue et al. [2021].[2] The model loss starts plateauing at almost 20-25 epochs, thus, we train the models for 30 epochs with early stopping (n=5) based on validation loss. We use the batch size of 8, due to available GPU memory and model size. We use beam search decoding with beam size 4 (cf. Ch. 2.1.6). To prevent repetitive outputs, we also use a repetition penalty $rp = 3.5$ Keskar et al. [2019].[3]

Model Variants

We explore various system configurations based on approaches described in Section 5.1.1.

baseline For each language, we straightforwardly fine-tune mT5 using the provided training data provided by the organizers.

base-NLLB & **base-DF (data filtering)** For *Mt*, *Ga*, and *Cy*, we improve the training data by retranslating it using NLLB Costa-jussà et al. [2022]. For *Br*, we implement straightforward data filtering as outlined in Section 5.1.1. This step is not necessary for *Ru* due to the sufficient quality of its training data.

On dev data trials, we found out that the retranslated and filtered data outperforms the baseline, thus, consequently we adopt it as our primary training data for all subsequent experiments.

MTL We finetune separate mT5 models using MTL for *Mt*, *Ga*, and *Cy* (utilizing NLLB-retranslated data), and for *Ru* (using original data). This approach includes translation to English and English RDF-to-text as auxiliary tasks (see Section 5.1.1). For English translation, our prompt format is "*Translate from En to* ⟨*lang*⟩*:* ⟨*input*⟩", and the prompt for English text generation is "*RDF-to-text in English:* ⟨*input*⟩".

Multilingual Model (MLM) We finetune a single mT5 model on combined data from NLLB-retranslated *Mt*, *Ga*, and *Cy* datasets along with the existing *Ru* data. Since the quality of *Br* data is lower, it is excluded in this configuration.

SaG During inference, we optionally employ SaG to ensure simpler and more fluent outputs (see Section 5.1.1). We only apply the subset size limit n to *Br*, as our experiments indicate no performance benefits for other languages. Specifically, setting $n = 2$ yields optimal performance for *Br*.

[2] `https://huggingface.co/google/mt5-base`

[3] The value was found by development data trials.

Model	BLEU	METEOR	ChrF	TER
Kazakov et al. [2023]	54.71	-	0.69	0.37
Kasner and Dušek [2020]	52.90	-	0.68	0.40
Mille et al. [2019b] + GT	25.50	-	0.51	0.67
baseline	51.39	0.37	0.67	0.41
MTL	53.48	0.39	0.68	0.40
MLM ⋄	**54.52**	**0.39**	**0.69**	**0.38**
MTL + SaG	50.15	0.38	0.67	0.44
MLM + SaG	50.12	0.38	0.68	0.44

Table 5.1: Automatic evaluation scores for *Ru*. The English outputs of Mille et al. [2019b]'s system were processed by Google Translate (GT). Along with the scores of the best system in 2023, we also include the scores from the systems submitted in previous edition (2020) (see Table 5.3 for explanations). Ranking – 2nd of 4 submitted systems in the WebNLG 2023 Challenge.

Model	BLEU	MET	ChrF	TER
Guo et al. [2020b] + Edin	9.92	-	0.32	**0.77**
baseline	7.02	0.14	0.27	0.79
base-DF	8.84	0.15	0.29	0.91
base-DF + SaG ⋄	10.09	0.17	0.33	0.80
base-DF + SaG ($n = 2$)	**11.31**	**0.19**	**0.36**	0.83

Table 5.2: Automatic evaluation scores for *Br* (see Table 5.3 for explanations). Our submission was the only one competing in the WebNLG challenge 2023.

5.1.3 Results

We adopt the metrics used in the WebNLG 2023 Challenge (Section 3.3.1). Specifically, we use BLEU, METEOR, ChrF, and TER as assessment metrics for our NLG systems.

Table 5.3 presents a comprehensive overview of our experiments' evaluation results across *Ru*, *Mt*, *Ga*, and *Cy*. Notably, the *base-NLLB* configuration shows significant improvements over the *baseline* for *Mt*, *Ga*, and *Cy*. This enhancement is the most noteworthy since adopting NLLB represents the largest change in the setup.

In the case of *Ru* (Table 5.1), we observe improvements with the *MTL* and *MLM* setups, although the *SaG* decoding appears to degrade performance. Given that *Ru* is a language with ample resources, the model likely already handles complex outputs effectively in a single step, potentially losing context when splitting inputs into individual steps.

In contrast, for *Mt* (Table 5.3a), we witness a decline in performance with *MTL*/*MLM* compared to *base-NLLB*. Incorporating SaG provides marginal improvement but fails to match the results achieved by *base-NLLB*. The organizers' baseline and generate-and-translate with NLLB consistently outperform our models.

Model	BLEU	METEOR	ChrF	TER
Lorandi and Belz [2023]	21.27	-	0.52	0.65
Guo et al. [2020b] + Edin	15.60	-	0.42	**0.67**
Guo et al. [2020b] + NLLB	**16.07**	0.26	**0.47**	0.71
baseline	12.37	0.20	0.36	0.72
base-NLLB	14.08	0.25	0.44	0.77
MTL	13.95	0.25	0.44	0.78
MLM	13.91	0.25	0.45	0.77
MTL + SaG ⋄	14.02	0.26	0.45	0.78
MLM + SaG	14.02	**0.29**	0.45	0.79

(a) Maltese (*Mt*) – 3rd of 5 submitted systems in the WebNLG 2023 Challenge

Model	BLEU	METEOR	ChrF	TER
Lorandi and Belz [2023]	20.40	-	0.51	0.69
Guo et al. [2020b] + Edin	11.63	-	0.36	0.74
Guo et al. [2020b] + NLLB	**17.95**	0.23	**0.46**	**0.70**
baseline	6.53	0.13	0.27	0.77
base-NLLB	15.65	0.22	0.43	0.78
MTL	15.65	0.22	0.43	0.77
MLM	15.70	0.22	0.43	0.78
MTL + SaG ⋄	15.87	0.22	0.43	0.78
MLM + SaG	15.15	**0.32**	0.42	0.80

(b) Irish (*Ga*) – 3rd/4th of 5 submitted systems in the WebNLG 2023 Challenge

Model	BLEU	METEOR	ChrF	TER
Lorandi and Belz [2023]	25.11	-	0.55	0.64
Guo et al. [2020b] + Edin	10.70	-	0.36	0.77
Guo et al. [2020b] + NLLB	**18.77**	0.25	**0.48**	**0.70**
baseline	7.80	0.15	0.29	0.78
base-NLLB	16.13	0.24	0.44	0.79
MTL	16.73	0.24	0.45	0.78
MLM	16.65	0.24	0.44	0.80
MTL + SaG ⋄	17.01	0.24	0.45	0.79
MLM + SaG	16.28	**0.35**	0.45	0.81

(c) Welsh (*Cy*) – 3rd of 4 submitted systems in the WebNLG 2023 Challenge

Table 5.3: Automatic Evaluation Scores for *Mt*, *Ga*, *Cy*. We include the scores for the best WebNLG 2023 system Lorandi and Belz [2023] and baseline systems above the dividing line in each table. For *Mt*, *Ga*, *Cy*, the English outputs of Guo et al. [2020b]'s system were processed by Edinburgh Zero Zhang et al. [2020a] (Edin) and re-processed by ourselves using NLLB Costa-jussà et al. [2022], see Section 5.1.1. ⋄ denotes our system used for the challenge submission. The rankings of our submission and the number of submitted systems are included in each sub-caption.

