FACULTY
OF MATHEMATICS
AND PHYSICS
Charles University

DOCTORAL THESIS

Zdeněk Kasner

Data-to-Text Generation with Neural Language Models

Institute of Formal and Applied Linguistics

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

Study Program: Computational Linguistics

Prague 2024

Title: Data-to-Text Generation with Neural Language Models

Author: Zdeněk Kasner

Department: Institute of Formal and Applied Linguistics

Supervisor: Mgr. et Mgr. Ondřej Dušek, Ph.D.,
Institute of Formal and Applied Linguistics

Abstract:

Keywords: TODO

Název práce:	TODO
Autor:	Zdeněk Kasner
Katedra:	Ústav formální a aplikované lingvistiky
Vedoucí práce:	Mgr. et Mgr. Ondřej Dušek, Ph.D., Ústav formální a aplikované lingvistiky

Abstrakt:

TODO

Klíčová slova: TODO

Acknowledgements

TODO

This work has been using language resources and tools developed and/or stored and/or distributed by the LINDAT/CLARIN project of the Ministry of Education, Youth and Sports of the Czech Republic (project LM2015071).

Contents

1 Introduction

Producing *natural language* comes *natural* to us, humans. The key to computers' versatility and efficiency—their "language"—are data structures: arrays and lists, trees and graphs, tables and databases. Without appropriate tools, reading structured data is to most people like deciphering a foreign language. What is the best tool to undestand it? The problem lies not just in the unfamiliar format of such data, but in its scale. As the volume of structured data in our world is ever-growing, it becomes rather tempting to turn the question around: Can we instead make the computer translate the data to a language we already understand?

The attempts at generating natural language with a computer date back to the 1950s, when IBM researchers succesfully used a computer for translating between English and Russian (Sheridan, 1955). Shortly after, the work of Chomsky (1957) introduced formal grammar, providing a principled way for generating language with a set of rules. These initial successes stirred a lot of excitement; fully automated human-level language generation seemed within reach. In the 1960s, people slowly began to notice its difficulties: for example, Yngve (1961) notes there is "surprisingly wide linguistic diversity" when constructing grammar rules for the first ten sentences of a children's book. Still, the field of text generation gained momentum and descriptions of text generation systems started to appear (Woolley, 1969; Meehan, 1975; McDonald, 1975; Wang, 1980, *inter alia*). The report on the state of the art in text generation in 1982 predicted that within five years:

> *The resulting system can be expected to create acceptable, effective texts, limited by quality considerations to be about one page in length.*
>
> (Mann, 1982)

Fast forward to the present, and the research community is beaming with excitement again, this time about the unprecedented capabilities of neural language models (LMs) in generating fluent texts (Radford et al., 2019b; Brown et al., 2020). In the end, it took us over fifty years to build such systems. Similarly to other tasks in artificial intelligence (AI), from object recognition (Papert, 1966) to self-driving cars (Driverless Future, 2017), the apparent ease of the task for humans has proven deceptive. To achieve progress, we had to move away from linguistic theories and rule-based systems, re-defining our systems in terms of data-based approaches and generic learning algorithms.

Natural language generation (NLG) has meanwhile established itself as a standalone scientific discipline, with its journals, conferences, and stable base of researchers.[1] The research in the preceding decades was characterized by using a varied assortment of tools including grammars, formalisms, linguistic theories, and custom components. Combining these tools was understood as necessary for building text generation systems (Mann, 1982; Reiter and Dale, 1997). As a result, many systems from that time—from chart captioning systems (Mittal et al., 1998) and graph descriptors (Sun and Mellish, 2006), to weather forecast systems (Belz, 2008b) and healthcare report generators (Portet et al., 2009b)—were accurate and reliable, but domain-specific and rigid.

With neural models, natural language processing (NLP) as a research field, along with NLG as one of its subfields, has changed dramatically (Gururaja et al., 2023; Li et al., 2023). Most notably, these fields have become more experimental. While neural language models (LMs) opened up fascinating possibilities in building end-to-end systems and solving the long-standing issues with fluency and domain-independence (Ferreira et al., 2019; Dušek et al., 2020; Sharma et al., 2022), working with neural models turned out to be closer to behavioral sciences than engineering (Holtzman et al., 2023). As the researchers began to "throw" neural LMs at all sorts of problems, the issues concerning experimental design and evaluation came to the surface (Gehrmann et al., 2022). Due to this, some researchers perceived the change as questionable at the very least (Reiter, 2020; Gururaja et al., 2023; Michael et al., 2023). The shift towards experimental approaches has also created a gap between research and industry; the industry opted for established approaches meeting industrial standards instead of trying new research artifacts (Dale, 2020, 2023).

Nevertheless, the progressive approach adopted by NLP over the past few years turned out to have its merits. The general emphasis on openness, inherited from the machine learning (ML) field—where publicly releasing papers, code, and models has become commonplace—has allowed everybody to stand on the proverbial shoulders

[1] See the history of SIGGEN meetings: `https://aclanthology.org/sigs/siggen/`.

of giants. Thanks to open-science initiatives such as arXiv.org[2] or HuggingFace[3], research became more accessible to both researchers and the general public. The convergence towards generic approaches has also led to heavy cross-pollination of ideas, making specific solutions easily applicable to other tasks. As such, NLG is helping to advance other areas of NLP and contribute to general knowledge of the natural language, its production and processing.

Finally, as we gained ways to generate language that do not require starting from structured representations (summarize and paraphrase texts, continue text segments, generate stories and answers to questions, or describe images and videos), the original field concerned with generating descriptions of structured data has adopted the—perhaps more apt—name of *data-to-text (D2T) generation*.

This thesis tells a story about how data-to-text generation and neural language models came together. On the way, it touches various facets of D2T generation: from improving generation in a low-resource setting (Chapter 3), over evaluating generated texts (Chapter 4), processing and visualizing data (Chapter 5), to interpreting system behavior (Chapter 6). The thesis inevitably reflects the shifts in NLP between 2020 and 2024: from the preliminary attempts at generating fluent language with small pretrained LMs, all the way up to dealing with the hype surrounding the large language models (LLMs). The approaches presented in this thesis are primarily motivated by the idea that adopting neural models in D2T may help us solve some long-standing issues with flexibility and text fluency, which were out of reach for the best approaches from previous decades.

1.1 Motivation

The main goal of the thesis is to close the gap outlined in the introduction: turning experimental approaches into reliable and accurate D2T generation systems. As a premise, we consider neural LMs[4] as a useful tool of producing fluent and natural-sounding text. However, we do not take neural LMs as a one-size-fits-all solution. Instead, we carefully study how to integrate LMs in D2T systems while following the strict demands on fluency, controllability, and semantic accuracy of the output.

[2]`https://arxiv.org/`

[3]`https://huggingface.co/`

[4]For brevity, we will commonly use "LMs" to denote "neural LMs" throughout this work unless stated otherwise.

The side goal of the thesis is then to *understand*: understand the data we are dealing with, the outputs we can reasonably expect, and the behavior of neural-based systems in certain conditions. D2T generation has several specifics that make it a good subject for this kind of study: its resource scarcity (due to which there are still questions that cannot be answered by scaling up the models), the tension between the established rule-based and new-coming neural approaches, and the fact that the specific format and size of the data makes it less suitable for end-to-end solutions.

To make the goals more tangible, we split them into the following research questions, which we address further on in the thesis:

RQ1 **In which scenarios are LMs useful for D2T generation?** First, it is crucial to identify the strong sides of LMs and get an intuition of where the models can make the most impact. How far can we get with LM-only baselines? And are there outcomes that we can get with LMs that are better than previous approaches?

RQ2 **How to efficiently process the structured data with LMs?** With structured data, we need to deal with the fact that LMs were pre-trained on modeling plain text only. To efficiently leverage the knowledge in LMs—especially in low-resource settings—we need to find the way to transform the data into a suitable input format while keeping its structure (along with other information in the data) intact.

RQ3 **How to make LM-based systems more controllable?** A neural component introduced in the D2T generation system will inevitably make the system less controllable. The question is if we can minimize these issues by building systems out of smaller and simpler components, training the models for more predictable tasks, or producing intermediate outputs that can be manually examined.

RQ4 **How to evaluate the outputs of D2T generation systems?** Evaluating generated text gets harder as the quality of the texts starts to approach the human level. Since human evaluation is costly and time-consuming, we study how to build automatic metrics that can be used for system development and evaluation. Particularly, we focus on the most pressing issue in D2T generation: semantic accuracy of the generated texts with respect to the input data.

RQ5 **Do D2T generation systems generalize to unseen domains and datasets?** D2T generation systems are often evaluated on a limited subset of domains and datasets. Investigating how the models perform on unseen domains, multiple datasets, and real-world data would give us better picture of the limitations of the current approaches.

1.2 Main Contributions

The following are our main contributions, following the research questions outlined above:

Ad RQ1 We show that with a very **simple LM-based finetuned baseline**, we can achieve strong results on a shared task of generating texts from a knowledge graph (Section 3.1). We also point out the advantages and limitations of open LLMs on D2T generation in zero-shot settings (Section 6.2).

Ad RQ2 We show how to **transform the data to intermediate text-like input** suitable for LMs using handcrafted or automatically extracted templates (Sections 3.2, 3.3, and 4.1), rule-based NLG methods (Section 4.2), and specialized LMs (Section 6.1). We show that these methods can serve as a basis both for competitive neural-based D2T generation systems and for novel LM-based evaluation metrics.

Ad RQ3 We show how we can limit LMs to the task of improving text fluency and use these LMs for building **more controllable D2T generation systems** with an iterative approach (Section 3.2) and modular architecture (Section 3.3). We show that these systems open up a new way of thinking about neural-based LM with a different set of trade-offs than rule-based or end-to-end systems.

Ad RQ4 We develop **LM-based automatic metrics** for evaluating outputs of D2T generation systems on the level of data item mentions (Section 4.1) and output tokens (Section 4.2). We show that the metrics achieve strong correlations with human judgment in comparison with other metrics.

Ad RQ5 We **unify the format** of multiple D2T generation datasets for easier processing and visualization (Section 5.1). Using novel datasets, we also **evaluate the output quality and semantic accuracy** of LMs across multiple D2T tasks, data formats, and domains (Sections 6.1 and 6.2).

1.3 Thesis Overview

The thesis is organized into the background chapter (Chapter 2), the content chapters (Chapters 3 to 6), and the concluding chapter (Chapter 7).

Sec.	Topic	Publication
Chapter 3: Low-Resource Data-to-Text Generation		
§3.1	D2T generation with a finetuned LM	Kasner and Dušek (2020b)
§3.2	D2T generation with an editing LM	Kasner and Dušek (2020a)
§3.3	D2T generation with a pipeline of LMs	Kasner and Dušek (2022)
Chapter 4: Evaluating Semantic Accuracy		
§4.1	Evaluating D2T with natural language inference	Dušek and Kasner (2020)
§4.2	Evaluating token-level accuracy of complex D2T	Kasner et al. (2021)
Chapter 5: Unified Data Processing		
§5.1	TabGenie toolkit for D2T datasets	Kasner et al. (2023a)
Chapter 6: Examining Model Behavior		
§6.1	Describing unseen triples in a knowledge graph	Kasner et al. (2023b)
§6.2	D2T generation across domains with open LLMs	Kasner and Dušek (2024)

Table 1.1: Overview of the thesis.

The Chapters 3 to 6, which describe our contributions, are outlined in Table 1.1. First, we describe our work on improving D2T generation in low-resource scenarios in Chapter 3. We continue with our work on evaluating the semantic accuracy of D2T generation in Chapter 4. In Chapter 5, we describe TabGenie, our toolkit for processing and visualization of D2T generation datasets. Finally, in Chapter 6, we present our experiments with cross-domain performance of pretrained and large LMs on D2T generation.

Publications The thesis includes the content of eight publications written by the author of the thesis. Except for the paper Dušek and Kasner (2020), where the experimental part was done by the author's supervisor, the author of the thesis was the main author of all the publications and executed major part of the work.[5] All the publications were (or are to be) published at top-tier NLP conferences ACL, EACL and INLG.

[5]The contributions for publications with multiple authors are detailed in the respective chapters.

2 Background

This chapter explains the basic concepts used throughout the thesis.

In Section 2.1, we first cover *neural language models (LMs)*. We start with a brief theory of neural networks and text representation for neural networks. This theoretical grounding will help us to define the task of language modeling and its connection to neural networks. We then look at specific neural architectures, particularly the transformer architecture, and show how pretraining models based on this architecture can produce models with strong natural language processing (NLP) capabilities.

In Section 2.2, we turn our attention to data-to-text (D2T) generation, the central task explored in this thesis. To motivate the task, we start with an overview of real-world D2T applications. We also explain the subtasks into which D2T generation can be decomposed. We show how various approaches tackle these subtasks, starting from early rule-based approaches to recent neural-based systems. Finally, we describe D2T datasets and evaluation metrics, focusing on the ones relevant to this thesis.

We assume that the typical reader of this thesis comes from a related NLP area and is familiar with the relevant concepts. That being said, the thesis can also be used by a beginner in the field, as the work is self-contained and provides pointers to relevant work for more details. The chapter serves as the main point of reference; we will only briefly revisit the most relevant concepts in the respective chapters.

2.1 Neural Language Models

In this section, we work our way towards neural LMs. We start with the mathematical foundations of neural networks (NNs) on which the neural LMs are built (Section 2.1.1), the ways we can represent text in neural networks (Section 2.1.2), and the basic ideas of language modeling (Section 2.1.3). Equipped with the necessary theoretical background, we then introduce the transformer architecture (Section 2.1.4) and how it serves as a basis for pretrained (Section 2.1.5) and large (Section 2.1.6) language models.

2.1.1 Neural Networks

Neural networks are a tool for learning patterns from data.[1] Such a tool will help us with learning language patterns from large-scale textual data, and in turn with generating text.

To begin, let's say our goal is to predict real-number output $y \in \mathbb{R}$ for the given vector of real numbers $\mathbf{x} = (x_1, \dots, x_d) \in \mathbb{R}^d$. We assume that the $\mathbf{x} \to y$ mapping is not arbitrary—that would leave us with memorizing all the $(\mathbf{x}, y)$ pairs—but follows some regularities and underlying patterns that can be learned. This assumption is naturally satisfied if we consider $(\mathbf{x}, y)$ to be representations of real-world data, e.g., documents and their labels.

We use mathematical models designed to capture the patterns in their parameters to learn the mapping. The models estimate the parameters from a set of $(\mathbf{x}, y)$ examples called the *training data* and use the parameters to predict the outputs on a separate set of examples (generally assumed to come from the same data-generating distribution) called the *test data.*

Perceptron Algorithm One of the early mathematical models designed for learning patterns from data is the *perceptron algorithm* (Rosenblatt, 1958, Bishop, 2006, p. 192). For the perceptron algorithm, we need to restrict the output to a binary class label: $y \in \{-1, 1\}$. The algorithm learns the parameters $\mathbf{w} = (w_1, \dots, w_d) \in \mathbb{R}^d$ and the bias $b \in \mathbb{R}$ describing a linear decision boundary separating the data points according to their class label. The algorithm proceeds as follows:

(1) The parameters $\mathbf{w}$ and b are initialized to small random values (or zeros).

[1] Until we get to D2T generation in Section 2.2, we use the word "*data*" only in its abstract sense, as in "any inputs we can apply our algorithms to". We use the term "structured data" whenever it is necessary to make the distinction.

(2) For each training example $(\mathbf{x}_i, y_i)$, the algorithm updates the weights and bias to adjust their current estimate $\hat{y}_i$ towards the ground truth target y_i:

$$\hat{y}_i = \text{sign}(\mathbf{w} \cdot \mathbf{x} + b) \tag{2.1}$$

$$\mathbf{w} = \mathbf{w} + (y - \hat{y})\mathbf{x} \tag{2.2}$$

$$b = b + y - \hat{y} \tag{2.3}$$

(3) The step (2) is repeated until convergence.

The perceptron algorithm is guaranteed to converge if a hyperplane exists that separates the data belonging to one class from another; otherwise, it does not converge (Novikoff, 1962).

Multi-layer Perceptron To overcome the fact that the perceptron is limited to linear decision boundaries, we can use a *multi-layer perceptron* (MLP; Goodfellow et al., 2016, p. 164). This mathematical model—also known as a *feed-forward neural network*—can approximate any bounded continuous function (Hornik et al., 1989).

As the name suggests, MLP processes the input with multiple perceptron-like units called *neurons*. Analogically to the perceptron (Equation 2.1), each neuron computes its output o using the rule:

$$o = f(\mathbf{x}^\top \mathbf{w} + b), \tag{2.4}$$

where $f : \mathbb{R} \to \mathbb{R}$ is the *activation function*, $\mathbf{x} \in \mathbb{R}^n$ is the input vector, and $\mathbf{w} \in \mathbb{R}^n$ and $b \in \mathbb{R}$ are learnable parameters. Instead of signum, MLP uses differentiable non-linear functions, nowadays most commonly the rectified linear unit (ReLU; Nair and Hinton, 2010), where $f(x) = \max(0, x)$, or its variants (Hendrycks and Gimpel, 2016; Dubey et al., 2022).

For efficiency, the neurons are organized in layers, which enables formulating MLP computations in terms of matrix multiplication. The i-th layer of MLP is parametrized with a matrix $\mathbf{W}_i \in \mathbb{R}^{d \times n}$ and a vector of biases $\mathbf{b}_i \in \mathbb{R}^n$, processing the intermediate output called the *hidden state* $\mathbf{h}_{i-1} \in \mathbb{R}^d$ (where we set $\mathbf{h}_0 = \mathbf{x}$):

$$\mathbf{h}_i = f(\mathbf{h}_{i-1}\mathbf{W}_i + \mathbf{b}_i). \tag{2.5}$$

To estimate the parameters of the network, we need to *train* the network using the training data. For each training example $(\mathbf{x}, y)$, we first do a *feed-forward pass*: we feed $\mathbf{x}$ into the network and use the current parameters of the network to get the prediction $\hat{y}$. We then update the parameters to minimize the gap between the predicted output $\hat{y}$ and the ground truth output y. This gap is described by a *loss* *function* $\mathcal{L}(y, \hat{y}) \to \mathbb{R}$. Since all the computations in MLP are differentiable, we can compute exactly how much each parameter contributes to the loss function using the chain rule for derivatives. The derivative for each parameter—called a *gradient* when grouped in a vector—directly influences the size of the update. The process of computing and applying the updates is called a *backward pass* (or backpropagation; Kelley, 1960; Rumelhart et al., 1986) and operates in the reverse order of layers. The magnitude of the updates is controlled by the *learning rate* $\alpha \in \mathbb{R}$.

One of the basic optimizers (i.e., the algorithms for updating the parameters) is the stochastic gradient descent (SGD; Goodfellow et al., 2016, p. 275). SGD estimates the gradient in each step using a limited number of examples called a *batch* and directly updates the parameters in the direction of the gradient. More advanced optimizers, such as Adam (Kingma and Ba, 2014), also adapt the learning rate for each parameter based on the history of the gradients, helping to make the convergence more robust.

Recurrent Neural Networks Unlike with MLPs, where we the size of the input is fixed, recurrent neural networks (RNNs) allow us to process a sequence of inputs $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_n)$ of arbitrary length. An RNN computes a sequence of hidden states $\mathbf{H} = (\mathbf{h}^{(0)}, \ldots, \mathbf{h}^{(n)})$, i.e., one state for each input in the sequence. $\mathbf{H}$—or sometimes more specifically, the last hidden state $\mathbf{h}^{(n)}$—is the encoded representation of the input sequence, which can be in turn used in downstream tasks (e.g., for tagging or classifying the sequence).

The RNN computes the hidden states $\mathbf{H}$ by repeatedly applying a function f parametrized by the parameters of the network $\boldsymbol{\theta}$, the current input $\mathbf{x}_i$, and the previous hidden state $\mathbf{h}^{(i-1)}$ (Goodfellow et al., 2016, p. 367):

$$\mathbf{h}^{(i)} = f(\boldsymbol{\theta}, \mathbf{h}^{(i-1)}, \mathbf{x}_i), \tag{2.6}$$

where the first hidden state $\mathbf{h}^{(0)} \in \mathbb{R}^k$ is initialized randomly. The function f is generally implemented as a series of matrix multiplications and non-linear functions. The exact implementation of f can get more complex with advanced RNN architectures such as LSTM (Hochreiter and Schmidhuber, 1997) or GRU (Cho et al., 2014).

RNNs had various successes across NLP on sequence processing tasks (Karpathy, 2015; Salehinejad et al., 2017). At the same time, RNNs turned out to have various shortcomings, such as the vanishing gradient problem, the fixed size of the hidden state, and limited possibilities of parallelization; all of which made it hard to model long-term dependencies and train the network efficiently (Hochreiter, 1998; Pascanu et al., 2013). We return to RNNs mainly in Section 2.1.4, showing how it became a predecessor to the transformer architecture.

2.1.2 Text Representation

In this section, we look into how we can represent text in neural networks.

One-Hot Encoding A text is a sequence of discrete units such as characters or words. To convert these units—called *tokens*—to a numerical representation, we first enumerate a set of all possible tokens (a *vocabulary* V) and assign each token an integer index $i \in \{0, \dots, |V| - 1\}$.

The naive way to represent each token would be using its integer value. However, this would misleadingly suggest linear dependence between tokens. A better way is to use the index i for constructing a *one-hot* vector $\mathbf{x} \in \{0, 1\}^{|V|}$ for each token:

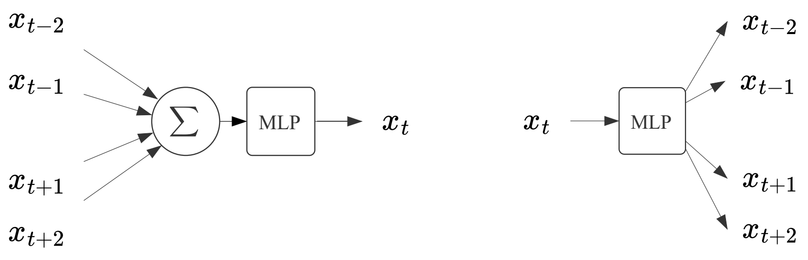
$$x_j = \begin{cases} 1 & \text{if } i = j, \\ 0 & \text{otherwise.} \end{cases} \tag{2.7}$$

While this representation is more sound, it is quite sparse and does not capture the semantics of individual tokens, which puts high requirements on the representational capacity of the network.

Word Embeddings A more useful representation of tokens is the one in which the tokens with similar meanings are also close in the vector space. To get this representation, we can build on the distributional theory of meaning, according to which the meaning of a token is defined by a context in which it occurs (Harris, 1954; Firth, 1957).

In the context of neural networks, this idea is used in the Word2Vec algorithm (Mikolov et al., 2013). The algorithm trains an MLP for a specific objective and uses its weigths as token representations. The MLP training objective (illustrated in Figure 2.1) either consists of predicting the tokens in the neighborhood of each token (the *skip-gram* objective) or vice versa, the token itself based on the tokens in its neighborhood (the *continuous bag-of-words* objective). The output of the algorithm is an *embedding matrix* $\mathbf{W}_e \in \mathbb{R}^{|V| \times d}$, which assigns each token a d-dimensional *embedding vector* $\mathbf{x} \in \mathbb{R}^d$.

The notion of using an embedding matrix for representing tokens is used also in neural LMs (Section 2.1.3). However, in majority of neural LMs, the embedding matrix is not trained separately with a specific algorithm, but jointly with the rest of the network.



(a) Skip-gram (b) Continuous Bag-of-Words

Figure 2.1: The objectives employed by the Word2Vec algorithm (Mikolov et al., 2013). The algorithm here uses a context window of size $k = 5$. In the (a) *skip-gram* algorithm, we sum the embeddings of $k - 1$ surrounding tokens and predict the original token. In the (b) *continuous bag-of-words* algorithm, we use the original token for predicting the $k - 1$ surrounding tokens.

Tokenization Until now, we have implicitly assumed that we have a way of tokenizing the text, i.e. splitting it into discrete units. The most straightforward way is to tokenize the text into words or characters. However, these approaches have major shortcomings. Word-level tokenization considers morphologically similar words as independent units, forcing the model to learn their representation separately. It also becomes more difficult for languages such as Chinese, which do not separate words with spaces. In contrast, character-level tokenization is computationally inefficient and gives us less meaningful units.

Subword tokenization is the middle ground between the word-level and character-level tokenization. It splits the text into smaller pieces called *subwords*, which are continous character spans of various length. With subword tokenization, frequently used words typically get their own subword while less frequent words are split into multiple subwords.

The subword tokenization algorithm that is commonly used in neural LMs is *byte-pair encoding* (BPE; Sennrich et al., 2016). The BPE algorithm starts with the vocabulary of individual bytes, iteratively merging the most frequent tokens and adding them to the vocabulary V until we reach the target vocabulary size. An example subword tokenization of the expression "Subword tokenization" could be the subwords `['Sub', 'word', '_token', 'ization']`, where "`_`" is a special character denoting a preceding space.

2.1.3 Language Modeling

After establishing the theory of neural networks and showing how to represent text in neural networks, we are ready to explain the notion of *language modeling*, a central concept for building neural LMs.

Language Model A *language model* is a mathematical model that estimates a probability of a sequence of tokens $X = (x_1, \ldots, x_n)$. To estimate the probability, we can factorize the sequence probability using the chain rule:

$$P(X) = \prod_{i=1}^{n} P(x_i|x_1, \ldots, x_{i-1}). \tag{2.8}$$

This formulation gives us a way to compute the probability of the whole sequence as the product of probabilities of individual sequence prefixes.

***n*-gram Language Model** An *n*-gram LM (parametrized by a positive integer n) further simplifies the product using the assumption that the probability of a token depends only on $N - 1$ previous tokens (Jurafsky and Martin, 2024, p.32):

$$P(X) = \prod_{i=1}^{T} P(x_i|x_{i-N+1}, \ldots, x_{i-1}). \tag{2.9}$$

The probabilities for individual *n*-grams are estimated by counting their occurrences over a text corpus, storing the counts in a look-up table. The main limitation of n-gram LMs (besides the size of the look-up tables) is the limit on the length of the context for each token, due to which *n*-gram LMs fail to capture long-term dependencies.

Neural Language Model A neural LM is a language model that estimates the text probability $P_\theta(X)$ using a neural network with parameters θ. In contrast with *n*-gram LMs, neural LMs can capture long-term dependencies and efficiently store the probability distribution in their parameters.

The parameters of the neural LM are also estimated using a text corpus. For each word x_i in the corpus, we aim to maximize the conditional probability that the model assigns to this word: $P_\theta(x_i|x_{<i})$. If we express the gap between the model distribution P_θ and the empirical distribution of sequences in the corpus P using cross-entropy, this formulation is equal to minimizing the negative log-likelihood of the next word (Jurafsky and Martin, 2024, p.158):

$$\mathcal{L}_i = -\log P_\theta(x_i|x_{<i}). \tag{2.10}$$

This type of training is also called *self-supervised*: each token in the corpus serves as the ground-truth label that the model aims to predict. We discuss the training objectives for specific model architectures in Section 2.1.5.

2.1.4 Transformer Architecture

In this section, we pave the way towards the *transformer* (Vaswani et al., 2017): the core neural architecture used in NLP nowadays. We describe its individual components and how the transformer is used for text processing.

Encoder-Decoder Framework We have described the RNN (Section 2.1.1) as a neural network that can *encode* an input sequence into hidden states. The encoder-decoder framework (Sutskever et al., 2014; Cho et al., 2014) allows us to also *generate an output sequence*. The idea is to use another network called the *decoder* for generating the sequence, using the last hidden state of the encoder as its initial state. Here, we illustrate how the framework is applied using two RNNs:[2]

(1) The RNN called the *encoder* encodes the sentence of input embeddings $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_n)$ into a sequence of hidden states $\mathbf{H}_e = \{\mathbf{h}_e^{(0)}, \ldots, \mathbf{h}_e^{(n)}\}$ (where $\mathbf{h}_e^{(0)}$ is a null vector) by repeatedly applying a transformation $\mathcal{E}$ in each timestep $i \in (1, n)$:

$$\mathbf{h}_e^{(i)} = \mathcal{E}(\mathbf{h}_e^{(i-1)}, \mathbf{x}_i). \tag{2.11}$$

(2) The RNN called the *decoder* uses $\mathbf{h}_e^{(n)}$ as its initial state $\mathbf{h}_d^{(0)}$ and produces the sequence of output tokens $Y = (y_1, \ldots, y_m)$ by repeatedly applying a transformation $\mathcal{D}$ in each timestep $j \in (1, m)$:

$$\mathbf{h}_d^{(j)}, y_j = \mathcal{D}(\mathbf{h}_d^{(j-1)}, y_{j-1}). \tag{2.12}$$

Note that the decoder produces the output sequence iteratively, yielding a token y_j in each timestep. This process is called *autoregressive decoding* and is described in more detail in Section 2.1.5.

Attention Mechanism We have mentioned that the hidden state of an RNN has a fixed size, which limits the amount of information the network can capture about a sequence. The *attention mechanism* (Bahdanau et al., 2015) enables the decoder to extract the information dynamically from the encoded sequence. In each step j, the decoder computes a context vector c_j as the weighted sum of the hidden states of the

[2]Later adapting the general idea also for the transformer architecture.

encoder $\mathbf{H}_e$ using the attention matrix $\mathbf{W}_a$:

$$\alpha_{ji} = \text{softmax}(\mathbf{h}_d^{(j)} \mathbf{W}_a \mathbf{h}_e^{(i)}), \tag{2.13}$$

$$\mathbf{c}_j = \sum_i \alpha_{ji} \mathbf{h}_e^{(i)}. \tag{2.14}$$

The context vector is used as an additional input for the decoder:

$$\mathbf{h}_d^{(j)}, y_j = \mathcal{D}(\mathbf{h}_d^{(j-1)}, y_{j-1}, \mathbf{c}_j). \tag{2.15}$$

Self-attention Mechanism Self-attention (Cheng et al., 2016; Vaswani et al., 2017) is a variant of the attention mechanism in which the source and the target sequence are identical. Given the input $\mathbf{X} \in \mathbb{R}^{n,d}$, the *self-attention* produces the output $\mathbf{H} \in \mathbb{R}^{n,d}$ of the same size. For each token, the resulting vector $\mathbf{h}_i \in \mathbf{H}$ is a weighted combination of the value vectors corresponding to all the tokens in a sequence (including the token itself):

$$\mathbf{h}_j = \sum_{i \in 1..n} \alpha_{ji} \mathbf{v}_i, \tag{2.16}$$

where the *value vector* of each token is computed using a trainable *value matrix* $\mathbf{W_v} \in \mathbb{R}^{n,d}$:

$$\mathbf{v}_i = \mathbf{x}_i \mathbf{W_v}. \tag{2.17}$$

To get the attention weights α_{ij}, we first compute *query* and *key* vectors for each token using trainable matrices $\mathbf{W_q}$ and $\mathbf{W_k} \in \mathbb{R}^{n,d}$. Each weight is a normalized dot product of the corresponding vectors:

$$\mathbf{q}_i = \mathbf{x}_i \mathbf{W_q} \tag{2.18}$$

$$\mathbf{k}_i = \mathbf{x}_i \mathbf{W_k} \tag{2.19}$$

$$\alpha_{ij} = \text{softmax}\left(\frac{\mathbf{q}_i \mathbf{k}_j}{\sqrt{d}}\right). \tag{2.20}$$

The operations can be efficiently parallelized using matrix multiplication:

$$\mathbf{Q} = \mathbf{X}\mathbf{W_q}, \quad \mathbf{K} = \mathbf{X}\mathbf{W_k}, \quad \mathbf{V} = \mathbf{X}\mathbf{W_v} \tag{2.21}$$

$$\text{Attn}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d}}\right)\mathbf{V}. \tag{2.22}$$

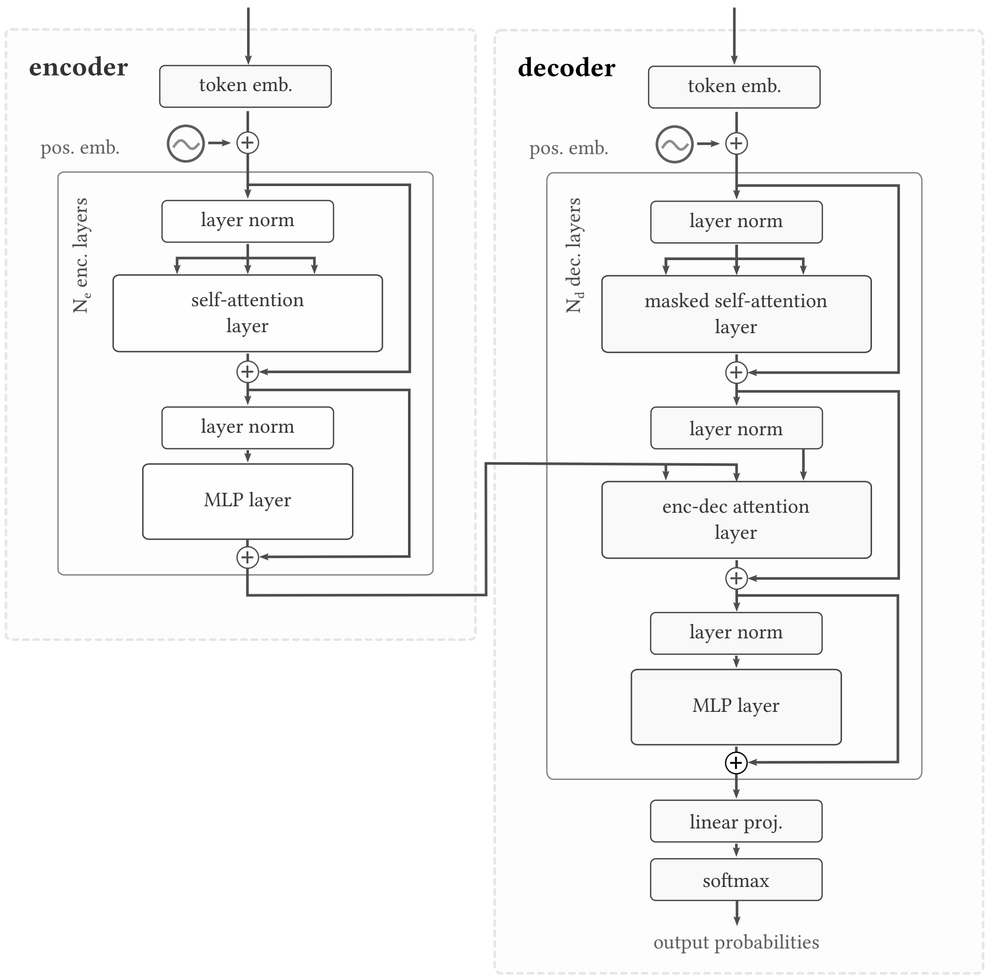


Figure 2.2: An encoder-decoder variant of the Transformer architecture. The encoder has N_e layers, each consisting of a self-attention and MLP sublayer. The decoder has N_d layers with masked self-attention and encoder-decoder attention, again followed by an MLP sublayer. The input to each sublayer is normalized using layer norm. After the last decoder layer, the output probabilities are computed using a linear projection and softmax. The figure is adapted from https://github.com/bbycroft/llm-viz.

Transformer Architecture The *transformer*[3] (Vaswani et al., 2017) is a neural network architecture that can process sequences efficiently in parallel. To achieve that, the transformer replaces the RNN hidden state (previously used for sharing information among tokens in a single sequence) with the self-attention mechanism applied over a series of layers.