Both *Ga* and *Cy* exhibit improved performance with *MTL* and *MTM* over *base-NLLB* (Tables 5.3b and 5.3c), similar to the trends observed in *Ru*. While there is a slight benefit from *SaG* decoding over *MTL* in these languages, it does not hold for the *MTM* case. Despite our direct translation models surpassing the challenge baseline, they fall short of the performance achieved by generate-and-translate using NLLB

Table 5.2 displays the scores for *Br*. Implementing *SaG* decoding alongside base-DF enhances performance over the organizers' baseline. Notably, unlike other languages, *Br* shows a significant performance boost with *SaG* decoding compared to methods without it.

5.1.4 Conclusion

Our submissions generally rank in the middle of the challenge leaderboard; notably, our submission for *Br* was the sole participant. For *Ru*, *Ga*, and *Cy*, our systems significantly outperform the baseline established by the organizers. However, other submissions were substantially better than us in all languages except *Br*. Notably, we performed relatively well in *Ru*, given that the best system (Kazakov et al. [2023]) utilizes a model with parameters almost three times of our model. For the other three languages, methods utilizing GPT3.5 and 4 won Lorandi and Belz [2023].

On human evaluation, we were almost on similar ranks. However, for *Ga*, unlike automatic evaluation metrics scores, the rule-based method Mille et al. [2023] performed worse than the method using generate (using T5) and translate (using NLLB) Hari et al. [2023]. The best system remained unchanged.

5.2 Is Tuning All Model Parameters Really Necessary?

In this section, we will evaluate the effectiveness of fine-tuning entire model weights versus using the prompt tuning approach to assess the capability of current models in simulated low-resource conditions (Section 2.1.5). Our experiments involve varying training sample sizes and examining how instruction tuned models perform relative to pretrained models. As we do not have a gold data in other low-resource languages, we base our experiments only on *En* language. We preprocess the triples similar to the experiments in previous section. Furthermore, we work only on direct generation setting.

We describe the model hyperparameters in Section 5.2.1. We present the automatic evaluation scores and manual analysis in Section 5.2.2 and finally conclude this study in Section 5.2.3

5.2.1 Experimental Settings

Model Hyperparameters

We utilize T5 models of varying sizes (small, base, and large) Raffel et al. [2020], alongside Flan T5 Chung et al. [2024] models of corresponding sizes, as our pretrained and instruction tuned setups respectively. We train the models for 10,000

# Training Samples	Finetuned				Prompttuned			
	BLEU	chrf	MET	TER	BLEU	chrf	MET	TER
0.5k	31.73	0.61	0.36	1.68	31.72	0.51	0.29	**0.94**
2.5k	35.15	0.62	0.37	1.56	34.21	**0.53**	**0.30**	1.13
10k	47.54	0.65	0.39	0.86	33.39	0.52	**0.30**	1.52
Whole	**49.78**	**0.67**	**0.40**	**0.63**	**34.34**	**0.53**	**0.30**	1.09

(a) T5 small

# Training Samples	Finetuned				Prompttuned			
	BLEU	chrf	MET	TER	BLEU	chrf	MET	TER
0.5k	39.30	0.59	0.35	**0.31**	39.78	0.60	0.35	0.86
2.5k	46.31	0.64	0.38	0.66	44.84	0.63	0.37	0.94
10k	50.55	0.67	0.40	0.51	**46.25**	**0.64**	**0.38**	**0.70**
Whole	**52.23**	**0.68**	**0.41**	0.35	45.06	**0.64**	0.37	0.98

(b) T5 base

# Training Samples	Finetuned				Prompttuned			
	BLEU	chrf	MET	TER	BLEU	chrf	MET	TER
0.5k	44.44	0.62	0.37	1.05	45.60	0.64	0.38	0.90
2.5k	47.97	0.66	0.39	0.78	47.66	0.65	0.39	0.78
10k	52.70	0.68	0.41	0.78	48.81	**0.67**	**0.40**	0.63
Whole	**53.41**	**0.69**	**0.42**	**0.35**	**49.29**	0.66	**0.40**	**0.51**

(c) T5 large

Table 5.4: fine-tuning vs Prompt tuning for pretrained model (T5). We use MET for METEOR score.

steps with early stopping (n=5) and a learning rate of 2e-05. For finding the number of training steps we gradually increased it from 5k to 10k. We also tried several more values of learning rate, however, the default value had the best performance on dev data. To investigate the impact of training sample sizes, we experiment with 0.5k, 2.5k, 10k, and the entire dataset.

For prompt tuning our models, we adopt the HuggingFace PEFT method with prompt token sizes set to 100 (we gradually increased it from 10 to 100 on dev data trials). As T5 is trained on implicit and shorter prompts, we use "RDF to Text in English language:" as prompt text for T5. On the other hand, Flan-T5 is trained on longer instructions and more explicit prompts, thus, we use "Generate fluent verbalisation for the following RDF Triples in English language:" for Flan-T5. Additionally, we set the learning rate to 0.3. We check several values of learning rate, and it turned out that the prompt tuning requires higher learning rate for our experiments.

# Training Samples	Finetuned				Prompttuned			
	BLEU	chrf	MET	TER	BLEU	chrf	MET	TER
0.5k	39.34	0.58	0.34	**0.47**	40.69	0.58	0.34	**0.39**
2.5k	40.07	0.60	**0.35**	0.90	41.98	0.58	0.34	0.70
10k	**40.64**	**0.61**	**0.35**	0.82	41.79	0.58	0.33	0.47
Whole	40.16	0.60	**0.35**	0.70	**42.36**	**0.60**	**0.35**	0.55

(a) Flan-T5 small

# Training Samples	Finetuned				Prompttuned			
	BLEU	chrf	MET	TER	BLEU	chrf	MET	TER
0.5k	42.27	0.62	0.36	0.82	49.59	0.66	0.39	0.59
2.5k	**48.50**	**0.66**	**0.39**	0.66	49.93	0.66	0.39	0.59
10k	45.66	0.65	0.38	**0.63**	49.97	0.66	0.39	0.63
Whole	45.54	0.65	0.38	**0.63**	**50.24**	0.66	0.39	**0.51**

(b) Flan-T5 base

# Training Samples	Finetuned				Prompttuned			
	BLEU	chrf	MET	TER	BLEU	chrf	MET	TER
0.5k	46.80	0.65	0.39	0.59	52.49	0.68	0.40	**0.74**
2.5k	**47.78**	0.65	0.39	0.74	52.51	0.68	0.40	0.78
10k	47.59	**0.66**	**0.40**	**0.51**	52.98	0.68	**0.41**	**0.74**
Whole	47.06	0.65	0.39	**0.51**	**53.54**	0.68	**0.41**	**0.74**

(c) Flan-T5 large

Table 5.5: fine-tuning vs Prompt tuning for instruction tuned model (Flan T5; [Chung et al., 2024])

5.2.2 Results

We evaluate the models on the WebNLG 2020 *En* test set (Section 3.2.1). Tables Tables 5.4 and 5.5 provide a thorough comparison between fine-tuning and prompt tuning across different configurations for T5 and Flan-T5, respectively.