[3]Although Vaswani et al. (2017) use "Transformer" with a capital "T", the orthography is gradually shifting towards the variant with a lower-case "t". See, e.g., Jurafsky and Martin (2024, p. 215).

Specifically, each layer is composed of two sublayers: (a) the *self-attention layer* and (b) the *MLP layer*. The output of the i-th sublayer is added to its original input $\mathbf{H}^{(i)}$ in a *residual connection*:

$$\mathbf{H}^{(i+1)} = \mathbf{H}^{(i)} + \text{sublayer}(\mathbf{H}^{(i)}). \tag{2.23}$$

The sublayers serve a different purpose: while the MLP layer computes element-wise operations over each token, the self-attention layer enables sharing information among tokens. The sublayer input (or output, depending on the architecture variant) is normalized using *layer normalization* (Ba et al., 2016).

To get the input representation $\mathbf{H}^{(0)}$, we sum the token embeddings $\mathbf{X} \in \mathbb{R}^{n,d}$ with *positional embeddings*. Positional embeddings encode the absolute or relative position information of individual tokens, which needs to be represented explicitly in parallel processing.[4]

As shown in Figure 2.2, the original Transformer architecture is based on the encoder-decoder framework. The decoder layers implement two additional features:

- Each layer contains an additional sublayer called the *encoder-decoder attention*. In contrast to the self-attention mechanism, the *keys* and *values* come from the last layer of the encoder, enabling the decoder to attend to the encoded sequence.
- The self-attention is *masked* so that each token can collect information only from the preceding tokens, which is necessary to enable training the model for left-to-right autoregressive decoding.

After the last decoder layer, the hidden states are projected into a matrix of size $\mathbb{R}^{|V| \times n}$ and normalized using softmax, producing a probability distribution over the vocabulary for each input token.

2.1.5 Pretrained Language Models

To achieve good performance on downstream tasks, we typically need to spend extensive time for training the transformer on large-scale training data. A more efficient workflow is as follows: the models are first *pretrained* using a generic objective on large-scale data—such as The Pile (Gao et al., 2020), or C4 (Raffel et al., 2019)—and then *finetuned* for downstream tasks on a smaller, task-specific dataset. Although pretraining a model still requires significant computational resources, pretrained language models (PLMs) can be then distributed for further finetuning, which is in itself more efficient.

[4]There are multiple variants of positional embeddings with various trade-offs; see Dufter et al. (2022) for an overview.

Type	Example Models	# Parameters	Note
Encoder	BERT (Devlin et al., 2019)	110M-340M	landmark pretrained encoder
	RoBERTa (Liu et al., 2019b)	125M-355M	improves BERT pretraining
	LaserTagger (Malmi et al., 2019)	110M	text-editing model
Enc-Dec	BART (Lewis et al., 2019)	139M-406M	landmark encoder-decoders
	T5 (Raffel et al., 2019)	220M-11B	
	mBART (Liu et al., 2020)	680M	multilingual version of BART
Decoder	GPT-2 (Radford et al., 2019a)	117M-1.5B	landmark pretrained decoder
	Llama2 (Touvron et al., 2023)		
	Mistral (Jiang et al., 2023)	7B-70B	open large language models
	Zephyr (Tunstall et al., 2023)		

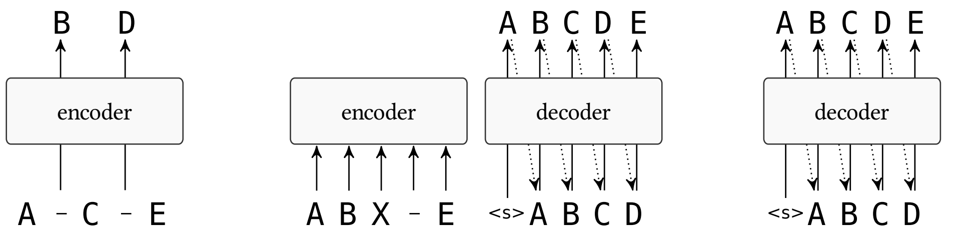
Table 2.1: Types of transformer architectures and specific models used in this work. The number of parameters may vary based on the model variant.

Model types Depending on the downstream task, different variants of the transformer architecture are used (see Table 2.1 for an overview):

- **Encoder models** use only the *encoder* part of the transformer architecture. These models are not generative; instead, they produce a contextualized representation of the input sequence $\mathbf{X}$. The representation can be used for downstream tasks such as sequence classification, sequence tagging, or computing sequence similarity.
- **Encoder-decoder models** use the original *encoder-decoder* architecture, and are explicitly trained to transform an input sequence $\mathbf{X}$ into a target sequence $\mathbf{Y}$. Encoder-decoder models are mostly used for sequence-to-sequence tasks, such as machine translation (MT), question answering, or summarization.
- **Decoder models** use only the *decoder* part of the transformer architecture, which makes them suitable for generating text continuations. While seemingly less expressive, the models can be used for the same tasks as the encoder-decoder models, using the input sequence $\mathbf{X}$ as the prefix for generating the output sequence $\mathbf{Y}$.

Pretraining objectives There are multiple ways to use the ground truth sequence for pretraining the transformer models (see Figure 2.3 for illustration):

- **Masked Language Modeling**: The goal is to predict a token at a masked position given both its left and right context. The objective is inspired by the Cloze task in psychology, where a similar task is given to human subjects (Taylor, 1953). The objective is commonly used for the encoder-only models such as BERT (Devlin et al., 2019).



(1) Masked language modeling

(2) Text denoising

(3) Causal language modeling

Figure 2.3: A scheme of the common objectives used by pretrained models: (1) masked language modeling, (2) text denoising, (3) causal language modeling. The special symbol <s> (beginning of a sentence) is used to bootstrap the decoding process.

- **Text Denoising**: The goal is generally to predict the original sequence from its corrupted version. The objective combines masked language modeling (MLM) with other tasks such as predicting a deleted token or predicting a number of missing tokens. It is used for pretraining encoder-decoder models such as BART (Lewis et al., 2019) or T5 (Raffel et al., 2019).
- **Causal language modeling**: The goal is to predict the next token given the previous sequence of tokens, as described in Equation 2.10. The objective used for pretraining decoder-only models, including GPT-2 (Radford et al., 2019b) and most of large language models (LLMs).

As a matter of fact, only causal language modeling adheres to the strict definition of a language model in Section 2.1.3 (Cotterell et al., 2024). However, all of these objectives are used in practice and are often combined with other auxiliary objectives such as next sentence prediction or token frequency prediction (Aroca-Ouellette and Rudzicz, 2020).

Finetuning By *finetuning* a model, we mean additional training of a pretrained model on a task-specific dataset. Finetuning is more efficient than training a model from scratch, but it is only a one-time process: repeated model updates can lead to erasing previous knowledge, also known as "catastrophic forgetting" (McCloskey and Cohen, 1989; Kirkpatrick et al., 2017). This problem is partially mitigated by multi-task finetuning, in which the tasks are unified under a pre-specified input format and the model is finetuned for all the tasks at once (Sanh et al., 2021; Xie et al., 2022).

Few-shot and Zero-shot Settings If the size of the finetuning data is very limited (under ~100 examples), we talk about *few-shot* setting. By limiting the finetuning data to zero, we arrive at a *zero-shot* setting, in which we use a model on the task which it has not been trained for. These settings are crucial for tasks with scarce data, also called *low-resource scenarios* (Hedderich et al., 2021).

Text Generation For generating text from a transformer decoder, we can use *left-to-right autoregressive decoding* (Jurafsky and Martin, 2024, p.196). The decoding process starts by feeding a special `<s>` (beginning of sequence) token into the decoder and iteratively selecting the i-th token based on the probability distribution for the i-th position. The decoding stops once the `</s>` (end of sequence) token is decoded. The procedure is outlined in Algorithm 1.

Algorithm 1 Autoregressive decoding

```
1: Initialize: $Y =$ <s>, $y =$ <s>                 ▷ Output sequence, current token
2: while $y \neq$ </s> do
3:     Predict next token probability distribution: $p(y|Y)$
4:     Select the next token: $y \sim p(y|Y)$
5:     Update output sequence: $Y = Y \cup y$
6: end while
7: Return $Y$
```

The sampling step (line 4) can be realized in various ways, including:

- **Greedy decoding**: Selecting the most probable token: $y_i = \arg\max_{y \in V} p_\theta(y|y_{<i})$.
- **Top-k sampling**: Sampling the next token from the k most probable tokens.
- **Top-p (nucleus) sampling** (Holtzman et al., 2019): Sampling the next token from the tokens with cumulative probability p.
- **Beam search**: Extending the k most probable sequences from the previous step with the next tokens, and selecting the k most probable sequences for the next step.

While greedy decoding and beam search are used to generate more probable sequences (approximating the exact algorithm for estimating the most probable sequence, which has exponential complexity), sampling algorithms are used to decode more creative outputs.[5]

[5] The list of the decoding algorithms as presented here is not exhaustive; see Zarrieß et al. (2021) and Wiher et al. (2022) for an overview and further discussion.

2.1.6 Large Language Models

Scaling the models in terms of the number of parameters and the size of the training data has turned out to further improve the performance of the models (Kaplan et al., 2020; Hoffmann et al., 2022). Larger models were shown to exhibit unprecedented capabilities in terms of language fluency, language understanding, and reasoning skills (Wei et al., 2022a; Bubeck et al., 2023), giving name to a specific category of *large language models (LLMs;* Brown et al., 2020; Zhao et al., 2023a). Broadly speaking, LLMs are larger than 1B parameters and based on the transformer decoder architecture. In many NLP tasks, from document-level translation (Wang et al., 2023b) and MT evaluation (Kocmi and Federmann, 2023b) to news summarization (Zhang et al., 2024) and story generation (Xie et al., 2023), LLMs have comparable or better performance than previous task-specific approaches.

In-context Learning LLMs can perform certain tasks without the need for fine-tuning on task-specific data $E_{\text{task}} = \{(x_1, y_1), \ldots, (x_n, y_n)\}$. Instead of training, we provide E_{task} as a part of the *prompt* (i.e., the text used as a decoding prefix). After E_{task}, we also append our test input x_{n+1}. By the virtue of causal language modeling and using other examples for the context, the model can be expected to decode the corresponding output $\hat{y}_{n+1}$. This ability is known as *in-context learning* (Brown et al., 2020; Dong et al., 2022). As the set of input-output examples is usually limited by the context size, we talk about *few-shot prompting*. If the model is given only the test input, we talk about *zero-shot prompting*.

Instruction Tuning Another key to strong cross-task performance of LLMs is instruction tuning: finetuning on a large dataset of tasks formulated using natural language instructions, such as *"Answer this question: {question}"* or *"Translate this sentence: {sentence}"* (Sanh et al., 2021; Ouyang et al., 2022). Due to their strong generalization abilities, the instruction-tuned models can be easily prompted to perform a task of choice in natural language, even without being directly trained for it.

2.2 Data-to-Text Generation

data-to-text (D2T) data-to-text

In this section, we provide background for the task of D2T generation. First, we present task itself, ... task itself the D2T applications (Section 2.2.1) and the subtasks to which D2T generation can be decomposed (Section 2.2.2). For the subtasks, we present rule-based (Section 2.2.3), (non-neural) statistical (Section 2.2.4), and neural (Section 2.2.5) approaches. In the final part, we describe the datasets (Section 2.2.6) and evaluation metrics (Section 2.2.7) we use in the thesis.

D2T

2.2.1 Task and Applications

D2T generation is an umbrella term for tasks that require transforming structured data into natural language. The input can take various forms, including graphs, trees, 2D tables, charts, or databases; the output is a fluent text that accurately conveys the information from the data (Gatt and Krahmer, 2018; Sharma et al., 2022).

Before we talk D2T generation from the research point of view, we present an overview of its practical applications:

- **Automated Journalism**: Augmenting (or, in simple cases, even replacing) human journalists for writing data-based reports, including:
  - **News reports**: Automating news writing, e.g., for election results (Leppänen et al., 2017), incidents (van der Lee et al., 2020), earthquakes (Oremus, 2014), or wildlife tracking (Siddharthan et al., 2012; Ponnamperuma et al., 2013).
  - **Sport reports**: Generating game summaries for sports such as basketball (Wiseman et al., 2017; Thomson et al., 2020), baseball (Puduppully et al., 2019), or soccer (van der Lee et al., 2017).
  - **Financial reports**: Supporting financial decisions by generating comments on stock prices (Murakami et al., 2017; Aoki et al., 2018) and summarizing financial documents (Chapman et al., 2022).
  - **Weather reports**: Generating weather forecasts and weather-related reports (Goldberg et al., 1994; Belz, 2005, 2008a; Angeli et al., 2010; Balakrishnan et al., 2019).

- **Business Intelligence Reports**: Providing decision support in business reports alongside data summaries and visualizations (mostly developed by commercial companies such as Arria, InfoSentience, or vPhrase; see also Dale (2023) for a recent overview). [illegible]

- **Chart Captioning**: Generating captions[6] for charts or graphs, e.g., for assistive technologies, document indexing, or simplifying decision support (Demir et al., 2008, 2012; Obeid and Hoque, 2020; Kantharaj et al., 2022).

- **Healthcare Summaries**: Providing clinical data summaries about patients to clinicians (Portet et al., 2009a; Scott et al., 2013), or providing medical information to patients, e.g., for behavioral change (Reiter et al., 2003) or nutritional counseling (Balloccu and Reiter, 2022).

[6]In contrast to image captioning (Stefanini et al., 2022), here the systems can rely on the underlying data in textual form (although the approaches can be hybrid, see e.g. Kantharaj et al., 2022).

- **Product Descriptions**: Automating generating product descriptions in specific domains such as for laptops and TVs (Wen et al., 2015a, 2016), or general-domain approaches for big e-commerce platforms (Shao et al., 2021; Koto et al., 2022).

2.2.2 D2T Generation Pipeline

#

Until recently, D2T generation was decomposed into approximately 4-6 subtasks[7] which were addressed separately (Reiter and Dale, 1997; Reiter, 2007; Gatt and Krahmer, 2018). Even though ~~the~~ recent advances enable approaches which solve the task in an *end-to-end* fashion (i.e., without intermediate steps), the subtasks are still relevant for conceptualization of D2T generation. We selected the following five representative subtasks, as illustrated in Figure 2.4:

choose to split the pipeline into

(1) **Content Selection**: Deciding which facts from the structured data to include in the text.

(2) **Document Planning**: Determining the order of the facts~~,~~ and dividing the facts into paragraphs.

(3) **Sentence Planning**: Aggregating the facts into sentences.

(4) **Lexicalisation**: Transforming the facts to text segments.

(5) **Surface Realisation**: Combining the text segments into a well-formed text in natural language.

Decomposing D2T generation into subtasks helps to modularize the system. Each module ~~can have~~ has a specific and well-defined function, which makes the system more explainable. Modularization also enables realizing each subtask using a different approach (see Sections 2.2.3 to 2.2.5).

The subtasks are typically executed in a *pipeline*, i.e., the input is sequentially processed by a series of modules. The main issue of pipeline-based approaches is error accumulation: the errors from one module propagat~~ing~~e to downstream modules. Despite this issue, the pipeline approach is the basis of rule-based D2T generation systems (Mille et al., 2023) and can also benefit neural-based systems (Moryossef et al., 2019b; Puduppully and Lapata, 2021; see also Section 3.3).

most? — templa

[7]The count is only approximate: for example, Mille et al. (2023) further subdivides some of the subtasks, leading to 10 subtasks in total.

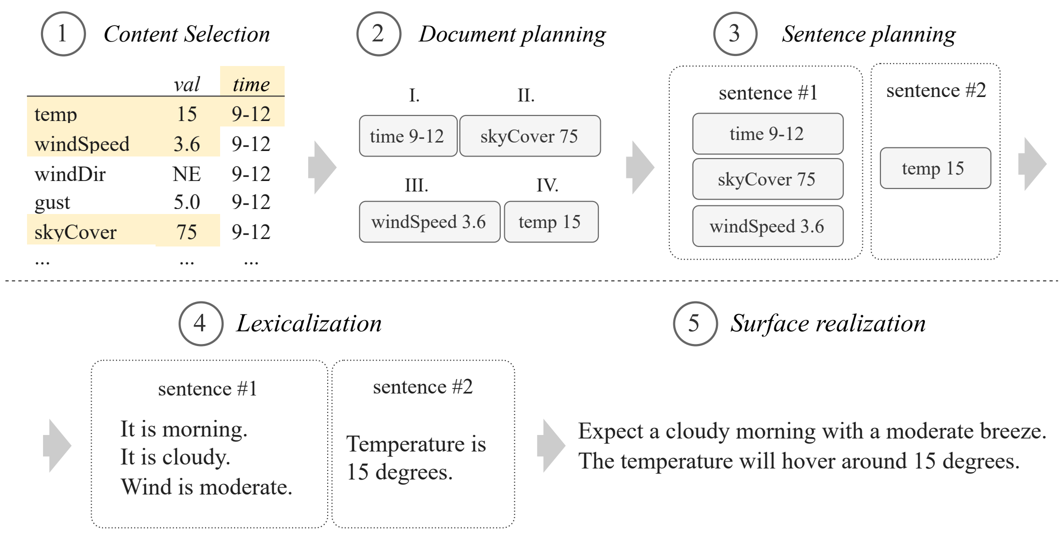


Figure 2.4: A five-step D2T generation pipeline on the example of generating a weather forecast. (1) The relevant fields for the forecast are selected from the data table. (2) The fields are ordered. (3) The ordered fields are aggregated into sentences. (4) Each field is transformed into a text segment. (5) The text segments are combined into the final text.

2.2.3 Rule-based Approaches

By *rule-based approaches* for D2T generation, we mean the approaches using manually defined rules or grammars.[8] Rule-based approaches are still in use in various forms today (Gatt and Krahmer, 2018; Dale, 2020, 2023). It is helpful to view these approaches through the lens of the whole D2T generation pipeline (Section 2.2.2), as these approaches typically tackle particular subtasks of the pipeline individually.