As anticipated, the BLEU, chrF, and METEOR scores for the fine-tuned T5 model increase as the training sample size increases (see Table 5.4). However, for prompt tuned T5, the score improvements are less substantial with larger datasets starting from 2.5k samples onwards. This trend suggests potential underfitting, possibly due to the small increase in parameters introduced by prompt tuning. Notably, the training loss for T5-base-whole at the best step is significantly higher than validation loss ($0.42 > 0.28$), resulting in lower test scores compared to fine-tuning. Nonetheless, this difference in scores diminishes with larger model sizes.

Regarding Flan T5 (ref. Table 5.5), both prompt tuning and fine-tuning scores plateau or even decline in some instances as the training data size increases. This behavior led us to a detailed analysis of the published composition of Flan T5 training data. We found out that the model is already trained on the WebNLG dataset and several other similar datasets. Thus, the model just overfits when

Model Size	BLEU	chrf	MET	TER
small	48.27	0.47	0.27	0.54
base	50.49	0.64	0.38	0.43
large	54.29	0.69	0.41	0.40

Table 5.6: Scores for Zero Shot Prompting (Flan T5)

we further fine-tune it on the dataset (also evident from Table 5.6). Given that a substantial portion of the model parameters remains frozen during instruction tuning, prompt tuned Flan T5 consistently outperforms its fine-tuned counterpart.

Since Flan T5 is already trained on WebNLG and several other similar datasets, the zero-shot scores for Flan T5 is greater than both, finetuned and prompttuned Flan T5. Consequently, the scores for prompttuned Flan T5 are significantly higher than the T5 one.

Manual Analysis

We perform a small in-house manual evaluation on around 50 generated texts. The generated texts, while generally fluent, manage to incorporate nearly all of the input triples. The fine-tuned Flan T5 outputs still look fine, even though the score drops, but they struggle with some things, such as numbers, and hallucinate occasionally. Automatic evaluation metrics do not adequately capture the quality of these outputs. For instance, when calculated on individual sentences, the generated output with the lowest BLEU score observed for Flan T5 large prompt tuned on the entire dataset (~ 6.5) exhibits discrepancies where the reference is in active voice while the output itself is in passive voice. Moreover, there is a slight decline in automatic evaluation scores when generating text from a larger number of triples compared to fewer ones, despite maintaining faithfulness to the input triples. The smaller models occasionally exhibit instances of repetition and hallucination in their outputs, along with less comprehensive coverage of the input triples. However, these issues diminish with larger models.

5.2.3 Conclusion

Taking into account the conducted experiments, there is no definitive evidence that prompt tuning consistently outperforms fine-tuning for instruction-tuned models, especially when the model has already been pretrained on the WebNLG dataset. Overall, prompt tuning proves advantageous for larger models with smaller training datasets, along with being efficient in memory usage during training.

5.3 Takeaway

In this chapter, we explored several approaches for generating multilingual text from RDF triples in languages such as *Ru*, *Br*, *Mt*, *Ga*, and *Cy*. We described our

submitted systems for the WebNLG 2023 Challenge. The provided silver training dataset for less-resourced languages occasionally includes incomplete translations. To address this issue, we generate additional parallel data using NLLB, which proves to be highly effective in enhancing performance. Additionally, we achieve incremental improvements through multitask and multilingual training strategies, alongside a split-and-generate decoding technique that generates both simpler and more detailed outputs. While our systems demonstrated strong performance in languages with relatively larger amount of available resources, they faced challenges in generating text for extremely low-resource *Br* language.

We also experimented with generating texts using models finetuned on minimal training data. We introduced an additional adapter with a small number of parameters for prompt tuning. We hypothesized that along with having efficient memory usage while training, prompt tuning would have comparable performance in a low resource setting. Indeed, for smaller training samples, prompt tuning performed adequately and even outperformed fine-tuning in certain cases. However, as the training sample size increased, fine-tuning consistently performed better. Notably, larger models exhibited a narrower performance gap between the two methods. Looking ahead, it would be beneficial to explore the performance of even larger models. Additionally, it would make an interesting case to study the fine-tuning methods for languages other than *En*.

6. Conclusion

In this thesis, we provided a detailed overview of several proposed methods aimed at improving semantic parsing for English. We approach the task as a sequence-to-sequence generation in the majority of the experiments and base our methods on it. We experiment with different auxiliary tasks involving NER, triples set cardinality prediction, etc, and use T5 small as our fundamental model. The automatic evaluation scores for the methods are decent, making the methods usable in real cases. Our manual analysis identified limitations in our approaches and suggested strategies for addressing them. We also experiment with several methods for semantic parsing in a multilingual setting. The results are decent given that our proposed systems relied exclusively on WebNLG data.

We also experiment with various strategies for data-to-text generation. We showed that a straightforward way of splitting and decoding helps with Breton, an extremely low-resourced language. We also showed the efficacy of fine-tuning the models in multi-task/lingual settings. Furthermore, we discussed the applicability of the efficient prompt tuning method. We compared it with basic fine-tuning of the entire model. We also experimented with changing the models from pretrained (T5) to instruction-tuned (Flan T5). For T5 (pretrained model), fine-tuned one performed exceedingly well on a larger dataset. However, this performance gap decreased with an increase in model size. While the results do not give any evidence that the prompt tuning worsens/enhances the instruction-tuned model's performance, it definitely is easier and more compute-efficient to train.

Given that our models use only the WebNLG datasets, future research directions could involve integrating external data sources such as monolingual corpora of other languages. For example, pretraining models using noisy RDF triples generated from parts of speech tags and corresponding monolingual sentences could be beneficial. Additionally, we suspect using larger multilingual models might work better for both the tasks given they would have better zero-shot properties.

Bibliography

Oshin Agarwal, Mihir Kale, Heming Ge, Siamak Shakeri, and Rami Al-Rfou. Machine translation aided bilingual data-to-text generation and semantic parsing. In *Proceedings of the 3rd International Workshop on Natural Language Generation from the Semantic Web (WebNLG+)*, pages 125–130, Dublin, Ireland (Virtual), 12 2020. Association for Computational Linguistics. URL `https://aclanthology.org/2020.webnlg-1.13`.

Alan Akbik, Tanja Bergmann, Duncan Blythe, Kashif Rasul, Stefan Schweter, and Roland Vollgraf. FLAIR: An easy-to-use framework for state-of-the-art NLP. In *NAACL 2019, 2019 Annual Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pages 54–59, 2019.

Ebtesam Almazrouei, Hamza Alobeidli, Abdulaziz Alshamsi, Alessandro Cappelli, Ruxandra Cojocaru, Mérouane Debbah, Étienne Goffinet, Daniel Hesslow, Julien Launay, Quentin Malartic, et al. The falcon series of open language models. *arXiv preprint arXiv:2311.16867*, 2023.