Content Selection Extracting meaningful information from the data typically relies on domain-specific heuristics, e.g., *"if a pattern is detected in the signal, include it in the report"* (Portet et al., 2009a). Various factors can influence the decision, including the target length of the report, the type of the report, and its target audience (Gkatzia, 2016).

Text Planning Rule-based text planning follows discourse strategies~~,~~ which are designed to satisfy the desired communicative goals (such as *define*, *compare*, or *describe*; McKeown, 1985). The resulting rules are typically in the form *"if a player scores two consecutive goals, describe these in the same sentence"* (Gatt and Krahmer, 2018).

[8]In contrast to the data-driven approaches (presented in Sections 2.2.4 and 2.2.5), which derive the system inner workings from the data.

Template-based Lexicalization Simpler rule-based approaches for lexicalization and surface realization are typically based on *templates*: pre-written text snippets with placeholders which are filled with values from the data. Templates can range from simple fill-in-the-blank approaches (such as *"The temperature will be {temp} degrees"*) to more sophisticated templates using a templating language (Gatt and Reiter, 2009; Reiter, 2016). Rules are used for selecting the templates, combining them, and filling the placeholders with values with (the last step being non-trivial in languages with rich morphology). The resulting rule-based system is usually tied to ~~the~~ a specific task and domain, but it can be a way to generate sufficient outputs with reasonable development time and costs (van der Lee et al., 2018).

Grammar-based Lexicalization To handle more complex cases of lexicalization in rule-based systems, we can use grammar-based approaches. Even though a *grammar* is technically also a set of rules, it differs by the fact that it describes the production rules for the whole sentence. Grammar-based approaches are rooted in linguistic theories, such as systemic grammars (Halliday, 1985; Matthiessen, 1991) or meaning-text theory (Mel'cuk et al., 1988; Goldberg et al., 1994). The implementation typically relies on off-the-shelf realizers such as FUF/SURGE (Elhadad and Robin, 1997) or KPML (Bateman, 1997). Grammar-based approaches are more general-purpose than rule-based approaches; however, they require considerable manual effort, detailed input, and often also additional rules for choosing among related options (Gatt and Krahmer, 2018).

2.2.4 Statistical Approaches

The idea of statistical[9] D2T generation approaches is to *estimate the parameters* of a system using a text corpus. This idea is not mutually exclusive with rule-based and grammar-based approaches; in fact, corpus statistics were initially used for re-ranking ~~the~~ outputs generated from a grammar-based system (Bangalore and Rambow, 2000; Langkilde, 2000; Ratnaparkhi, 2000) or even integrated directly at the level of generation decisions (Belz, 2008a).

[9]Since statistical D2T generation approaches overlap with classical machine learning methods, these approaches are perhaps better described as *pre-neural data-driven* approaches. However, we will stick to the more established term.

Even entirely fully data-driven approaches still relied on grammatical rules; [illegible] the rules were derived from treebanks, i.e., text corpora annotated with syntactic and semantic sentence structures. For example, the approach of White et al. (2007) relied on a Combinatory Categorial Grammar (Steedman, 2001) derived from the Penn Treebank (Hockenmaier and Steedman, 2007). Hybrid approaches ~~then~~ combined a set of hand-written rules or grammars with statistical models (Konstas and Lapata, 2012; Gardent and Perez-Beltrachini, 2017).

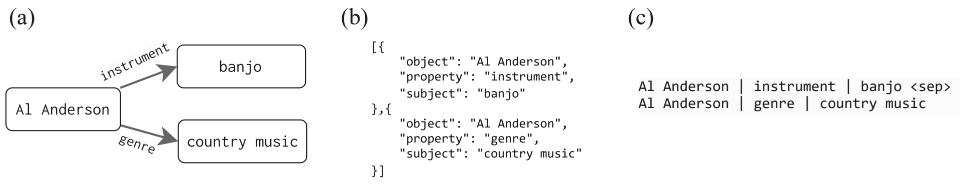
[illegible] ?

The earlier stages of the D2T generation pipeline, such as content selection or text planning, were usually tackled by *unsupervised* machine learning methods. For example, Duboue and Mckeown (2003) proposed to use a clustering-based method for content selection, estimating the relative importance of each cluster for the final text. Barzilay and Lee (2004) modelled the content structure using Hidden Markov Models (Baum and Petrie, 1966), learning the structure from unannotated documents. An example of a statistical approach for text planning is presented in Liang et al. (2009), who learn latent alignment between the text and the data for text segmentation and structuring.

2.2.5 Neural Approaches

Building upon the previous data-driven approaches, neural networks (see Section 2.1.1) began to be studied more widely in the context of D2T generation around 2015 (~~Dušek and Jurcicek, 2015~~; Wen et al., 2015a). Thanks to advances in hardware (Hooker, 2021) and capabilities of learning from large data (LeCun et al., 2015), neural networks enabled not only building more powerful modules for the D2T generation pipeline but also replacing the pipeline entirely with end-to-end models (Dušek et al., 2020). For a more detailed overview of neural D2T generation in recent years, we point the reader to the surveys of Sharma et al. (2022) and Lin et al. (2023); here, we mainly focus on the concepts and model architectures related to this thesis.

Linearization ~~For getting~~ To get an input sequence suitable for the neural model, structured data first needs to be converted into a sequence of tokens. To preserve the data structure while keeping the input simple, a common practice is to *linearize* the input: convert the data to a minimalistic representation with a handful of dedicated special tokens serving as delimiters. An example linearization of a knowledge graph is depicted in Figure 2.5 (c). Linearization can be very effective (Yang et al., 2020; Hoyle et al., 2021; Xie et al., 2022), beating specialized representations such as graph embeddings (Marcheggiani and Perez-Beltrachini, 2018; Koncel-Kedziorski et al., 2019).



```
[{
    "object": "Al Anderson",
    "property": "instrument",
    "subject": "banjo"
},{
    "object": "Al Anderson",
    "property": "genre",
    "subject": "country music"
}]
```

```
Al Anderson | instrument | banjo <sep>
Al Anderson | genre | country music
```

Figure 2.5: Representations of a simple knowledge graph: (a) the original knowledge graph, (b) JSON representation, (c) linearized representation.

is it?

An alternative to linearization is to use special embeddings for the structural elements (e.g., for table rows and columns), which are summed with the positional and token embeddings (Wang et al., 2021b; Yang et al., 2022). Since ~~the embeddings~~ these additional embeddings need to be learned, ~~the~~ this approach is more suitable for high-resource tasks.

also used in [illegible]

Delexicalization A specific data value may appear only a few or zero times in the training data, making it difficult for the model to learn its representation. Delexicalization is the process of replacing the values with placeholders, allowing the model to work only with the fill-in-the-blank templates instead of actual values (Oh and Rudnicky, 2000; Mairesse et al., 2010; Wen et al., 2015b; Dušek and Jurčíček, 2016). The values are filled in the post-processing step using simple rules. This approach was shown to be useful even for languages with rich morphology, where the values can be inflected using a dedicated language model (Dušek and Jurčíček, 2019).

Sequence-to-Sequence Generation Generating text from data in the end-to-end fashion, i.e., without intermediate steps, is enabled by neural sequence-to-sequence (seq2seq) models. Seq2seq models are designed for transforming variable-length input sequences into variable-length output sequences (Cho et al., 2014; Sutskever et al., 2014). The typical seq2seq architecture is the encoder-decoder framework described in Section 2.1.4. In the case of D2T generation, the input sequence is the linearized version of structured data, and the output sequence is the target text.

RNN-based Approaches The original seq2seq approaches were designed for MT (Cho et al., 2014; Sutskever et al., 2014), but soon were also adopted for other natural language generation (NLG) tasks. Wen et al. (2015b) use RNNs to generate the response in a dialogue system, using a structured representation of the dialogue act as the input. On a similar task, Dušek and Jurčíček (2016) show that instead of combining

linearization

the model with a separate surface realization system, it is better to generate output strings directly with an RNN. For generating weather reports, Mei et al. (2016) use RNNs to address the content selection step, identifying salient data records using the attention mechanism.

An important addition to the RNN-based approaches was the *copy mechanism*, which allows the model to generate the tokens by copying them from the input sequence (Gu et al., 2016; See et al., 2017). The copy mechanism is an alternative to delexicalization, delegating the job of filling the values to enabling the model to fill in lexical values by itself the model. Unlike delexicalization, the copy mechanism is trainable along with the rest of the model (Gehrmann et al., 2018).

RNNs were still used even after the introduction of the transformer model (Section 2.1.4), since they tend to work better in low-resource settings. For example, Freitag and Roy (2018) experimented with using text denoising as an objective pretraining a text denoising objective to pretrain... an RNN-based system for D2T generation. For adapting an RNN-based model to other domains, Wen and Young (2020) proposed *data counterfeiting*, i.e., replacing delexicalized slots with slots from another domain. To improve the faithfulness of the outputs, Rebuffel et al. (2021) propose an architecture based on 3 RNNs computing content, faithfulness, and fluency scores. Various shared tasks and comparisons (Gardent et al., 2017b; Dušek et al., 2020; Ferreira et al., 2019) showed that RNN-based approaches were generally competitive with rule-based approaches: the RNNs produce more fluent text, while the pipeline-based approaches ma~~d~~ke less semantic errors.

PLM-based Approaches Using a transformer model for D2T generation became practical with the arrival of PLMs discussed in Section 2.1.5. As an example, the 2020 WebNLG+ shared task (see Section 2.2.6) was dominated by systems based on pretrained encoder-decoder transformer models (Ferreira et al., 2020; Yang et al., 2020; Agarwal et al., 2020; Kasner and Dušek, 2020b). → see

PLMs made it possible to get rid of remove both *delexicalization* and *copy mechanism*. The general knowledge of language general language modeling pretraining, along with the learned ability to copy tokens from the input, allow the model to handle rare entities not present in the task-specific training data. PLMs are also able to produce outputs with considerably better fluency than RNN-based models. Moreover, ~~the~~ variants of PLMs pretrained on multilingual corpora (Liu et al., 2020; Xue et al., 2021) are able to handle produce outputs in a variety of languages.

Due to the advantages above, PLM-based approaches excel in low-resource settings, which is a default setup are common for many D2T generation tasks. Following Chen et al. (2019), other works adopted PLMs for few-shot or zero-shot D2T generation. In these scenarios, the models are typically finetuned on domain-specific data for few-shot generation (Chang et al., 2021b; Su et al., 2021a) or on ~~the~~ related domains for zero-shot

-> see Chapter X

generation (Kasner and Dušek, 2022; Kasner et al., 2023b). Improving PLM-based D2T generation in English revolves mainly around (1) finding *suitable data representations* and (2) ensuring the *semantic accuracy* of the outputs, both of which we will discuss in the following chapters.

LLM-based Approaches At the time of writing, approaches using ... are using LLMs for D2T generation is still in its naissance. which ~~The~~ works that compared zero-shot or few-shot LLM prompting with finetuned PLMs on existing datasets have found that ~~the~~ LLMs rank behind state-of-the-art finetuned models on automatic metrics (Axelsson and Skantze, 2023; Yuan and Färber, 2023). In Section 6.2, we will also show that LLMs can be employed for ~~the~~ zero-shot generation of data in standard data formats, with the main issue remaining still being the semantical accuracy of the outputs. However, there are , as of May 2024, yet no large-scale comparisons or attempts on of finetuning LLMs for D2T generation.

2.2.6 Datasets

In this section, we outline the format and structure of D2T generation datasets used in this thesis. The overview of the datasets is presented in Table 2.2 (note that we mainly focus on the datasets in boldface).[10]

Data Formats The following formats of structured data are present in the datasets which we employ in this thesis:

- **Key-value pairs**: The input is a set of tuples (k, v), where k is a key (also called a slot), which is typically a descriptive text string, and v is a value, which can be a text string or a number. The format is often used as a *meaning representation (MR)* in dialogue systems, e.g., for representing dialogue states or dialogue acts (Rastogi et al., 2020; Budzianowski et al., 2020). ≠ [illegible]
- **RDF (Resource Description Framework)[11] triples**: The input is a set of triples (s, p, o), where s is a *subject*, p is a *predicate*, and o is an *object*. This formalism directly translates to a *directed graph*, where s and o are nodes, and p is a directed edge between these nodes. In a knowledge graph such as Wikidata or DBPedia, the subject is usually an entity with a given identifier (e.g., a person, an object, or a place); the object is either another entity or a generic value. The predicate expresses the relation between the subject and the object. (string or [illegible])

[10] We do not describe here our novel datasets presented in Kasner et al. (2023b) and Kasner and Dušek (2024); these are described in their respective sections in Chapter 6.

[11] See https://www.w3.org/TR/PR-rdf-syntax/.

Dataset	Data Format	Domain(s)	# Ex.
CACAPO (van der Lee et al., 2020)	Key-value	News♦	20,149
DART (Nan et al., 2021)	RDF triples	Wikipedia♢	70,524
E2E (Dušek et al., 2019, 2020)	Key-value	Restaurants	36,856
EventNarrative (Colas et al., 2021)	RDF triples	Events♢	224,428
HiTab (Cheng et al., 2021)	Table w/hl	Statistics♢	10,672
Chart-To-Text (Kantharaj et al., 2022)	Table	Statistics♢	34,811
Logic2Text (Chen et al., 2020c)	Table w/hl	Wikipedia♢	10,753
LogicNLG (Chen et al., 2020a)	Table	Wikipedia♢	37,015
NumericNLG (Suadaa et al., 2021)	Table	Science♢	1,355
SciGen (Moosavi et al., 2021)	Table	Science♢	17,551
Rotowire* (Wiseman et al., 2017)	Table	Basketball	6,150
ToTTo (Parikh et al., 2020)	Table w/hl	Wikipedia♢	136,553
WebNLG (Gardent et al., 2017b)	RDF triples	DBPedia♦	42,873
WikiBio (Lebret et al., 2016)	Key-value	Biographies♢	728,321
WikiSQL (Zhong et al., 2017)	Table + SQL	Wikipedia♢	80,654
WikiTableText (Bao et al., 2018)	Key-value	Wikipedia♢	13,318

# of ex

Table 2.2: The list of D2T datasets used in this work. All the listed datasets are included in the TabGenie framework (Section 5.1), except for Rotowire, where we include instead an its updated version, dubbed SportSett:Basketball, instead (Thomson et al., 2020). Our main focus is on the datasets in **boldface**, which are used in multiple experiments. Glossary of data types: *Key-value*: key-value pairs, *Graph*: subject-predicate-object triples, *Table*: tabular data (*w/hl*: with highlighted cells), *Chart*: chart data, *SQL*: strings with SQL queries. ♦ indicates that the dataset is multi-domain; ♢ indicates that the dataset is open-domain.

- **Table**: The input is structured as a *table*, i.e., a two-dimensional cell matrix of m columns and n rows. A table cell can contain a textual or a numerical value. If a cell is marked as a heading, it contains the a "key" (a label) for the respective row or column. In some datasets, a subset of cells is *pre-highlighted* – in that case, the output text should describe only that particular subset of cells. Analogically to real-world data, we can also extend the definition of table with merged cells, nested headings, or multiple disjoint table regions. ← to [illegible] moment, change

As we show in Chapter 5, key-value pairs and RDF triples can be converted to a tabular format with minimal information loss. In Section 6.2, we also show how to handle data in the standard JSON[12] format, which is not commonly used in D2T generation datasets. We do not consider here data formats with a more task-specific structure such as abstract meaning representation (AMR; Banarescu et al., 2013).

omit [illegible]

Cut? A jet or a midpoint?

[12]JavaScript Object Notation; see https://www.json.org.

Domains In D2T generation, the notion of a *domain*—commonly used for drawing boundaries between the datasets or their subsets—mostly adheres to the dictionary definition of *an area of interest.*[13] However, its exact scope may vary: for example, while Wen et al. (2016) consider datasheets for TVs and laptops as coming from distinct domains, Lin et al. (2023) group all tables from ACL Anthology papers in a single domain (Suadaa et al., 2021).

The definition is more clear for the term *multi-domain*. Most commonly, a dataset is called *multi-domain* if two subsets of data come from distributions so different that the model trained on one subset does not straightforwardly generalize to the second (van der Lee et al., 2020; Budzianowski et al., 2020; Rastogi et al., 2020). If the topic of the dataset is unrestricted, or if it is based on a large-scale data source such as Wikipedia, the dataset is considered *open-domain* (see, e.g., Chen et al., 2020a; Nan et al., 2021; Kann et al., 2022).

(hl. tables)

Datasets The following D2T generation datasets are the most relevant for the thesis:

- **WebNLG**: The WebNLG dataset (Gardent et al., 2017a,b) contains RDF triples from DBPedia (Auer et al., 2007) and their crowdsourced descriptions. Each example consists of 1-7 triples, forming a subgraph in the DBPedia knowledge graph. The target text should describe all the entities and the relations between them. The original WebNLG dataset (Gardent et al., 2017a) contains 15 domains, out of which 5 are *unseen*, i.e., included only in the test set. Each set of triples included several verbalizations to promote lexical variability. In the version 2, the dataset has been fixed, annotated for intermediate subtasks, and enriched with semi-automated German translations (Shimorina and Gardent, 2018; Castro Ferreira et al., 2018). ~~The version 3~~ Version 3 of the data (Ferreira et al., 2020) contains one additional domain and automatic translations to Russian.

  # samples / domains

  - We participated in the 2020 edition of *WebNLG Challenge*, which is a series of shared tasks based on the WebNLG dataset (Gardent et al., 2017b; Shimorina et al., 2019; Ferreira et al., 2020; Cripwell et al., 2023; see Section 3.1). We also used the dataset in the experiments on low-resource D2T generation (Sections 3.2 and 3.3), evaluation (Section 4.1), data processing (Section 5.1), and out-of-domain generalization (Section 6.1).

- **E2E**: The E2E dataset (Dušek et al., 2020, 2019) contains restaurant descriptions in the form of key-value pairs (3-8 items per example) and corresponding human-written restaurant recommendations. The name of the dataset is derived from the E2E Challenge, a shared task that focused on evaluating end-to-end

[13] https://dictionary.cambridge.org/dictionary/english/domain

[illegible] ?

D2T generation systems (Dušek et al., 2020). Since the original version of the dataset ~~contained~~ was riddled with semantic noise (incorrect or missing facts in the crowdsourced descriptions), we use the cleaned version from Dušek et al. (2019) as the default version for our experiments.

  - Similarly to WebNLG, we used the dataset in ~~the~~ our experiments on low-resource D2T generation (Sections 3.2 and 3.3), evaluation (Section 4.1), and data processing (Section 5.1).

- **Rotowire**: Rotowire (Wiseman et al., 2017) is a dataset with tabular statistics of basketball games and their corresponding game summaries. The target text contains only a small subset of the full input table, which is not highlighted a priori, so the systems also need to model the content selection step. Together with the ~~above-average~~ full-paragraph length of the target summaries, this aspect makes the dataset particularly challenging for D2T generation systems.

  - We used the outputs from the neural systems on this dataset for building a token-level evaluation metric (Section 4.2). We also included its updated version SportSett:Basketball (Thomson et al., 2020) in our data processing toolkit (Section 5.1).

2.2.7 Evaluation Metrics

& this is our focus

The most common evaluation measures for D2T generation are *intrinsic*, i.e., focusing on evaluating certain aspects of the quality of the system and its outputs (Gkatzia and Mahamood, 2015; Celikyilmaz et al., 2021).[14] The intrinsic measures can be divided between *automatic metrics* and *human evaluation*. Automatic metrics are generally cheaper, faster, and more easily replicable. However, they mostly serve only as a crude heuristic for the desired performance measure. Human evaluation is more expensive and difficult to execute, but if executed correctly, it can give us a more precise and fine-grained picture of system performance. A rule of thumb is that an experimental result should be supported by both kinds of metrics.

[14]As opposed to *extrinsic* measures, which evaluate the impact of the system in the external environment (Celikyilmaz et al., 2021). While *extrinsic* metrics could give us a better picture of the real-world impact, they are not suitable for early research stages due to high demands on the system quality, and they are also less suitable for evaluating individual subtasks (van der Lee et al., 2019).

gold-standard

If we have human-written (also called *ground truth* or *golden*[15]) reference texts at our disposal, we can use *reference-based* automatic metrics. The implicit assumption with reference-based metrics is that the more similar the generated text is to the respective human-written reference text, the better. *Referenceless* metrics, on the other hand, can be more varied: they can either judge the intrinsic qualities of the text, such as ~~text~~ its fluency, ~~xxxx~~ diversity, and reading level, or—taking the input data into account—the faithfulness of the text with respect to the input data.

intro!

[illegible] LLM

Lexical Similarity Lexical similarity metrics measure the similarity between the generated and reference text using word-level (or character-level) overlap. ~~The~~ These metrics are fast, easy to compute, and have been used for decades as a convenient proxy for system comparison in various NLP areas (Celikyilmaz et al., 2021). However, there is a recent upsurge of works arguing against these metrics because their correlations with human judgments for high-quality outputs are low or negative, and the metrics fail to capture fine-grained phenomena (Mathur et al., 2020; Kocmi et al., 2021; Gehrmann et al., 2022). As a general rule, lexical similarity metrics (if used, e.g., for comparison with prior work) should be accompanied by other metrics.

Here are some of the common metrics which we use in this work:

- **BLEU** (Papineni et al., 2002) measures n-gram precision, i.e., to which extent ~~do~~ the n-grams in the generated text correspond to the reference text. It is computed as a geometric mean of the individual 1-4-gram ~~n-gram~~ precisions, with a brevity penalty to penalize ~~shorter outputs~~ outputs shorter than the reference. BLEU was originally used for evaluating MT, but it has become commonplace in NLP. [illegible] The sacreBLEU library (Post, 2018) was developed to tackle inconsistencies in implementations of the metric.
- **ROUGE** (Lin, 2004) is a set of metrics that focus on recall, i.e., to which extent does the generated text preserve the information in the reference text. ROUGE has been originally designed for evaluating automatic summarization, but similarly to BLEU, it has been used widely (and as recently found by Grusky (2023), oftentimes incorrectly) across the NLP literature. ROUGE includes several variants, such as ROUGE-L, which measures the longest matching word sequence, and ROUGE-1/2/3/4, which measures the overlap on the respective n-grams.

[15]The term *golden* can misleadingly suggest that human-written references are the "holy grail" which the systems should imitate. This is generally an overstatement, as human-written references are often noisy and faulty (Dušek et al., 2019; Clark et al., 2021), but they can still serve as a valuable point of reference.

- **METEOR** (Banerjee and Lavie, 2005) is a metric that computes the harmonic mean of precision and recall, computed on unigrams. METEOR also partially addresses semantic similarities of texts by using stemming and synonym matching. It has been shown to produce better correlations with human judgments than BLEU (Agarwal and Lavie, 2008) but is more complex and expensive to compute.
- **NIST** (Martin and Przybocki, 2000) is a metric which focuses on precision similarly to BLEU. However, it assigns ~~lower~~ higher weights to less common n-grams, which are considered ~~less~~ more informative (Doddington, 2002). Its length penalty is also more robust to slight variations in text length.
- **ChrF++** (Popović, 2015, 2017) is a metric which computes the F1-score on *character* n-grams. The metric is more robust to morphological variations than word-level metrics. On top of the original ChrF metric, Chr~~f~~F++ also considers word unigrams and bigrams along with the character n-grams.

Semantic Similarity As described in Section 2.1.2, word embeddings map words with similar meanings to the [illegible]red part of the vector space. Semantic similarity metrics use this fact to measure the similarity of ~~the~~ texts as a distance between their embeddings. The metrics most often rely on *contextual embeddings* computed by pretrained RNN or transformer encoders (Peters et al., 2018; Devlin et al., 2019; see Section 2.1.~~4~~5). In contrast ~~with~~ to lexical similarity metrics, semantic similarity metrics are more robust to ~~the~~ lexical variations but are more computationally expensive. They are also subject to ~~the~~ limitations of pretrained models, including their biases and black-box nature.

The following are the metrics which we use in this work:

- **BERTScore** (Zhang et al., 2019) measures the semantic similarity of ~~the~~ texts by computing cosine similarity between the embeddings of the texts encoded by a pretrained transformer model. It was initially developed on top of BERT (Devlin et al., 2019), but it now also supports other transformer encoder models. Its flexibility helps to achieve better correlations with human judgment but makes it less suitable for i[illegible]a-work comparison.
- **BLEURT** (Sellam et al., 2020) measures semantic similarity using a BERT model (Devlin et al., 2019) which is further finetuned on synthetically labeled data. The similarity is computed as a cosine similarity of the contextual embeddings. Compared to BERTScore, BLEURT is less flexible but ensures a more consistent setup across works.

- **NUBIA** (Kane et al., 2020) measures semantic equivalence of texts by combining features from two finetuned RoBERTa models (Liu et al., 2019b), on the semantic similarity benchmark STS (Cer et al., 2017) and on the natural language inference benchmark MNLI (Williams et al., 2018); along with ~~the~~ perplexity from the GPT-2 model (Radford et al., 2019b). These features are combined using an MLP layer. Combining the features ensures better robustness of the metric at the cost of higher complexity and higher computational requirements.

Semantic Accuracy Semantic accuracy—also called *faithfulness* or *factual consistency* (Celikyilmaz et al., 2021)—measures inaccuracies in the output text with respect to the input data. The inaccuracies can be broadly divided into *omissions* (the model not mentioning facts in the input data) and *hallucinations* (the model mentioning extra facts that are not supported by the input data). Naturally, omissions apply only if the task requires mentioning all the facts in the input data. Further, ~~the~~ hallucinations can be *extrinsic*, i.e., the model introduces external information not present in the data, or *intrinsic*, i.e., the model uses the data incorrectly (Maynez et al., 2020). + a?

There are no widely accepted metrics for measuring semantic accuracy in D2T generation. The closest notion is the *slot error rate* from task-oriented dialogue systems, which is typically implemented by exact string match or regular expressions (Wen et al., 2015b; Dušek et al., 2020). For tabular data, PARENT (Dhingra et al., 2019) was proposed as a reference-based metric, which uses lexical alignment models for computing precision and recall for tabular values. In Sections 4.1 and 4.2, we present two novel *referenceless* metrics for evaluating semantic accuracy of D2T generation using PLMs. In Section 6.2, we also show how to evaluate the semantic accuracy of texts using a LLM. + the metric

Text Fluency Text fluency is a catch-all term for measuring grammatical correctness, spelling, word, and stylistical choices of text (Celikyilmaz et al., 2021). Since texts that are more similar to human written text tend to be more fluent, lexical and semantic similarity metrics (such as BLEU) are often used as a proxy for measuring text fluency. Alternatively, we can use the *perplexity* of the text under a neural LM. This approach assumes that the LM assigns lower probability to less fluent? probable sentences (Lee et al., 2022; Kane et al., 2020). really

Lexical Diversity Lexical diversity measures the variability and richness of expressions in the text (van Miltenburg et al., 2018). One way to express lexical diversity is the ratio between the average number of different words and the total number of words, called *type-token ratio (TTR)* or *distinct n-grams* (Johnson, 1944; Li et al., 2016). Another way is to measure (conditional) entropy of n-grams (Shannon, 1948). Lexical diversity is not generally required ~~for~~ in D2T generation, although there are approaches explicitly aiming to decode diverse outputs (Han et al., 2021; Perlitz et al., 2022).

Human Evaluation Since automatic metrics serve only as imperfect proxies for human judgment, using human annotators is a crucial part of any NLG experimental evaluation (Gehrmann et al., 2022). Although there are attempts at standardizing human evaluation (Thomson and Reiter, 2020), human annotation protocols are usually task-specific (van der Lee et al., 2019; Belz et al., 2020). There are two main paradigms of human evaluation: large-scale evaluation using crowd workers mainly focusing on quantitative aspects (*crowdsourcing*), and small-scale evaluation using expert annotators focusing primarily on qualitative aspects (*manual evaluation*).

- **Crowdsourcing**: Crowdsourcing platforms such as Amazon Mechanical Turk[16] or Prolific[17] are often used for distributing the work between human annotators. These platforms offer a convenient interface for hiring annotators with a specific background. Due to financial incentives, the quality of outputs may vary, especially since the workers are nowadays prone to delegating the task to LLMs (Veselovsky et al., 2023). It is, therefore, necessary to employ quality assurance checks in the annotation process.

Expert EV analysis (?)

- **Manual Evaluation**: To measure fine-grained aspects of output quality, manual evaluation can be performed by the paper authors or other domain experts on a moderate-sized sample of data (~100 examples). The main goal of manual evaluation is to provide insights into the kinds of errors that appear in the output texts.

LLM-based Evaluation Recently, researchers have started to examine the potential of replacing human annotators with LLM~~s~~-based metrics (Zhao et al., 2023b; Sottana et al., 2023; Kocmi and Federmann, 2023b; Chiang and Lee, 2023; Wang et al., 2023a; Fu et al., 2023). In particular, the GPT-4 model (OpenAI, 2023b) was shown to be better capable of following fine-grained instructions compared to other LLMs and of having high correlations with human judgment on evaluating generated texts. Since the model

[16] https://www.mturk.com
[17] https://prolific.com

can be prompted for the specific task, using LLMs can be cheaper and more robust than human annotators. However, due to concerns about its non-reproducibility (Kocmi and Federmann, 2023a) and bias (Wang et al., 2023c), this evaluation method is only experimental.

(→ §.[illegible].XX)

3 Low-Resource Data-to-Text Generation

This chapter introduces three low-resource data-to-text (D2T) generation approaches based on pretrained language models (PLMs). By *low-resource*, we mean using as little data as possible for generating fluent and accurate texts. We develop approaches that leverage the general-domain pretraining of PLMs to generate texts in domains with thousands, hundreds, or even zero training examples.

The data we focus on are *RDF triples* from factual knowledge graphs in the WebNLG dataset and *key-value meaning representations* in the E2E dataset. Both datasets are described in Section 2.2.6. These datasets assume that content selection was performed beforehand, i.e., we always want to verbalize the whole input.

The most straightforward setting, presented in Section 3.1, consists of finetuning mBART (Liu et al., 2020), a pretrained transformer encoder-decoder model. For finetuning the model, we need approximately thousands of in-domain examples. We show that this baseline is powerful, achieving competitive results on a shared task for generating knowledge graph descriptions. On top of that, we show that this approach can be directly applied extended to non-English settings, namely to text generation in Russian.

In Sections 3.2 and 3.3, we present approaches that can generate texts with an even more limited amount of in-domain training examples. Our key idea is to use a PLM only as a tool for improving text fluency *regardless of the content* and delegating (possibly crude and basic, but factually correct) verbalization of the content to different, more controllable means, i.e. templates. Section 3.2 shows an approach based on a text-editing model, which has a limited vocabulary and is trained on iteratively fusing simple templates. The limited vocabulary and a specific training objective help~~s~~ the model to generate factually correct sentences. In Section 3.3, we present an alternative approach that adds an ordering and aggregation step for generating more fluent texts. Moreover, we show how to train a PLM for all the steps entirely on domain-general? general-domain operations, eliminating the need for in-domain training examples.

3.1 Finetuning Pretrained Language Models

This section is based on the paper *Train Hard, Finetune Easy: Multilingual Denoising for RDF-to-Text Generation* (Kasner and Dušek, 2020b), joint work with Ondřej Dušek, published in the Proceedings of the 3rd International Workshop on Natural Language Generation from the Semantic Web (WebNLG+) at INLG 2020.

This section introduces a simple approach for generating knowledge graph descriptions. Our approach is based on finetuning a multi-lingual PLM on linearized graphs and the accompanying human-written descriptions from the WebNLG dataset. In the WebNLG+ 2020 Shared Task, cite our model ranked in the first third of the leaderboard for English and the first or second for Russian on automatic metrics. It also made it in the best or second-best system cluster on human evaluation. We show that with a moderate amount of in-domain finetuning data, a simple PLM-based baseline can achieve satisfactory results in generating descriptions of knowledge graphs. We also point out its limitations, namely its inability to infer semantics of ambiguous relation labels; a topic to which we will return in Section 6.1.

3.1.1 WebNLG+ Shared Task

The WebNLG Challenge 2020[1] (WebNLG+; Ferreira et al., 2020) was the second edition of the shared task in graph-to-text generation. The task was based on the WebNLG dataset containing subgraphs from the DBpedia knowledge graph. Each subgraph is described by a set of RDF triples and accompanied by crowdsourced text descriptions (see Section 2.2.6). On top of the original challenge (Gardent et al., 2017b), WebNLG+ included a separate track for generating texts in Russian, in which we also participated.

3.1.2 Problem Formulation

Our input is a set of RDF triples $x \in X$, where each triple $x = (s, p, o)$ describes the relation p between the entities s and o in the knowledge graph. Our target output Y is a fluent and semantically accurate natural language description of X.

[1] https://synalp.gitlabpages.inria.fr/webnlg-challenge/challenge_2020/

We formulate the task as *sequence-to-sequence* generation. First, we linearize the input sequence (see Section 2.2.5) in the default order using two arbitrary separator tokens: one to delimit the triple constituents and another to delimit individual triples. Using the linearized sequence as an input and the target text as an output, we finetune a pretrained encoder-decoder model for the cross-entropy objective (Equation 2.10). With the finetuned model, we generate the target texts using autoregressive decoding (see Algorithm 1).

3.1.3 Implementation

Data Preprocessing We use the provided XML WebNLG data reader[2] to load and linearize the triples. For each triple, we use the `flat_triple()` method which converts each triple into the "`s | p | o`" string, using a pipe ("|") as a separator. We use another token not present in the training data ("▶") for delimiting individual triples to avoid extending the model vocabulary.[3] We linearize the triples in their default order. For the input to the model, we tokenize the data using SentencePiece tokenizer (Kudo and Richardson, 2018) trained on the training dataset, using a vocabulary of 250,000 subword tokens.

Model We use mBART (Liu et al., 2020), a multilingual PLM based on BART, a transformer model pretrained on text denoising (see Section 2.1.5). The model uses 12 layers for the encoder and 12 layers for the decoder ($\sim$680M parameters), and it is pretrained on the large-scale CC25 corpus extracted from Common Crawl, which contains data in 25 languages (Wenzek et al., 2020).

Training We finetune the pre-trained `mbart.CC25` model from the FAIRSEQ toolkit (Ott et al., 2019) using the default parameters,[4] changing only the total number of updates from 40k to 10k to reflect the smaller size of our data. We train a separate version of mBART for each language: mBART$_{\text{en}}$ on English inputs and English outputs, and mBART$_{\text{ru}}$ on English inputs and Russian outputs.

3.1.4 Results

We report on WebNLG automatic and human evaluation results, as well as our own error analysis.

[2] `https://gitlab.com/webnlg/corpus-reader`

[3] We chose the separators arbitrarily, as the model is to be finetuned with the selected separators.

[4] We use dropout 0.3, attention dropout 0.1, and 1024 tokens per batch; we set the initial learning rate to 0.0003 and use polynomial decay with 2500 warmup steps. We train the model using the Adam optimizer (Kingma and Ba, 2014) with $\beta_1 = 0.9$, $\beta_2 = 0.98$ and $\varepsilon = 1e-06$. For the full set of training arguments, see `https://github.com/facebookresearch/fairseq/tree/main/examples/mbart`.

input	`Piotr_Hallmann	weight	70.308 ▶ Piotr_Hallmann	birthDate	1987-08-25`
out (en)	Born on August 25th 1987, Piotr Hallmann has a weight of 70.308.
in	`Ciudad_Ayala	populationMetro	1777539`
out (en)	The population metro of Ciudad Ayala is 1777539.
in	`Bakewell_tart	ingredient	Frangipane`
out (ru)	Франжипан - один из ингредиентов тарта Бейквелл.
transcr.	Franzhipan - odin iz ingredientov tarta Bejkvell.
transl.	Frangipane is one of the ingredients of the Bakewell tart.