Gabor Angeli and Christopher D Manning. Naturalli: Natural logic inference for common sense reasoning. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 534–545, 2014.

Dhananjay Ashok and Zachary C Lipton. Promptner: Prompting for named entity recognition. *arXiv preprint arXiv:2305.15444*, 2023.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

Hadi Banaee, Mobyen Uddin Ahmed, and Amy Loutfi. A framework for automatic text generation of trends in physiological time series data. In *2013 IEEE International Conference on Systems, Man, and Cybernetics*, pages 3876–3881. IEEE, 2013.

Satanjeev Banerjee and Alon Lavie. Meteor: An automatic metric for mt evaluation with improved correlation with human judgments. In *Proceedings of the acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*, pages 65–72, 2005.

Loïc Barrault, Magdalena Biesialska, Ondřej Bojar, Marta R. Costa-jussà, Christian Federmann, Yvette Graham, Roman Grundkiewicz, Barry Haddow, Matthias Huck, Eric Joanis, Tom Kocmi, Philipp Koehn, Chi-kiu Lo, Nikola Ljubešić, Christof Monz, Makoto Morishita, Masaaki Nagata, Toshiaki Nakazawa, Santanu Pal, Matt Post, and Marcos Zampieri. Findings of the 2020 conference on machine translation (WMT20). In *Proceedings of the Fifth Conference on Machine Translation*, pages 1–55, Online, November 2020. Association for Computational Linguistics. URL `https://www.aclweb.org/anthology/2020.wmt-1.1`.

Regina Barzilay and Noemie Elhadad. Inferring strategies for sentence ordering in multidocument news summarization. *Journal of Artificial Intelligence Research*, 17:35–55, 2002.

Regina Barzilay and Mirella Lapata. Modeling local coherence: An entity-based approach. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 141–148, Ann Arbor, Michigan, June 2005. Association for Computational Linguistics. doi: 10.3115/1219840.1219858. URL https://aclanthology.org/P05-1018.

Luisa Bentivogli, Arianna Bisazza, Mauro Cettolo, and Marcello Federico. Neural versus phrase-based machine translation quality: a case study. *arXiv preprint arXiv:1608.04631*, 2016.

Ondřej Bojar, Rajen Chatterjee, Christian Federmann, Yvette Graham, Barry Haddow, Matthias Huck, Antonio Jimeno Yepes, Philipp Koehn, Varvara Logacheva, Christof Monz, Matteo Negri, Aurélie Névéol, Mariana Neves, Martin Popel, Matt Post, Raphael Rubino, Carolina Scarton, Lucia Specia, Marco Turchi, Karin Verspoor, and Marcos Zampieri. Findings of the 2016 conference on machine translation. In *Proceedings of the First Conference on Machine Translation: Volume 2, Shared Task Papers*, pages 131–198, Berlin, Germany, August 2016. Association for Computational Linguistics. doi: 10.18653/v1/W16-2301. URL https://www.aclweb.org/anthology/W16-2301.

Danushka Bollegala, Naoaki Okazaki, and Mitsuru Ishizuka. A bottom-up approach to sentence ordering for multi-document summarization. *Information processing & management*, 46(1):89–109, 2010.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

Thiago Castro Ferreira, Iacer Calixto, Sander Wubben, and Emiel Krahmer. Linguistic realisation as machine translation: Comparing different MT models for AMR-to-text generation. In *Proceedings of the 10th International Conference on Natural Language Generation*, pages 1–10, Santiago de Compostela, Spain, September 2017. Association for Computational Linguistics. doi: 10.18653/v1/W17-3501. URL https://aclanthology.org/W17-3501.

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 Thiago Castro Ferreira, Claire Gardent, Nikolai Ilinykh, Chris van der Lee, Simon Mille, Diego Moussallem, and Anastasia Shimorina, editors, *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://aclanthology.org/2020.webnlg-1.7.

Ernie Chang, Xiaoyu Shen, Dawei Zhu, Vera Demberg, and Hui Su. Neural data-to-text generation with LM-based text augmentation. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 758–768, Online, April 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.eacl-main.64. URL https://aclanthology.org/2021.eacl-main.64.

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*, pages 128–135, 2008.

Yubo Chen, Yunqi Zhang, Changran Hu, and Yongfeng Huang. Jointly extracting explicit and implicit relational triples with reasoning pattern enhanced binary pointer network. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5694–5703, Online, June 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.naacl-main.453. URL https://aclanthology.org/2021.naacl-main.453.

Kyunghyun Cho, Bart van Merriënboer, 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. doi: 10.3115/v1/D14-1179. URL https://aclanthology.org/D14-1179.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, et al. Scaling instruction-finetuned language models. *Journal of Machine Learning Research*, 25(70):1–53, 2024.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. Unsupervised cross-lingual representation learning at scale. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8440–8451, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.747. URL https://aclanthology.org/2020.acl-main.747.

Marta R Costa-jussà, James Cross, Onur Çelebi, Maha Elbayad, Kenneth Heafield, Kevin Heffernan, Elahe Kalbassi, Janice Lam, Daniel Licht, Jean Maillard, et al. No language left behind: Scaling human-centered machine translation. *arXiv preprint arXiv:2207.04672*, 2022.

Liam Cripwell, Anya Belz, Claire Gardent, Albert Gatt, Claudia Borg, Marthese Borg, John Judge, Michela Lorandi, Anna Nikiforovskaya, and William Soto Martinez. The 2023 WebNLG shared task on low resource languages. overview and evaluation results (WebNLG 2023). In Albert Gatt, Claire Gardent, Liam Cripwell, Anya Belz, Claudia Borg, Aykut Erdem, and Erkut Erdem, editors, *Proceedings of the Workshop on Multimodal, Multilingual Natural*

Language Generation and Multilingual WebNLG Challenge (MM-NLG 2023), pages 55–66, Prague, Czech Republic, September 2023. Association for Computational Linguistics. URL https://aclanthology.org/2023.mmnlg-1.6.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1423. URL https://www.aclweb.org/anthology/N19-1423.

Li Dong and Mirella Lapata. Language to logical form with neural attention. *arXiv preprint arXiv:1601.01280*, 2016.

Pablo Ariel Duboue and Kathleen R. McKeown. Statistical acquisition of content selection rules for natural language generation. In *Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing*, pages 121–128, 2003. URL https://aclanthology.org/W03-1016.

Thiago Castro Ferreira, Chris van der Lee, Emiel van Miltenburg, and Emiel Krahmer. Neural data-to-text generation: A comparison between pipeline and end-to-end architectures. In *2019 Conference on Empirical Methods in Natural Language Processing (EMNLP) and 9th International Joint Conference on Natural Language Processing (IJCNLP)*, Hong Kong, November 2019. URL https://www.aclweb.org/anthology/D19-1052/. arXiv: 1908.09022.

Tsu-Jui Fu, Peng-Hsuan Li, and Wei-Yun Ma. GraphRel: Modeling text as relational graphs for joint entity and relation extraction. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1409–1418, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1136. URL https://aclanthology.org/P19-1136.