Table 3.1: Example outputs from the mBART model(s) finetuned for RDF-to-text generation. (1) The model can work with unseen entities, dates, and numbers. (2) The label deviates too much from its meaning for the unseen property `populationMetro`, leading to incorrect output. (3) The model trained on Russian targets can use English data to form sentences in Russian, transcribing the entities to Cyrillic.

Automatic Metrics The results of our approach for English are shown in Table 3.2. Our approach beats the baseline in all metrics and places in the first third of the submissions. While it loses performance on unseen categories, the drop is less dramatic than other competing approaches. For Russian, the results are shown in Table 3.3. Our system not only beats the baseline by a large margin (as did all competing submissions), but it ranks first in $\overset{\text{two}}{2}$ metrics out of $\overset{\text{four}}{4}$ (BLEU, BERTScore) and second in the remaining ones.

Human Evaluation The challenge organizers ran a human evaluation campaign, asking annotators to rate the texts for data coverage, relevance, correctness, text structure, and fluency. Each criterion has been rated with a number ranging from 0 (completely disagree) to 100 (completely agree). The scores were clustered into groups (1-[illegible] being the best), among which there are no statistically significant differences according to the Wilcoxon rank-sum test (Wilcoxon, 1992).

Our systems made it into the top clusters (1 or 2) for both English and Russian. For English, our mBART$_{en}$ system ranks first for all the categories in *seen domains*, and first or second in *unseen entities* and *unseen domains*. In total, our English system achieved rank 1 for relevance, correctness and text structure, and rank 2 for data coverage and fluency. For Russian, our mBART$_{ru}$ system ranks second for correctness and first in all other categories.

		BLEU		METEOR		ChrF++		BERTScore		BLEURT	
All	Ours	50.34	(10)	0.398	(8)	0.666	(8)	0.951	(8)	0.57	(8)
	Baseline	40.57	(14)	0.373	(15)	0.621	(15)	0.943	(14)	0.47	(12)
Seen Cat.	Ours	59.13	(10)	0.422	(10)	0.712	(9)	0.960	(9)	0.58	(14)
	Baseline	42.95	(31)	0.387	(27)	0.650	(28)	0.943	(31)	0.41	(31)
Unseen Cat.	Ours	42.24	(10)	0.375	(13)	0.617	(10)	0.943	(11)	0.52	(10)
	Baseline	37.56	(12)	0.357	(15)	0.584	(15)	0.940	(12)	0.44	(12)
Unseen Ent.	Ours	51.23	(4)	0.406	(8)	0.687	(7)	0.959	(8)	0.63	(8)
	Baseline	40.22	(17)	0.384	(15)	0.648	(15)	0.949	(13)	0.55	(12)

Table 3.2: Results of mBART_{en} (all data, seen categories, unseen categories, unseen entities), compared to the baseline from the organizers. The numbers in brackets show the rank of each model (out of 35 submissions) with respect to the given metric.

	BLEU		METEOR		ChrF++		BERTScore	
Ours	52.93	(1)	0.672	(2)	0.677	(2)	0.909	(1)
Baseline	23.53	(12)	0.461	(12)	0.511	(12)	0.836	(12)

Table 3.3: Results of mBART_{ru}, compared to the baseline. The numbers in brackets show the rank of each model (out of 12 submissions) if ordered by the given metric.

Manual Analysis To better understand the nature of errors made by our system, we manually inspected a sample of 50 outputs in each language.[5] We found factual errors in 12 English outputs, mostly concentrated along the unseen categories (*Scientist*, *Movie*, *Musical Record*). The model tends to describe musical works and movies in terms of written works ("written", "published" etc.), i.e., the closest seen category. There are also several swaps in roles of the entities (e.g., "is to southeast" instead of "has to its southeast", "follows" instead of "is followed by" etc.).

In a few cases, the model hallucinates a relation not specified in the data (e.g., "born on January 1, 1934 in Istanbul" when a date of birth and current residence is given, not the birthplace) or is not able to infer background knowledge not given on the input (it talks about a dead person in the present tense). Factual errors in Russian were less frequent (9 sentences), which is expected as there are no unseen categories. Moreover, the system shows an impressive performance at translating entity names from the English RDF into Russian.

We further found 10 outputs with suboptimal phrasing in English and 9 in Russian, where the model did not connect properties of the same type in coordination (e.g., two musical genres for a record) or gave numbers without proper units (e.g., "runtime of 89.0" or "area of 250493000000.0").

[5]Automatic back-translation to English was used to facilitate understanding of Russian.

3.1.5 Discussion

Why Does Our Approach Work Our solution benefits from the mBART model, which has absorbed vast amounts of factual world knowledge during pretraining (Petroni et al., 2019). Combined with its ability to produce fluent texts, the model expectedly performs well at generating short, factually grounded sentences. Moreover, the multilingual pretraining of the model allows us to use a single architecture for both English and Russian. We note that a careful choice of hyperparameters seems necessary for optimal performance, as other solutions in the challenge also used pretrained models with similar architecture but uneven results.[6]

Limitations Building a high-quality training set of in-domain data, as the one we had at our disposal, requires substantial human effort and financial resources. The generalization to other languages also does not come cheap: as English and Russian are the two most represented languages in the mBART pre-training corpora (ca. 300 GB of data each), the performance of our model would be supposedly lower in other languages. The performance of our model is noticeably lower on categories unseen in training (which, as we show in Section 6.1, is a non-trivial issue), and the model may not generalize well to examples longer than encountered in the training data (Zhou et al., 2023; Xu et al., 2023).

Beyond In-Domain Finetuning In Sections 3.2 and 3.3, we introduce approaches for D2T generation that cut down on the need for an extensive amount of in-domain training data. These approaches still rely on the existence of PLMs, but the models are given inductive bias necessary for the task in question: namely, that the goal is to transform a disfluent input (i.e., the structured data in its original format) into a fluent output (i.e., the structured data expressed in natural language). With these approaches, we circumvent the need for high-quality *in-domain* data by learning general-purpose text-to-text operations on *open-domain* data.

Future of the WebNLG Shared Task The 2023 WebNLG Shared Task, which took place three years later, featured our system as the baseline for the Russian graph-to-text generation track (Cripwell et al., 2023). The organizers of the task note that *"results on Russian for the present edition provide very small improvements over the best results for 2020."* These results suggest that (1) the WebNLG task is saturated (at least

[6]Nevertheless, in our case, we achieved satisfactory results using the default parameters.

for high-resource languages), yielding only small improvements regardless of the technique, and (2) fixing the long tail requires approaches allowing to identify and correct unclear input cases (e.g., by human interventions based on model uncertainty), which may not be possible in the framework of the shared task.

Relation to Large Language Models Surprisingly, our findings are still valid in the era of large language models (LLMs). As shown by Axelsson and Skantze (2023) and Yuan and Färber (2023), the GPT-3.5 model (OpenAI, 2023a) *does not* generally outperform finetuned systems on the WebNLG dataset. In particular, Axelsson and Skantze (2023) compared zero-shot performance GPT-3.5 to the systems of the WebNLG 2020 challenge and found the model achieves similar performance as our system on English while not outperforming the best systems in the challenge. The LLM makes semantic errors (as we also discuss in Section 6.2) and performs significantly worse on Russian data than on English data. Yuan and Färber (2023) further demonstrate that the LLM is hard to control with respect to the output format. In addition to that, the LLMs may have an unfair advantage in these evaluations since they may have memorized the outputs on the WebNLG test set (Balloccu et al., 2024).

3.2 Iterative Sentence Fusion

This section is based on the paper *Data-to-Text Generation with Iterative Text Editing* (Kasner and Dušek, 2020a), joint work with Ondřej Dušek, published in the Proceedings of the 13th International Conference on Natural Language Generation (INLG 2020).

In this section, we present an approach for generating semantically accurate texts from structured data in low-resource settings. Our approach builds on a text-editing model trained on the task of *sentence fusion*. After transforming individual data items to text using trivial templates, we iteratively improve the resulting text by applying sentence fusion, filtering, and re-ranking. Although our approach gets lower scores on lexical similarity metrics on WebNLG and E2E datasets than the state-of-the-art approaches, it achieves high levels of semantic accuracy due to the limited scope of the sentence fusion model and the guaranteed presence of the entities. We also demonstrate that our task formulation allows zero-shot D2T generation by training a model on a general-domain dataset for sentence fusion. The code for ~~the~~ our experiments is available on Github.[7]

domain-general?

[7]https://github.com/kasnerz/d2t_iterative_editing

3.2.1 Motivation

We aim to improve the semantic accuracy D2T generation. Other works have pursued this goal, e.g., by adapting the decoding algorithm (Tian et al., 2020), improving the robustness of the model by injecting noise in its hidden states (Kedzie and McKeown, 2019), or self-training with a natural language understanding model (Nie et al., 2019). Our approach is inspired by the systems which use a *generate-then-rerank* approach (Dušek and Jurčíček, 2016; Juraska et al., 2018), e.g., using a classifier to filter incorrect outputs (Harkous et al., 2020).

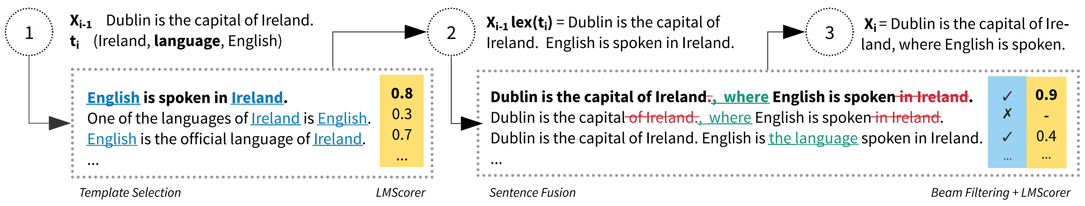
To generate outputs with sufficient semantic accuracy for the filtering step, we take advantage of three facts: (1) we can lexicalize individual data items using trivial templates, (2) concatenating the lexicalizations tends to produce an unnatural sounding but semantically accurate output, and (3) a PLM trained on improving the output fluency can be used for combining the lexicalizations.

3.2.2 Method

We focus on data structured as RDF triples. In our approach, we start from single-triple templates and iteratively fuse them into the resulting text while filtering and reranking the results. We first detail the main components of our system (template extraction, sentence fusion, PLM scoring) and then give the overall description of the generation algorithm.

Template Extraction We collect a set of templates for each unique predicate. We use two approaches: (a) handcrafting the template manually for each predicate in the training set and (b) automatically extracting the template from the lexicalizations of the examples in the training set. For unseen predicates, we add a single fallback template: *The <predicate> of <subject> is <object>.*

Sentence Fusion We train a model for the task of *sentence fusion*, i.e., combining sentences into a coherent text (Barzilay and McKeown, 2005). To construct the training data for the model, we select pairs of examples (X, X') and their corresponding text descriptions (Y, Y') from the original training set such that the examples consist of $(k, k+1)$ triples and have k triples in common. This leaves us with an extra triple x_{k+1} present only in X'. For each training example, we use the concatenated sequence $Y \text{lex}(x_{k+1})$ as a source and the sequence Y' as a target, where $\text{lex}(x_{k+1})$ denotes lexicalizing the triple x_{k+1} using an appropriate template. As a result, the model learns to integrate Y and x_{k+1} into a single coherent expression.



1
X_{i-1} Dublin is the capital of Ireland.
t_i (Ireland, language, English)
English is spoken in Ireland.
One of the languages of Ireland is English.
English is the official language of Ireland.
...
0.8
0.3
0.7
...
Template Selection
LMScorer
2
X_{i-1} lex(t_i) = Dublin is the capital of Ireland. English is spoken in Ireland.
Dublin is the capital of Ireland~~.~~, where English is spoken ~~in Ireland~~.
Dublin is the capital ~~of Ireland.~~, where English is spoken ~~in Ireland~~.
Dublin is the capital of Ireland. English is the language spoken in Ireland.
...
Sentence Fusion
3
X_i = Dublin is the capital of Ireland, where English is spoken.
✓ ✗ ✓ ...
0.9 - 0.4 ...
Beam Filtering + LMScorer

Figure 3.1: A single iteration of our algorithm for iterative D2T generation. In Step 1, the template for the triple is selected and filled. In Step 2, the sentence is fused with the template. In Step 3, the result for the next iteration is selected from the beam by filtering and language model scoring.

PLM Scoring For re-ranking the text, we use an additional component for computing text fluency, which we refer to as LMScorer. As described in Section 2.2.7, we use perplexity of the text under a a PLM, computing the score of the output text Y composed of tokens $(y_1, \dots, y_n)$ as a geometric mean of the token conditional probability:

$$\text{score}(Y) = \left(\prod_{i=1}^{n} P(y_i | y_1, \dots, y_{i-1}) \right)^{\frac{1}{n}}. \tag{3.1}$$

Generation Algorithm The input of the algorithm (Figure 3.1) is a set of T ordered triples. First, we lexicalize the triple x_0 to get the output text Y_0 by filling the available templates and using the template with the best score from LMScorer. In each of the following steps $i = (1, \dots, T-1)$, we lexicalize the triple x_i and concatenate it with Y_{i-1}. To improve the fluency of the text, we use the sentence fusion model with beam search to produce k hypotheses. We filter and re-rank the hypotheses (see the next paragraph), getting Y_i for the next step. The output is the text Y_{T-1} from the final step.

Filtering and Re-ranking In each decoding step, we remove hypotheses in the beam missing any entity from the input data using a simple heuristic based on string matching. We rescore the remaining hypotheses in the beam with LMScorer and set the hypothesis with the best score as Y_i. In case there are no sentences left in the beam after the filtering step, we let Y_i be the text in which the lexicalized x_i is appended after Y_{i-1} without sentence fusion, ensuring the semantic accuracy of the text.

dataset	method	predicate	example #1	example #2
WebNLG	extracted	`foundedBy`	o was the founder of s.	s was founded by o.
E2E	extracted	`area+food`	s offers o_2 cuisine in the o_1.	s in o_1 serves o_2 food.
E2E	manual	`near`	s is located near o.	o is close to s.

Table 3.4: Examples of templates we used in our experiments. The markers s and o are placeholders for the subject and the object, respectively. The templates for the single predicates in the WebNLG dataset and the pairs of predicates in the E2E dataset are extracted automatically from the training data; the templates for the single predicates in E2E are created manually.

3.2.3 Implementation

Template Extraction We experiment with the WebNLG and E2E datasets (see Section 2.2.6). For WebNLG, we extract the templates from the training examples containing only a *single* triple. In the E2E dataset, there are no such examples; therefore, we first extract the templates for *pairs* of predicates, using them as a starting point for the algorithm to leverage the lexical variability in the data (manually filtering out the templates with semantic noise). We also create a small set of templates for each *single* predicate manually, using them in the subsequent steps of the algorithm.[8] See Table 3.4 for examples of templates we used in our experiments.

Sentence Fusion Model We base our sentence fusion model on the text-editing model LaserTagger (Malmi et al., 2019). LaserTagger generates outputs by tagging inputs with edit operations (`KEEP` a token, `DELETE` a token, and `ADD` a phrase before the token), which makes it suitable for tasks where the output highly overlaps with the input. An important feature of LaserTagger is its limited vocabulary size, consisting of k most frequent (possibly multi-token) phrases used to transform inputs to outputs in the training data. After the vocabulary is precomputed, all infeasible examples in the training data are filtered out. At the cost of limiting the number of training examples, this filtering makes the training data cleaner by removing outliers. The limited vocabulary also makes the model less prone to hallucination errors.

LMScorer As the LMScorer backend, we use the pre-trained GPT-2 language model (Radford et al., 2019b) from the Transformers repository[9] (Wolf et al., 2019). We compute the perplexity scores using the *lm-scorer*[10] package.

[8]In the E2E dataset, the data is in the form of key-value pairs. We transform the data to RDF triples by using the name of the restaurant as a *subject* and the rest of the pairs as *predicate* and *object*. This creates *n-1* triples for *n* pairs.

[9]`https://github.com/huggingface/transformers`

[10]`https://github.com/simonepri/lm-scorer`

	WebNLG				**E2E**			
	BLEU	*NIST*	*METEOR*	*ROUGE$_L$*	*BLEU*	*NIST*	*METEOR*	*ROUGE$_L$*
baseline	0.277	6.328	0.379	0.524	0.207	3.679	0.334	0.401
zero-shot	0.288	6.677	0.385	0.530	0.220	3.941	0.340	0.408
w/fusion	0.353	7.923	0.386	0.555	0.252	4.460	0.338	0.436
SFC	0.524	-	0.424	0.660	0.436	-	0.390	0.575
T5	0.571	-	0.440	-	-	-	-	-

Table 3.5: Results of automatic metrics on the WebNLG and E2E test sets.

3.2.4 Experiments

Base Experiments As a *baseline*, we generate the best templates according to LMScorer without applying the sentence fusion (i.e., always using the fallback). For the *sentence fusion* experiments, we use LaserTagger with the autoregressive decoder with a beam of size 10. We use all reference lexicalizations and the vocabulary size $V = 100$, following our preliminary experiments. We finetune the model for 10,000 updates with batch size 32 and learning rate 2×10^{-5}. For the beam filtering heuristic, we check for the presence of entities by simple string matching in WebNLG; for the E2E dataset, we use a set of regular expressions from Dušek et al. (2019). We process the triples in their default order.

Zero-shot Generation Additionally, we conduct a *zero-shot* experiment. We train the sentence fusion model with the same setup, but instead of the in-domain datasets, we use a subset of the *balanced-Wikipedia* portion of the DiscoFuse dataset (Geva et al., 2019). In particular, we use the discourse types frequently occurring in our datasets, filtering the discourse types irrelevant to our use case.