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 2017a. Association for Computational Linguistics. doi: 10.18653/v1/W17-3518. URL https://aclanthology.org/W17-3518.

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 2017b. Association for Computational Linguistics. doi: 10.18653/v1/P17-1017. URL https://www.aclweb.org/anthology/P17-1017.pdf.

Albert Gatt, Francois Portet, Ehud Reiter, Jim Hunter, Saad Mahamood, Wendy Moncur, and Somayajulu Sripada. From data to text in the neonatal intensive care unit: Using nlg technology for decision support and information management. *Ai Communications*, 22(3):153–186, 2009.

Eli Goldberg, Norbert Driedger, and Richard I Kittredge. Using natural-language processing to produce weather forecasts. *IEEE Expert*, 9(2):45–53, 1994.

Heng Gong, Xiaocheng Feng, and Bing Qin. Diffud2t: Empowering data-to-text generation with diffusion. *Electronics*, 12(9), 2023. ISSN 2079-9292. doi: 10.3390/electronics12092136. URL https://www.mdpi.com/2079-9292/12/9/2136.

Raghav Goyal, Marc Dymetman, and Eric Gaussier. Natural language generation through character-based RNNs with finite-state prior knowledge. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 1083–1092, Osaka, Japan, December 2016. The COLING 2016 Organizing Committee. URL https://aclanthology.org/C16-1103.

Qipeng Guo, Zhijing Jin, Ning Dai, Xipeng Qiu, Xiangyang Xue, David Wipf, and Zheng Zhang. $\sqrt{}^2$: A plan-and-pretrain approach for knowledge graph-to-text generation. In *Proceedings of the 3rd International Workshop on Natural Language Generation from the Semantic Web (WebNLG+)*, pages 100–106, Dublin, Ireland (Virtual), 12 2020a. Association for Computational Linguistics. URL https://aclanthology.org/2020.webnlg-1.10.

Qipeng Guo, Zhijing Jin, Ning Dai, Xipeng Qiu, Xiangyang Xue, David Wipf, and Zheng Zhang. P2: A Plan-and-Pretrain Approach for Knowledge Graph-to-Text Generation. In *Proceedings of the 3rd International Workshop on Natural Language Generation from the Semantic Web (WebNLG+)*, pages 100–106, Dublin, Ireland (Virtual), December 2020b. Association for Computational Linguistics. URL https://aclanthology.org/2020.webnlg-1.10.

Qipeng Guo, Zhijing Jin, Xipeng Qiu, Weinan Zhang, David Wipf, and Zheng Zhang. CycleGT: Unsupervised graph-to-text and text-to-graph generation via cycle training. In *Proceedings of the 3rd International Workshop on Natural Language Generation from the Semantic Web (WebNLG+)*, pages 77–88, Dublin, Ireland (Virtual), 12 2020c. Association for Computational Linguistics. URL https://aclanthology.org/2020.webnlg-1.8.

Sonal Gupta, Rushin Shah, Mrinal Mohit, Anuj Kumar, and Mike Lewis. Semantic parsing for task oriented dialog using hierarchical representations. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2787–2792, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1300. URL https://aclanthology.org/D18-1300.

Izzeddin Gur, Semih Yavuz, Yu Su, and Xifeng Yan. DialSQL: Dialogue based structured query generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1339–1349, Melbourne, Australia, July 2018. Association for Computational Linguistics. doi: 10.18653/v1/P18-1124. URL https://aclanthology.org/P18-1124.

Kancharla Aditya Hari, Bhavyajeet Singh, Anubhav Sharma, and Vasudeva Varma. Webnlg challenge 2023: Domain adaptive machine translation for low-resource multilingual rdf-to-text generation (webnlg 2023). In *Proceedings of the Workshop on Multimodal, Multilingual Natural Language Generation and Multilingual WebNLG Challenge (MM-NLG 2023)*, pages 93–94, 2023.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, nov 1997. ISSN 0899-7667. doi: 10.1162/neco.1997.9.8.1735. URL https://doi.org/10.1162/neco.1997.9.8.1735.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for nlp. In *International conference on machine learning*, pages 2790–2799. PMLR, 2019.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models, 2021. URL https://arxiv.org/abs/2106.09685.

Dirk Hüske-Kraus. Text generation in clinical medicine–a review. *Methods of information in medicine*, 42(01):51–60, 2003.

Glorianna Jagfeld, Sabrina Jenne, and Ngoc Thang Vu. Sequence-to-sequence models for data-to-text natural language generation: Word- vs. character-based processing and output diversity. In *Proceedings of the 11th International Conference on Natural Language Generation*, pages 221–232, Tilburg University, The Netherlands, November 2018. Association for Computational Linguistics. doi: 10.18653/v1/W18-6529. URL https://aclanthology.org/W18-6529.

Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Ye Jin Bang, Andrea Madotto, and Pascale Fung. Survey of hallucination in natural language generation. *ACM Computing Surveys*, 55(12):1–38, 2023.

Robin Jia and Percy Liang. Data recombination for neural semantic parsing. *arXiv preprint arXiv:1606.03622*, 2016.

Tim Johnson. Natural language computing: The commercial applications. *The Knowledge Engineering Review*, 1(3):11–23, 1984. doi: 10.1017/S0269888900000588.

Nal Kalchbrenner, Lasse Espeholt, Karen Simonyan, Aäron van den Oord, Alexander Graves, and Koray Kavukcuoglu. Neural machine translation in linear time. 2016. URL https://arxiv.org/abs/1610.10099.

Mihir Kale and Abhinav Rastogi. Text-to-text pre-training for data-to-text tasks. In *Proceedings of the 13th International Conference on Natural Language Generation*, pages 97–102, Dublin, Ireland, December 2020. Association for Computational Linguistics. URL https://aclanthology.org/2020.inlg-1.14.

Zdeněk Kasner and Ondřej Dušek. Data-to-text generation with iterative text editing. In *Proceedings of the 13th International Conference on Natural Language Generation*, pages 60–67, Dublin, Ireland, December 2020a. Association for Computational Linguistics. URL https://aclanthology.org/2020.inlg-1.9.

Zdeněk Kasner and Ondřej Dušek. Train hard, finetune easy: Multilingual denoising for RDF-to-text generation. In *Proceedings of the 3rd International Workshop on Natural Language Generation from the Semantic Web (WebNLG+)*, pages 171–176, Dublin, Ireland (Virtual), 12 2020b. Association for Computational Linguistics. URL https://aclanthology.org/2020.webnlg-1.20.

Zdeněk Kasner and Ondřej Dušek. Train Hard, Finetune Easy: Multilingual Denoising for RDF-to-Text Generation. In *Proceedings of the 3rd International Workshop on Natural Language Generation from the Semantic Web (WebNLG+)*, pages 171–176, Online, December 2020. URL https://aclanthology.org/2020.webnlg-1.20/.

Maxim Kazakov, Julia Preobrazhenskaya, Ivan Bulychev, and Aleksandr Shain. Webnlg-interno: Utilizing fred-t5 to address the rdf-to-text problem (webnlg 2023). In *Proceedings of the Workshop on Multimodal, Multilingual Natural Language Generation and Multilingual WebNLG Challenge (MM-NLG 2023)*, pages 67–72, 2023.