3.2.5 Results

Accuracy vs. Fluency On lexical similarity metrics (Table 3.6), our system lags behind the state-of-the-art approaches selected for comparison: the Semantic Fidelity Classifier (SFC; Harkous et al., 2020) and the finetuned T5 model (T5; Kale, 2020). However, both the fusion and the zero-shot approaches show improvements over the baseline. It is also important to note that our approach ensures zero entity errors: the entities are filled verbatim into the templates, and if an entity is missing in the whole beam, a fallback is used instead (although semantic inconsistencies can still occur, e.g., if a verb or function words are missing).

Triples	*(Albert Jennings Fountain, deathPlace, New Mexico Territory); (Albert Jennings Fountain, birthPlace, New York City); (Albert Jennings Fountain, birthPlace, Staten Island)*
Step #0	Albert Jennings Fountain died in New Mexico Territory.
Step #1	Albert Jennings Fountain, who died in New Mexico Territory, was born in New York City.
Step #2	Albert Jennings Fountain, who died in New Mexico Territory, was born in New York City, Staten Island.
Reference	Albert Jennings Fountain was born in Staten Island, New York City and died in the New Mexico Territory.

Table 3.6: An example of correct behavior of the algorithm on the WebNLG dataset. Newly added entities are underlined, the output from Step #2 is the output text.

Error Analysis The fused sentences in the E2E dataset, where all the objects are related to a single subject, often lean towards compact forms, e.g., *Aromi is a family friendly chinese coffee shop with a low customer rating in riverside.* On the contrary, the sentence structure in WebNLG mostly follows the structure from the templates, and the model makes minimal changes to fuse the sentences. See Table 3.6 for an example of the system output. Of all steps, 28% are fallbacks (no fusion is performed) in WebNLG and 54% in the E2E dataset. The higher number of fallbacks in the E2E dataset can be explained by a higher lexical variability of the references, together with a higher number of data items per example. This variability makes it harder for the model to maintain the text coherency over multiple steps.

Templates On average, there are 12.4 templates per predicate in WebNLG and 8.3 in the E2E dataset. In cases where the set of templates is more diverse, e.g., if the template for the predicate *country* has to be selected from {*<subject> is situated within <object>, <subject> is a dish found in <object>*}, LMScorer helps to select the semantically accurate template for the specific entities. The literal copying of entities can be too rigid in some cases, e.g., *Atatürk Monument (İzmir) is made of "Bronze".*

Zero-shot Experiments The zero-shot model trained on DiscoFuse is able to correctly pronominalize or delete repeated entities and join the sentences with conjunctives, e.g. *William Anders was born in British Hong Kong, and was a member of the crew of Apollo 8.* While the model makes only a limited use of sentence fusion, it makes the output more fluent while keeping strong guarantees of the output accuracy.

3.2.6 Discussion

Fixed Triple Order LaserTagger does not allow arbitrary reordering of words in the sentence, which can limit the output fluency. Grajcar (2023) expands upon our approach by using Felix, a text-editing model that is capable of arbitrary reordering of words in the sentence (Mallinson et al., 2020). As noted by Grajcar (2023), the order is indeed more flexible with Felix, but the quality of the outputs is still limited by the abilities of text-editing models.[11] In Section 3.3, we use an ordering module along with autoregressive PLMs, showing that the explicit ordering step leads to improvements in output quality.

Sentence Fusion A major issue with sentence fusion is deciding when to apply it. In our approach, we rely on the implicit knowledge of the model learned from in-domain training data, which often leads to outputs that are too compact. In Section 3.3, we thus introduce an *aggregation* module which explicitly decides which facts should be mentioned together in a sentence. We also note that the term *sentence fusion* is not accurate as the sentences are sometimes kept separate; this is why we opt for using a more generally applicable term *paragraph compression*.

Benefits of Our Approach Our system generates outputs that are suboptimal in fluency when compared with larger models. However, certain unique features make our approach attractive even nowadays. Firstly, the approach guarantees the presence of the entities in the output, which is not guaranteed by any approach relying on a language model (LM) in the final step. Our approach also helps with direct control over the generative process. For example, one can accept or reject the changes at each step or build a set of custom rules for individual edit operations on specific tokens. This possibility can be useful for fine-grained hallucination control (Rebuffel et al., 2021; Chen et al., 2023) and increasing the robustness of the model in a production system (Heidari et al., 2021; Wang et al., 2023d).

3.3 Pipeline of Text-to-Text Neural Modules

This section is based on the paper *Neural Pipeline for Zero-Shot Data-to-Text Generation* (Kasner and Dušek, 2022), joint work with Ondřej Dušek, published in the Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (ACL 2022).

[11]With increasing capabilities of autoregressive models, the main advantage of text-editing models is their speed, especially for tasks where only minor edits are needed (Malmi et al., 2022).

In this section, we further develop the approach from Section 3.2 for generating semantically accurate text from structured data. The main limitation of the previous approach, based on iterative transformations of simple templates, was the limited fluency of the output texts. To improve text fluency, we propose adding modules for ordering and aggregation, making the process a three-step pipeline. We also propose a way to make each of these steps trainable on a generic synthetic corpus. We confirm that on WebNLG and E2E datasets, our approach can get lower rates of omissions and hallucinations than prior approaches according to a semantic accuracy metric while achieving levels of lexical similarity comparable to some of the prior systems; all of this without the need for in-domain training data. Our code and data is available on Github.[12]

3.3.1 Motivation

Our experiments in Section 3.2 with iterative sentence fusion led to several observations:

(1) The fixed order of triples limits the expressivity of the model, leading to unnatural outputs.

(2) Using the sentence fusion model on every sentence boundary tends to produce sentences which are too compact.

(3) Text-editing models underperform state-of-the-art autoregressive models in terms of output quality.

Following these observations, we improve our approach by (1) inserting a triple-ordering step in the process, (2) replacing the sentence fusion with paragraph compression, and (3) basing the approach on trainable autoregressive models.

Our approach follows the idea of pipeline-based approaches (see Section 2.2.2). In particular, our pipeline follows the concept of pipelines based on iterative improvements of simple templates (Laha et al., 2019) and neural modules (Ferreira et al., 2019). We focus on the *ordering* and *aggregation* steps, which were shown to improve the quality of D2T generation outputs in domain-specific setups (Moryossef et al., 2019a,b; Trisedya et al., 2020; Su et al., 2021b).

[12] https://github.com/kasnerz/zeroshot-d2t-pipeline

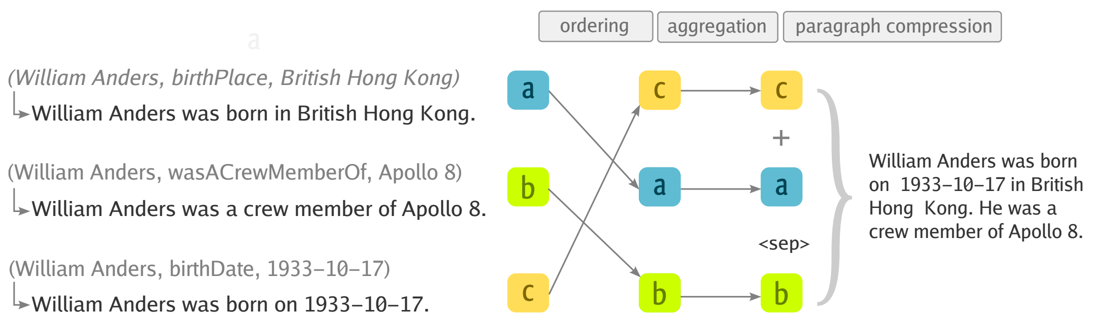

ordering
aggregation
paragraph compression
(William Anders, birthPlace, British Hong Kong)
William Anders was born in British Hong Kong.
(William Anders, wasACrewMemberOf, Apollo 8)
William Anders was a crew member of Apollo 8.
(William Anders, birthDate, 1933–10–17)
William Anders was born on 1933–10–17.
a
b
c
c
a
b
c
+
a
<sep>
b
William Anders was born on 1933–10–17 in British Hong Kong. He was a crew member of Apollo 8.

Figure 3.2: A scheme of our approach for zero-shot data-to-text generation from RDF triples. After a simple transformation of triples to facts, we apply the pipeline of modules for (1) ordering, (2) aggregation, (3) paragraph compression. Individual modules are trained on a large general-domain text corpus and operate over text in natural language.

In contrast to previous approaches, our pipeline is fully trainable on general-domain data, i.e., without using any training data from target D2T datasets. By eliminating the need for human references, we get rid remove of the costly and time-consuming data collection process. At the same time, we also avoid the brittleness of few-shot approaches, which are sensitive to the choice of finetuning examples (Chen et al., 2019; Su et al., 2021a; Chang et al., 2021a).

3.3.2 Method

Here, we provide a formal description of our approach. Similarly to Sections 3.1 and 3.2, we focus on the task of producing a natural language description Y a set of RDF triples $x \in X$, where each triple $x = (s, p, o)$ describes the relation p between the entities s and o in the knowledge graph.

Given a set of triples X on the input, we:

(1) transform the triples into *facts*, i.e., short sentences in natural language,

(2) sort the facts using an *ordering* module,

(3) insert sentence delimiters between the ordered facts using an *aggregation* module,

(4) input the ordered sequence of facts with delimiters into a *paragraph compression* module, which generates the final description Y.

In Sections 3.3.3 and 3.3.4, we show how to implement all these steps without the need for *any in-domain training data.*

Transforming Triples to Facts The first step in our pipeline involves transforming each of the input triples $x \in X$ into a fact $f \in F$ using a transformation $T : X \to F$. We define a fact f as a single sentence in natural language describing x. The transformation serves two purposes: (a) preparing the data for the subsequent text-to-text operations and (b) introducing in-domain knowledge about the semantics of individual predicates.

Ordering We assume that the default order of triples X is random. Note, however, that F is a set of meaningful sentences. We can use this to our advantage and apply a sentence ordering module (Barzilay et al., 2001; Lapata, 2003) to maximize the coherency of the paragraph resulting from their concatenation. The sentence ordering module $O(F)$ produces an ordered sequence of facts: $F_o = \{f_{o_1}, \ldots, f_{o_n}\}$, where $o_{1:n}$ is a permutation of fact indices. An example outcome of such operation may be grouping together facts mentioning *birth date* and *birth place* of a person, followed by their *occupation*, as it is shown in Figure 3.2. The ordering module allows downstream modules to focus only on operations over neighboring sentences.

Aggregation Some facts will be typically mentioned together in a single sentence. Considering the previous example, *occupation* is likely to be mentioned separately, while *birth date* and *birth place* are likely to be mentioned together. We make these decisions using the aggregation module, which takes a sequence of ordered facts F_o as input and produces a sequence of sentence delimiters $A(F_o) = \{\delta_{o_1}, \delta_{o_2}, \ldots, \delta_{o_{n-1}}\}$; $\delta_i \in \{0, 1\}$. Unlike previous works (Wiseman et al., 2018; Shao et al., 2019; Shen et al., 2020; Xu et al., 2021), which capture the segments corresponding to individual parts of the input as latent variables, we simply insert delimiters into the ordered sequence of facts to mark sentence boundaries. The output $\delta_i = 0$ means that the facts should be aggregated, and their corresponding sentences should be fused. Note that the markers serve only as a hint for the paragraph compression module, i.e., the
actually yet
sentences are not fused in this step.

Paragraph Compression The paragraph compression (PC) module takes as input the ordered sequence of facts with delimiters $F_a = \{f_{o_1}, \delta_{o_1}, f_{o_2}, \ldots, \delta_{o_{n-1}}, f_{o_n}\}$ and produces the final text Y. It has two main objectives: (a) *fusing* related sentences, i.e., sentences i and j in between which $\delta_i = 0$, and (b) *rephrasing* the text to improve its fluency, e.g., fixing disfluencies in the templates or replacing noun phrases with referring expressions. Unlike in text summarization or sentence simplification, the edits will typically be minor since we aim to preserve the semantics of the text.

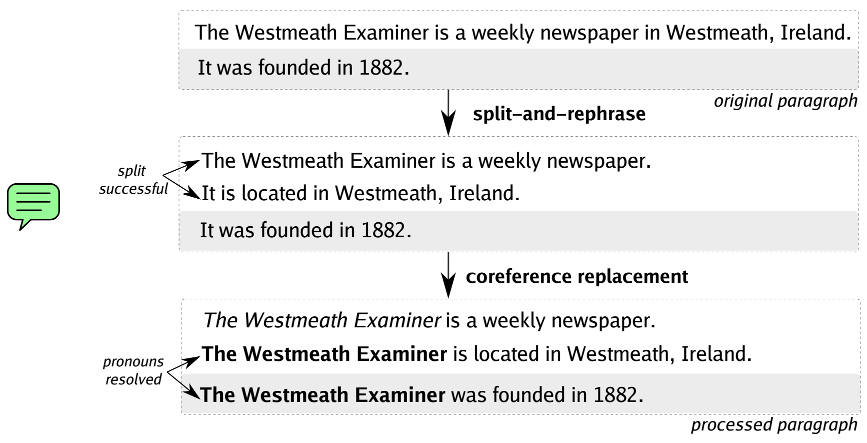

The Westmeath Examiner is a weekly newspaper in Westmeath, Ireland.
It was founded in 1882.
original paragraph
split-and-rephrase
split successful
The Westmeath Examiner is a weekly newspaper.
It is located in Westmeath, Ireland.
It was founded in 1882.
coreference replacement
The Westmeath Examiner is a weekly newspaper.
pronouns resolved
The Westmeath Examiner is located in Westmeath, Ireland.
The Westmeath Examiner was founded in 1882.
processed paragraph

Figure 3.3: The building process of the WikiFluent corpus. We apply a split-and-rephrase model on each sentence in the paragraph and resolve coreferences in the split sentences. The result is a set of simple sentences that convey the same meaning as the original paragraph. The synthesized sentences are used as *input* in our models; the original human-written texts are used as *ground truth*.

3.3.3 WikiFluent Corpus

For training the modules, we need to build a corpus in which where (1) the input is a set of simple, template-like sentences, (2) the output is a fluent text in natural language preserving the semantics of the input. Here, we propose a way ~~how~~ to build such a large-scale synthetic corpus from English Wikipedia. Our resulting corpus (WikiFluent) is orders of magnitude larger than ~~the~~ in-domain D2T datasets and provides training data for all the modules in our pipeline.

Data Source For building the WikiFluent corpus, we first extracted 934k first paragraphs of articles from a Wikipedia dump[13] using WikiExtractor (Attardi, 2015). Wikipedia is commonly used for large-scale pretraining of D2T generation models, as it provides a source of neutral texts based on factual data (Jin et al., 2020; Chen et al., 2020b). We used the first paragraphs of Wikipedia entries with lengths between 30-430 characters, filtering out lists, disambiguations, and malformed paragraphs. To balance the lengths of inputs, we divided the paragraphs according to their length into four equally-sized bins (30-130 characters, etc.) and selected 250k examples from each bin.

[13] `enwiki-20210401-pages-articles-multistream`

	#train	#dev	#test	tok/src	tok/tgt	sent/src	sent/tgt
WebNLG	18,102	870	1,862	26.8	22.6	3.0	1.4
Clean E2E	33,236	4,299	1,847	29.2	22.3	4.2	1.5
WikiFluent-*full*	915,855	9,346	9,346	52.9	41.1	3.9	2.0
WikiFluent-*filtered*	700,517	7,149	7,149	45.6	35.4	3.4	1.8

Table 3.7: Number of examples (train / dev / test), average number of tokens per source and target, average number of sentences per source and target (after filling the templates for the D2T datasets), total number of templates.

Split-and-Rephrase Split-and-rephrase is the task of splitting a complex sentence into a sequence of shorter sentences preserving the original meaning (Narayan et al., 2017). We train[14] BART-base (Lewis et al., 2019) for the split-and-rephrase task on the WikiSplit corpus, containing human-made sentence splits from Wikipedia edit history (Botha et al., 2018). We split each paragraph into sentences using NLTK (Bird, 2006) and apply the split-and-rephrase model to each sentence. To ensure that the splits are not deterministic, we choose uniformly randomly choosing between 0-2 recursive calls. If the sentence cannot be meaningfully split, the model tends to duplicate the sentence on the output; in that case, we use only the original sentence and do not proceed with ~~the~~ any splitting.

Coreference Replacement Unlike D2T templates, ~~that~~ which always mention a single fact, the split sentences heavily use referring expressions. To match the style of the sentences we apply a coreference resolution model (Lee et al., 2018) from the AllenNLP framework (Gardner et al., 2018). We replace referring expressions with their antecedents (e.g., pronouns with noun phrases). Note that we replace the referring expressions only in the synthesized sentences, not in the original paragraphs, so that the paragraph compression module is later implicitly trained to generate referring expressions in the final description.

Filtering To ensure that the generated sentences convey the same semantics as the original paragraph, we use the RoBERTa model[15] (Liu et al., 2019b) finetuned on the MultiNLI dataset (Williams et al., 2018) for checking the semantic accuracy of the generated text. Following Dušek and Kasner (2020) (see Section 4.1), we test if the original paragraph entails each of the synthesized sentences (checking for omissions)

[14]Following the same setup as for a paragraph compression model (Section 3.3.4).
[15]`https://huggingface.co/roberta-large-mnli`

and if the set of concatenated synthesized sentences entails the original paragraph (checking for hallucinations). In a filtered version of the WikiFluent corpus, we include only the examples without omissions or hallucinations (as computed by the model), reducing it to 714k examples (approximately 75% of the original size).

3.3.4 Implementation

This section describes how we implement our pipeline using simple template transformations and neural models trained on the WikiFluent dataset.