Nitish Shirish Keskar, Bryan McCann, Lav R Varshney, Caiming Xiong, and Richard Socher. Ctrl: A conditional transformer language model for controllable generation. *arXiv preprint arXiv:1909.05858*, 2019.

Ravi Kondadadi, Blake Howald, and Frank Schilder. A statistical NLG framework for aggregated planning and realization. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1406–1415, Sofia, Bulgaria, August 2013. Association for Computational Linguistics. URL https://aclanthology.org/P13-1138.

Taku Kudo and John Richardson. SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 66–71, Brussels, Belgium, November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-2012. URL https://aclanthology.org/D18-2012.

Jonáš Kulhánek, Vojtěch Hudeček, Tomáš Nekvinda, and Ondřej Dušek. AuGPT: Auxiliary tasks and data augmentation for end-to-end dialogue with pre-trained language models. In Alexandros Papangelis, Paweł Budzianowski, Bing Liu, Elnaz Nouri, Abhinav Rastogi, and Yun-Nung Chen, editors, *Proceedings of the 3rd Workshop on Natural Language Processing for Conversational AI*, pages 198–210, Online, November 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.nlp4convai-1.19. URL https://aclanthology.org/2021.nlp4convai-1.19.

Nalin Kumar, Saad Obaid Ul Islam, and Ondrej Dusek. Better translation + split and generate for multilingual RDF-to-text (WebNLG 2023). In Albert Gatt, Claire Gardent, Liam Cripwell, Anya Belz, Claudia Borg, Aykut Erdem, and Erkut Erdem, editors, *Proceedings of the Workshop on Multimodal, Multilingual Natural Language Generation and Multilingual WebNLG Challenge (MM-NLG 2023)*, pages 73–79, Prague, Czech Republic, September 2023. Association for Computational Linguistics. URL https://aclanthology.org/2023.mmnlg-1.8.

Tom Kwiatkowski, Luke Zettlemoyer, Sharon Goldwater, and Mark Steedman. Lexical generalization in ccg grammar induction for semantic parsing. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1512–1523, 2011.

Jason Lee, Kyunghyun Cho, and Thomas Hofmann. Fully character-level neural machine translation without explicit segmentation. *Transactions of the Association for Computational Linguistics*, 5:365–378, 2017. doi: 10.1162/tacl_a_00067. URL https://aclanthology.org/Q17-1026.

Oliver Lemon. Adaptive natural language generation in dialogue using reinforcement learning. *Proc. SEM-dial*, pages 141–148, 2008.

Leo Leppänen, Myriam Munezero, Mark Granroth-Wilding, and Hannu Toivonen. Data-driven news generation for automated journalism. In *Proceedings of the 10th International Conference on Natural Language Generation*, pages 188–197, Santiago de Compostela, Spain, September 2017. Association for Computational Linguistics. doi: 10.18653/v1/W17-3528. URL https://aclanthology.org/W17-3528.

Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning. In Marie-Francine Moens, Xuanjing Huang, Lucia Specia, and Scott Wen-tau Yih, editors, *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3045–3059, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-main.243. URL https://aclanthology.org/2021.emnlp-main.243.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*, 2019.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.703. URL https://aclanthology.org/2020.acl-main.703.

Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. In Chengqing Zong, Fei Xia, Wenjie Li, and Roberto Navigli, editors, *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4582–4597, Online, August 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-long.353. URL https://aclanthology.org/2021.acl-long.353.

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://aclanthology.org/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.

Xi Victoria Lin, Chenglong Wang, Luke Zettlemoyer, and Michael D. Ernst. NL2Bash: A corpus and semantic parser for natural language interface to the linux operating system. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan, May 2018. European Language Resources Association (ELRA). URL https://aclanthology.org/L18-1491.

Jie Liu, Shaowei Chen, Bingquan Wang, Jiaxin Zhang, Na Li, and Tong Xu. Attention as relation: Learning supervised multi-head self-attention for relation extraction. In Christian Bessiere, editor, *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, pages 3787–3793. International Joint Conferences on Artificial Intelligence Organization, 7 2020a. doi: 10.24963/ijcai.2020/524. URL https://doi.org/10.24963/ijcai.2020/524. Main track.

Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao. Multi-task deep neural networks for natural language understanding. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4487–4496, 2019.

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, 2020b. URL https://arxiv.org/abs/2001.08210.

Michela Lorandi and Anja Belz. Data-to-text generation for severely under-resourced languages with gpt-3.5: A bit of help needed from google translate (webnlg 2023). In *Proceedings of the Workshop on Multimodal, Multilingual Natural Language Generation and Multilingual WebNLG Challenge (MM-NLG 2023)*, pages 80–86, 2023.

Michela Lorandi and Anya Belz. High-quality data-to-text generation for severely under-resourced languages with out-of-the-box large language models. In Yvette Graham and Matthew Purver, editors, *Findings of the Association for Computational Linguistics: EACL 2024*, pages 1451–1461, St. Julian's, Malta, March 2024. Association for Computational Linguistics. URL https://aclanthology.org/2024.findings-eacl.98.

Minh-Thang Luong, Hieu Pham, and Christopher D Manning. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*, 2015.

Saad Mahamood and Ehud Reiter. Generating affective natural language for parents of neonatal infants. In *Proceedings of the 13th European Workshop on Natural Language Generation*, pages 12–21, 2011.

Christopher Manning and Hinrich Schutze. *Foundations of statistical natural language processing*. MIT press, 1999.

Christopher Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky. The Stanford CoreNLP natural language processing toolkit. In Kalina Bontcheva and Jingbo Zhu, editors, *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 55–60, Baltimore, Maryland, June 2014. Association for Computational Linguistics. doi: 10.3115/v1/P14-5010. URL https://aclanthology.org/P14-5010.

Susan W McRoy, Songsak Channarukul, and Syed S Ali. An augmented template-based approach to text realization. *Natural Language Engineering*, 9(4):381–420, 2003.

Simon Mille, Stamatia Dasiopoulou, Beatriz Fisas, and Leo Wanner. Teaching FORGe to verbalize DBpedia properties in Spanish. In Kees van Deemter, Chenghua Lin, and Hiroya Takamura, editors, *Proceedings of the 12th International Conference on Natural Language Generation*, pages 473–483, Tokyo, Japan, October–November 2019a. Association for Computational Linguistics. doi: 10.18653/v1/W19-8659. URL https://aclanthology.org/W19-8659.

Simon Mille, Stamatia Dasiopoulou, Beatriz Fisas, and Leo Wanner. Teaching FORGe to Verbalize DBpedia Properties in Spanish. In *Proceedings of the 12th International Conference on Natural Language Generation*, pages 473–483, Tokyo, Japan, October 2019b. Association for Computational Linguistics. doi: 10.18653/v1/W19-8659. URL https://aclanthology.org/W19-8659.

Simon Mille, Stamatia Dasiopoulou, and Leo Wanner. A portable grammar-based nlg system for verbalization of structured data. In *Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing*, SAC '19, page 1054–1056, New York, NY, USA, 2019c. Association for Computing Machinery. ISBN 9781450359337. doi: 10.1145/3297280.3297571. URL https://doi.org/10.1145/3297280.3297571.

Simon Mille, Elaine Uí Dhonnchadha, Stamatia Dasiopoulou, Lauren Cassidy, Brian Davis, and Anja Belz. Dcu/tcd-forge at webnlg'23: Irish rules!(wegnlg 2023). In *Proceedings of the Workshop on Multimodal, Multilingual Natural Language Generation and Multilingual WebNLG Challenge (MM-NLG 2023)*, pages 87–92, 2023.

Amit Moryossef, Yoav Goldberg, and Ido Dagan. Step-by-Step: Separating Planning from Realization in Neural Data-to-Text Generation. In *NAACL*, Minneapolis, MN, USA, June 2019. URL `https://www.aclweb.org/anthology/N19-1236/`. arXiv: 1904.03396.

Shashi Narayan, Claire Gardent, Shay B. Cohen, and Anastasia Shimorina. Split and rephrase. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 606–616, Copenhagen, Denmark, September 2017. Association for Computational Linguistics. doi: 10.18653/v1/D17-1064. URL `https://aclanthology.org/D17-1064`.

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 of the Association for Computational Linguistics*, pages 311–318, 2002.

Nivranshu Pasricha, Mihael Arcan, and Paul Buitelaar. NUIG-DSI at the WebNLG+ challenge: Leveraging transfer learning for RDF-to-text generation. In *Proceedings of the 3rd International Workshop on Natural Language Generation from the Semantic Web (WebNLG+)*, pages 137–143, Dublin, Ireland (Virtual), 12 2020. Association for Computational Linguistics. URL `https://aclanthology.org/2020.webnlg-1.15`.

Baolin Peng, Chunyuan Li, Jinchao Li, Shahin Shayandeh, Lars Liden, and Jianfeng Gao. Soloist: Building task bots at scale with transfer learning and machine teaching. *Transactions of the Association for Computational Linguistics*, 9:807–824, 2021. doi: 10.1162/tacl_a_00399. URL `https://aclanthology.org/2021.tacl-1.49`.

Maja Popović. chrF: character n-gram F-score for automatic MT evaluation. In Ondřej Bojar, Rajan Chatterjee, Christian Federmann, Barry Haddow, Chris Hokamp, Matthias Huck, Varvara Logacheva, and Pavel Pecina, editors, *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 392–395, Lisbon, Portugal, September 2015. Association for Computational Linguistics. doi: 10.18653/v1/W15-3049. URL `https://aclanthology.org/W15-3049`.

Ratish Puduppully, Li Dong, and Mirella Lapata. Data-to-text generation with content selection and planning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 6908–6915, 2019.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 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. *The Journal of Machine Learning Research*, 21(1):5485–5551, 2020.

Clément Rebuffel, Laure Soulier, Geoffrey Scoutheeten, and Patrick Gallinari. A hierarchical model for data-to-text generation, 2019.

Ehud Reiter. Nlg vs. templates. *arXiv preprint cmp-lg/9504013*, 1995.

Ehud Reiter and Robert Dale. Building applied natural language generation systems. *Natural Language Engineering*, 3(1):57–87, 1997. doi: 10.1017/S1351324997001502.

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

Verena Rieser and Oliver Lemon. Natural language generation as planning under uncertainty for spoken dialogue systems. In *Conference of the European Association for Computational Linguistics*, pages 105–120. Springer, 2009.

David E. Rumelhart and James L. McClelland. *Learning Internal Representations by Error Propagation*, pages 318–362. 1987.

Victor Sanh, Albert Webson, Colin Raffel, Stephen H Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, Teven Le Scao, Arun Raja, et al. Multitask prompted training enables zero-shot task generalization. In *ICLR 2022-Tenth International Conference on Learning Representations*, 2022.

Thibault Sellam, Dipanjan Das, and Ankur Parikh. BLEURT: Learning robust metrics for text generation. In Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel Tetreault, editors, *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7881–7892, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.704. URL `https://aclanthology.org/2020.acl-main.704`.

Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany, August 2016. Association for Computational Linguistics. doi: 10.18653/v1/P16-1162. URL `https://aclanthology.org/P16-1162`.

Rico Sennrich, Alexandra Birch, Anna Currey, Ulrich Germann, Barry Haddow, Kenneth Heafield, Antonio Valerio Miceli Barone, and Philip Williams. The University of Edinburgh's neural MT systems for WMT17. In Ondřej Bojar, Christian Buck, Rajen Chatterjee, Christian Federmann, Yvette Graham, Barry Haddow, Matthias Huck, Antonio Jimeno Yepes, Philipp Koehn, and Julia Kreutzer, editors, *Proceedings of the Second Conference on Machine*

Translation, pages 389–399, Copenhagen, Denmark, September 2017. Association for Computational Linguistics. doi: 10.18653/v1/W17-4739. URL https://aclanthology.org/W17-4739.

Anastasia Shimorina and Claire Gardent. Handling rare items in data-to-text generation. In *Proceedings of the 11th International Conference on Natural Language Generation*, pages 360–370, Tilburg University, The Netherlands, November 2018. Association for Computational Linguistics. doi: 10.18653/v1/W18-6543. URL https://aclanthology.org/W18-6543.

Matthew Snover, Bonnie Dorr, Rich Schwartz, Linnea Micciulla, and John Makhoul. A study of translation edit rate with targeted human annotation. In *Proceedings of the 7th Conference of the Association for Machine Translation in the Americas: Technical Papers*, pages 223–231, Cambridge, Massachusetts, USA, August 8-12 2006. Association for Machine Translation in the Americas. URL https://aclanthology.org/2006.amta-papers.25.

Kai Sun, Richong Zhang, Samuel Mensah, Yongyi Mao, and Xudong Liu. Recurrent interaction network for jointly extracting entities and classifying relations. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3722–3732, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.304. URL https://aclanthology.org/2020.emnlp-main.304.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112, 2014.

Lappoon R Tang and Raymond J Mooney. Using multiple clause constructors in inductive logic programming for semantic parsing. In *European Conference on Machine Learning*, pages 466–477. Springer, 2001.

Gemini Team, Rohan Anil, Sebastian Borgeaud, Yonghui Wu, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, et al. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*, 2023.

NLLB Team, Marta R. Costa-jussà, James Cross, Onur Çelebi, Maha Elbayad, Kenneth Heafield, Kevin Heffernan, Elahe Kalbassi, Janice Lam, Daniel Licht, Jean Maillard, Anna Sun, Skyler Wang, Guillaume Wenzek, Al Youngblood, Bapi Akula, Loic Barrault, Gabriel Mejia Gonzalez, Prangthip Hansanti, John Hoffman, Semarley Jarrett, Kaushik Ram Sadagopan, Dirk Rowe, Shannon Spruit, Chau Tran, Pierre Andrews, Necip Fazil Ayan, Shruti Bhosale, Sergey Edunov, Angela Fan, Cynthia Gao, Vedanuj Goswami, Francisco Guzmán, Philipp Koehn, Alexandre Mourachko, Christophe Ropers, Safiyyah Saleem, Holger Schwenk, and Jeff Wang. No language left behind: Scaling human-centered machine translation, 2022. URL https://arxiv.org/abs/2207.04672.

Mariët Theune, Esther Klabbers, Jan-Roelof de Pijper, Emiel Krahmer, and Jan Odijk. From data to speech: a general approach. *Natural Language Engi-*

neering, 7(1):47–86, 2001. URL http://journals.cambridge.org/action/displayAbstract?aid=73673.

Cynthia Thompson. Acquiring word-meaning mappings for natural language interfaces. *Journal of Artificial Intelligence Research*, 18:1–44, 2003.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

Bayu Distiawan Trisedya, Jianzhong Qi, Rui Zhang, and Wei Wang. GTR-LSTM: A triple encoder for sentence generation from RDF data. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1627–1637, Melbourne, Australia, July 2018. Association for Computational Linguistics. doi: 10.18653/v1/P18-1151. URL https://aclanthology.org/P18-1151.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *arXiv preprint arXiv:1706.03762*, 2017.

Shuhe Wang, Xiaofei Sun, Xiaoya Li, Rongbin Ouyang, Fei Wu, Tianwei Zhang, Jiwei Li, and Guoyin Wang. Gpt-ner: Named entity recognition via large language models. *arXiv preprint arXiv:2304.10428*, 2023a.

Yucheng Wang, Bowen Yu, Yueyang Zhang, Tingwen Liu, Hongsong Zhu, and Limin Sun. TPLinker: Single-stage joint extraction of entities and relations through token pair linking. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 1572–1582, Barcelona, Spain (Online), December 2020. International Committee on Computational Linguistics. doi: 10.18653/v1/2020.coling-main.138. URL https://aclanthology.org/2020.coling-main.138.

Zhuoer Wang, Marcus Collins, Nikhita Vedula, Simone Filice, Shervin Malmasi, and Oleg Rokhlenko. Faithful low-resource data-to-text generation through cycle training. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2847–2867, 2023b.

David HD Warren and Fernando CN Pereira. An efficient easily adaptable system for interpreting natural language queries. *American journal of computational linguistics*, 8(3-4):110–122, 1982.

Jason Wei, Maarten Bosma, Vincent Y Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M Dai, and Quoc V Le. Finetuned language models are zero-shot learners. *arXiv preprint arXiv:2109.01652*, 2021.

Tsung-Hsien Wen, Milica Gasic, Nikola Mrksic, Pei-Hao Su, David Vandyke, and Steve Young. Semantically conditioned lstm-based natural language generation for spoken dialogue systems. *arXiv preprint arXiv:1508.01745*, 2015.

Terry Winograd. Procedures as a representation for data in a computer program for understanding natural language. 1971.

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. doi: 10.18653/v1/D17-1239. URL https://aclanthology.org/D17-1239.

William A Woods. Progress in natural language understanding: an application to lunar geology. In *Proceedings of the June 4-8, 1973, national computer conference and exposition*, pages 441–450, 1973.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*, 2016.

Jiannan Xiang, Zhengzhong Liu, Yucheng Zhou, Eric Xing, and Zhiting Hu. Asdot: Any-shot data-to-text generation with pretrained language models. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 1886–1899, 2022.

Linting Xue, Noah Constant, Adam Roberts, Mihir Kale, Rami Al-Rfou, Aditya Siddhant, Aditya Barua, and Colin Raffel. mT5: A massively multilingual pre-trained text-to-text transformer. In Kristina Toutanova, Anna Rumshisky, Luke Zettlemoyer, Dilek Hakkani-Tur, Iz Beltagy, Steven Bethard, Ryan Cotterell, Tanmoy Chakraborty, and Yichao Zhou, editors, *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 483–498, Online, June 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.naacl-main.41. URL https://aclanthology.org/2021.naacl-main.41.

Zixiaofan Yang, Arash Einolghozati, Hakan Inan, Keith Diedrick, Angela Fan, Pinar Donmez, and Sonal Gupta. Improving text-to-text pre-trained models for the graph-to-text task. In *Proceedings of the 3rd International Workshop on Natural Language Generation from the Semantic Web (WebNLG+)*, pages 107–116, Dublin, Ireland (Virtual), 12 2020. Association for Computational Linguistics. URL https://aclanthology.org/2020.webnlg-1.11.

Yuan Yao, Deming Ye, Peng Li, Xu Han, Yankai Lin, Zhenghao Liu, Zhiyuan Liu, Lixin Huang, Jie Zhou, and Maosong Sun. Docred: A large-scale document-level relation extraction dataset. *arXiv preprint arXiv:1906.06127*, 2019.

Hongbin Ye, Ningyu Zhang, Shumin Deng, Mosha Chen, Chuanqi Tan, Fei Huang, and Huajun Chen. Contrastive triple extraction with generative transformer. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pages 14257–14265, 2021.

Pengcheng Yin and Graham Neubig. A syntactic neural model for general-purpose code generation. In *Proceedings of the 55th Annual Meeting of the Association*

for Computational Linguistics (Volume 1: Long Papers), pages 440–450, Vancouver, Canada, July 2017. Association for Computational Linguistics. doi: 10.18653/v1/P17-1041. URL https://aclanthology.org/P17-1041.

Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, et al. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task. *arXiv preprint arXiv:1809.08887*, 2018.

John M Zelle and Raymond J Mooney. Learning to parse database queries using inductive logic programming. In *Proceedings of the national conference on artificial intelligence*, pages 1050–1055, 1996.

Xiangrong Zeng, Daojian Zeng, Shizhu He, Kang Liu, and Jun Zhao. Extracting relational facts by an end-to-end neural model with copy mechanism. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 506–514, Melbourne, Australia, July 2018. Association for Computational Linguistics. doi: 10.18653/v1/P18-1047. URL https://aclanthology.org/P18-1047.

Luke S Zettlemoyer and Michael Collins. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. *arXiv preprint arXiv:1207.1420*, 2012.

Biao Zhang, Philip Williams, Ivan Titov, and Rico Sennrich. Improving massively multilingual neural machine translation and zero-shot translation. *arXiv preprint arXiv:2004.11867*, 2020a.

Jingqing Zhang, Yao Zhao, Mohammad Saleh, and Peter Liu. Pegasus: Pre-training with extracted gap-sentences for abstractive summarization. In *International Conference on Machine Learning*, pages 11328–11339. PMLR, 2020b.

Meishan Zhang, Bin Wang, Hao Fei, and Min Zhang. In-context learning for few-shot nested named entity recognition. In *ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 10026–10030. IEEE, 2024.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q Weinberger, and Yoav Artzi. Bertscore: Evaluating text generation with bert. *arXiv preprint arXiv:1904.09675*, 2019.

Chao Zhao, Marilyn Walker, and Snigdha Chaturvedi. Bridging the structural gap between encoding and decoding for data-to-text generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2481–2491, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.224. URL https://aclanthology.org/2020.acl-main.224.

Victor Zhong, Caiming Xiong, and Richard Socher. Seq2sql: Generating structured queries from natural language using reinforcement learning. *arXiv preprint arXiv:1709.00103*, 2017.

List of Figures

List of Tables

List of Abbreviations

A. Attachments

A.1 First Attachment