Templates We transform triples into facts using a single-triple template t_i for each predicate, analogically to our approach in Section 3.2.3. Compared to more complex rule-based template generation engines (Laha et al., 2019; Heidari et al., 2021; Mehta et al., 2021), the approach minimizes manual workload and makes it easier to control the quality of the input for the subsequent steps.

Ordering Model For our ordering model, we use the *Simple Pointer* model from Calizzano et al. (2021)[16]. The model is based on a pretrained BART-base model (Lewis et al., 2019) extended with a pointer network from Wang and Wan (2019). We train the model using the synthesized simple sentences in the WikiFluent corpus, randomly shuffling the order of the sentences and training the model to restore their original order.

Aggregation Model We base our aggregation model on RoBERTa-large (Liu et al., 2019b) with a token classification head. We input the sequence of ordered facts F_o into the model, separating each pair of facts f_{o_i} with a separator token. The token classification layer classifies each separator token into two classes $\{0, 1\}$ corresponding to the delimiter δ_i. We ignore the outputs for the non-separator tokens while
for aggregation
computing cross-entropy loss. We create t~~he~~ training examples using the synthesized sentences in the WikiFluent corpus, in which we set $\delta_i = 0$ for the sentences $i, i+1$ which are the result of splitting a single sentence and $\delta_i = 1$ otherwise.

[16]For details about the model, please refer to Calizzano et al. (2021).

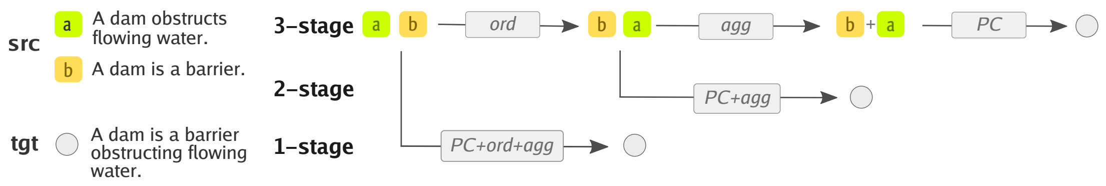


Figure 3.4: An example illustrating how the individual modules are trained and subsequently applied as the parts of the pipeline. See Section 3.3.4 for the description of the ordering model (ORD), the aggregation model (AGG), and the variants of the paragraph compression model (PC, PC+AGG, PC+ORD+AGG).

Paragraph Compression Model For the paragraph compression model, we finetune BART-base (Lewis et al., 2019) on the WikiFluent corpus, concatenating the synthesized sentences on the input, we add delimiters between the sentences i and $i+1$ where $\delta_i = 1$ using a special token `<sep>`, which we add to the model vocabulary. We expect that the model learns to fuse the sentences between which there are no delimiters on the input. We evaluate how the model learns to respect the order and aggregation markers in Section 3.3.6.

3.3.5 Experiments

We train our pipeline modules on the WikiFluent corpus as described in Section 3.3.4. Next, we use these modules *without any further finetuning* for generating descriptions for RDF triples on the WebNLG and E2E datasets.

Pipeline versions To evaluate individual components of our pipeline, we train three versions of the *paragraph compression* model (see Figure 3.4). The models share the same architecture and targets but differ in their inputs:

- PC – the model takes as an input ordered facts with delimiters (as described in Section 3.3.2),
- PC+AGG – the model takes as an input the ordered facts *without* delimiters (i.e., the aggregation is left implicitly to the model),
- PC+ORD+AGG – the model takes as an input the facts in *random* order and *without* delimiters (i.e., both ordering and aggregation are left implicitly to the model).

Correspondingly, we test three versions of the pipeline for the ablation study:

- 3-STAGE – a full version of the pipeline consisting of the ordering model (ORD), the aggregation model (AGG), and the PC model,
- 2-STAGE – a pipeline consisting of the ORD model and the PC+AGG model,

Input	*(Allen Forrest; background; solo singer), (Allen Forrest; genre; Pop music), (Allen Forrest; birthPlace; Dothan, Alabama)*
Templ.	Allen Forrest is a solo singer. Allen Forrest performs Pop music. Allen Forrest was born in Dothan, Alabama.
Model	Allen Forrest is a solo singer who performs Pop music. He was born in Dothan, Alabama.
Human	Born in Dothan, Alabama, Allen Forrest has a background as a solo singer and was a pop artist.
Input	*name[Wildwood], eatType[restaurant], food[French], area[riverside], near[Raja Indian Cuisine]*
Templ.	Wildwood is a restaurant. Wildwood serves French food. Wildwood is in the riverside. Wildwood is near Raja Indian Cuisine.
Model	Wildwood is a restaurant serving French food. It is in the riverside near Raja Indian Cuisine.
Human	A amazing French restaurant is called the Wildwood. The restaurant is near the Raja Indian Cuisine in riverside. They [illegible] kids.

Table 3.8: Example outputs of our model (3-STAGE, filtered). For each example, we show the input triples, the intermediate templates, the output from the model, and the corresponding human reference.

- 1-STAGE – a single stage consisting of the PC+ORD+AGG model.

3.3.6 Evaluation

We evaluate outputs from the {1,2,3}-STAGE variants of our pipeline using automatic metrics, and we perform a detailed manual error analysis of the model outputs. We also evaluate the performance of the content planning modules and the ability of the PC module to follow the content plan. Finally, we include an intrinsic evaluation of our modules on the WikiFluent test set.

Automatic Metrics Following prior work, we use BLEU (Papineni et al., 2002) and METEOR (Banerjee and Lavie, 2005) to evaluate the outputs against the human references.[17] We also evaluate the number of omission and hallucination errors (i.e., facts missing or added, respectively) using a metric from Dušek and Kasner (2020) based on a RoBERTa model (Liu et al., 2019b) pretrained on natural language inference (see Section 4.1.)

We include a diverse set of baselines for comparison. Copy denotes the baseline of copying the facts without further processing. For WebNLG, we further compare our systems with the results of:

- UPF-FORGe and Melbourne – systems (grammar-based and supervised, respectively) from the first run of WebNLG Challenge (Gardent et al., 2017b),

[17]We use the implementation from `https://github.com/tuetschek/e2e-metrics`.

		WebNLG				E2E			
		B	**M**	**O**	**H**	**B**	**M**	**O**	**H**
Copy		37.18	38.77	0.000	0.000	24.19	34.89	0.000	0.000
UPF-FORGe*		38.65	39.00	0.075	0.101	-	-	-	-
Melbourne*		45.13	37.00	0.237	0.202	-	-	-	-
Ke et al. (2021)†*		66.14	47.25	-	-	-	-	-	-
Laha et al. (2019)†		24.80	34.90	-	-	-	-	-	-
TGen*		-	-	-	-	40.73	37.76	0.016	0.083
Harkous et al. (2020)†*		-	-	-	-	43.60	39.00	-	-
full	3-stage	42.92	39.07	0.051	0.148	**36.04**	36.95	**0.001**	**0.001**
	2-stage	42.90	39.28	**0.043**	0.125	35.84	36.91	**0.001**	**0.001**
	1-stage	39.08	38.94	0.071	0.204	30.81	36.01	0.009	0.122
filtered	3-stage	43.19	39.13	0.152	**0.073**	35.88	36.95	**0.001**	**0.001**
	2-stage	**43.49**	**39.32**	0.146	0.096	36.01	**36.99**	**0.001**	**0.001**
	1-stage	42.99	38.81	0.202	0.093	34.08	36.32	0.012	0.050

Table 3.9: Automatic metrics on the WebNLG and E2E datasets. B = BLEU, M = METEOR, O = omissions / # facts, H = hallucinations / # examples. The systems marked with asterisk (*) are trained on ~~the~~ in-domain data. The results for the systems marked with † are taken from the respective works. **Boldface** denotes the best variant of our zero-shot system.

- Ke et al. (2021) – a state-of-the-art system with a structure-aware encoder and task-specific pretraining,
- Laha et al. (2019) – a zero-shot D2T generation system.

For E2E, we compare our systems with the results of:

- TGen (Dušek and Jurcicek, 2015) – the baseline system for the E2E Challenge (Dušek et al., 2020),
- Harkous et al. (2020) – a state-of-the-art supervised system on the cleaned version of E2E data.

The automatic evaluation (Table 3.9) shows that our systems consistently outperform the Copy baseline (e.g., ~12 BLEU points for E2E), which is already strong thanks to our manually curated set of templates.[18] Automatic scores also suggest that our systems are comparable with some older supervised systems, although they underperform the state-of-the-art supervised systems.

[18]On WebNLG, Copy achieves 37.18 BLEU points, compared to 24.80 BLEU points of the *full system* of Laha et al. (2019), which uses automatic template generation.

		WebNLG					**E2E**				
		H	**I**	**O**	**R**	**G**	**H**	**I**	**O**	**R**	**G**
full	3-STAGE	3	39	2	2	16	0	1	0	0	17
	2-STAGE	8	36	1	5	16	1	1	0	1	23
	1-STAGE	28	27	6	10	20	17	0	1	79	45
filtered	3-STAGE	2	37	2	1	15	0	0	0	0	17
	2-STAGE	5	32	1	2	14	0	0	0	0	11
	1-STAGE	8	40	6	6	16	11	2	1	41	22

Table 3.10: Number of manually annotated errors on 100 examples: H = hallucinations, I = incorrect fact merging, O = omissions, R = redundancies, G = grammar errors or disfluencies.

The 2-STAGE system is generally on par with the 3-STAGE system, which indicates that explicit aggregation using the AGG model may not be necessary. However, a separate aggregation module allows one to control the aggregation step explicitly. The models using the filtered version of the corpus generally produce better results, although they also bring in a larger number of omissions.

Manual Error Analysis We manually examined 100 model outputs, counting the number of factual errors (hallucinations, omissions, incorrect fact merging, redundancies) and grammatical errors. The results are summarized in Table 3.10.

The 1-STAGE model (which has to order the facts implicitly) tends to repeat the facts in the text (especially in E2E) and produces frequent hallucinations. These problems are largely eliminated with the 2-STAGE and 3-STAGE models, which produce almost no hallucinations or omissions.

However, the outputs on WebNLG for all systems suffer from semantic errors resulting from merging unrelated facts. This mostly happens with unrelated predicates connected to the same subject/object (e.g., “X was born in Y”, “X worked as Z” expressed as “X worked as Z in Y”). On the E2E data, where predicates share the same subject, the outputs are generally consistent, and the 2-STAGE and 3-STAGE models exhibit almost no semantic errors. Grammar errors and disfluencies stem mainly from over-eager paragraph compression or from artifacts in our templates and are relatively minor (e.g., missing “is” in “serves French food and family-friendly”).

Content Planning We manually evaluate how the PC model follows the content plan (i.e., keeping the predefined order and aggregating the sentences according to the delimiters) using 100 randomly chosen examples with more than one triple on WebNLG and E2E. We find that the model follows the content plan in 95% and 100% of cases, respectively. The incorrect cases include mainly a fact not properly mentioned or an extra boundary between sentences without a separator. We can thus conclude that the pretraining task successfully teaches the PC model to follow a given content plan.

3.3.7 Discussion

Ordering and Aggregation As we have shown, reducing the task to fusing neighboring sentences (by pre-ordering the triples with a dedicated module) makes the model less prone to producing omissions or hallucinations. The claim may hold even with larger language models, although additional experiments are needed to confirm it. As shown in Su et al. (2021c), another advantage of having an explicit order is that we can manually change the order to control the output on a fine-grained level. We have also shown that explicit aggregation may not necessarily improve output fluency, although it can still help with more explicit controllability.

Possible Extensions Beyond generating factual information from English knowledge graphs, one can imagine applying the approach to more complex cases of factual D2T generation, for example by prepending a context selection module for table-to-text generation (Parikh et al., 2020; Cheng et al., 2021) or by using templates for logical formulas in logical table-to-text generation (Chen et al., 2020a,c). That being said, we are unaware of a follow-up work that would go in this direction, perhaps due to the harder scalability of such a modularized system. For applying the approach to other languages, a go-to approach would be using the respective language edition of Wikipedia to train a sentence splitting model along the lines of Botha et al. (2018), followed by building the respective language variant of the WikiFluent corpus.

Can we do without the templates? Hand-crafting the templates as the first step ~~somehow~~ *somewhat* lessens the advantages of our data-driven approach. For this reason, we show in Section 6.1 that we *can replace this step* with a model finetuned on a suitable dataset and get better results on lexical similarity metrics. Alternatively, we can also replace this step by prompting ~~a~~ *an* LLM, as shown in Xiang et al. (2022) and Saha et al. (2022). In this case, however, special care needs to be ~~given~~ *taken* to ensure a consistent style of the model outputs.

3.4 Conclusion

We introduced three approaches for data-driven D2T generation focused on generating descriptions of a knowledge graph. In Section 3.1, we showed that finetuning of a PLM can achieve satisfactory results with minimal effort given a moderate amount of in-domain training data. Subsequently, we identified the shortcomings of such an approach, which we addressed in Sections 3.2 and 3.3. First, we focused on improving the need for costly in-domain training data. We reduced the amount of data necessary in Section 3.2 by limiting the model vocabulary and eliminated the need for in-domain data completely in Section 3.3 by using general-domain training data. Second, we focused on improving the semantic accuracy of the system: we ensured the presence of entities in the output (Section 3.2) and showed how to balance semantic accuracy with fluency (Section 3.3). We also discussed the benefits of our approaches in the light of recent progress.

4 Evaluating Semantic Accuracy

This chapter introduces two approaches for evaluating the semantic accuracy of data-to-text (D2T) generation. As discussed in Section 2.2.7, it is essential that the texts based on the input data are *faithful to the data*, i.e., semantically accurate. Yet, there is a lack of automatic metrics to assess the semantic accuracy of generated texts. The metrics we introduce are based on pretrained language models (PLMs) and generic text-to-text operations, which makes our approaches applicable to various tasks and domains.

In Section 4.1, we first describe a metric suitable for D2T generation tasks where all the facts in the input data should be mentioned. The metric is based on an off-the-shelf PLM for natural language inference (NLI), which we repurpose for detecting omissions and hallucinations in the output text. We show that our metric correlates well with human judgments, and that in some cases, the metric even provides judgments that are more accurate.

In Section 4.2, we focus on detecting factual errors in D2T generation from complex tabular data. We propose a metric based on a PLM-based tagger, which can mark individual tokens with fine-grained error categories. To provide relevant information for the tagger, we combine it with a rule-based generator and a neural-based retriever. This The metric ranked first out of four automatic metrics in the Shared Task on Evaluating Accuracy in Generated Texts cite

4.1 Detecting Omissions and Hallucinations

This section is based on the paper *Evaluating Semantic Accuracy of Data-to-Text Generation with Natural Language Inference* (Dušek and Kasner, 2020), joint work with Ondřej Dušek, published in the Proceedings of the The 13th International Conference on Natural Language Generation (INLG 2020). The experimental part was done by Ondřej Dušek; the author of this thesis came up with the initial idea and wrote the paper. The paper has received the award for the best short paper at INLG 2020.

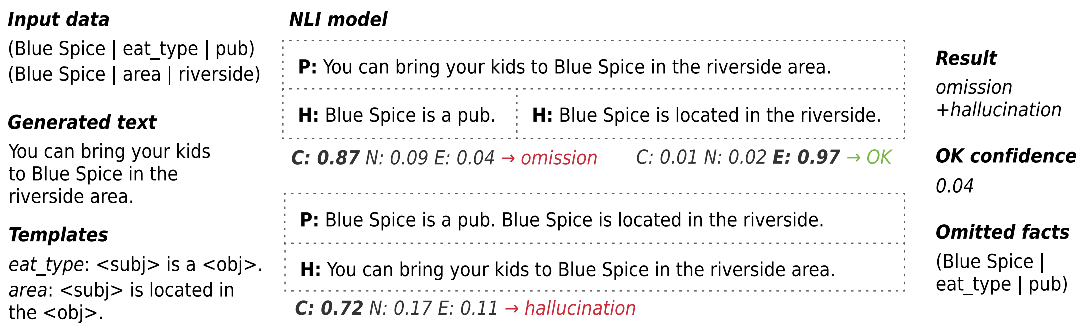
In this section, we propose a new metric for evaluating the semantic accuracy of D2T generation. Our metric is based on a neural model pretrained for NLI. We use the NLI model to check textual entailment between the input data and the output text in both directions, allowing us to reveal omissions or hallucinations. We demonstrate that even without any extra model training and with minimal handcrafting, our approach achieves high accuracy (>90%) on the E2E dataset, competitive with scripts specifically handcrafted for the domain, and produces useful results (>75% accuracy) on the more challenging WebNLG dataset. Additionally, we show with manual error analysis that some instances marked as errors were in fact assessed correctly by our metric. The experimental code for our metric is available on GitHub.[1]

4.1.1 Motivation

In Section 2.2.7, we described two ways ~~how~~ in which the semantic accuracy ~~of the text~~ NLG outputs can be compromised: the text may be missing some data (*omission*) or contain extra information not supported by the data (*hallucination*). Since state-of-the-art neural D2T generation models are prone to both of these (Gehrmann et al., 2018; Ferreira et al., 2019; Dušek et al., 2020), recognizing the violations of semantic accuracy is essential for proper system evaluation and further development. However, it is difficult for handcrafted heuristics to cover all edge cases, as minor changes in wording may cause major differences in the meaning of the text. Human evaluation, on the other hand, is expensive and difficult to scale.

We note that if we transform individual data items to short sentences (*facts*), we can check whether each ~~sentence~~ fact is entailed by the generated text. Specifically, if we find that ~~the sentence~~ a given fact is not entailed by the generated text, we can report an *omission* of the corresponding data item. Vice versa, if we concatenate all the facts and these

[1] https://github.com/ufal/nlgi_eval



Input data
(Blue Spice | eat_type | pub)
(Blue Spice | area | riverside)
Generated text
You can bring your kids to Blue Spice in the riverside area.
Templates
eat_type: <subj> is a <obj>.
area: <subj> is located in the <obj>.
NLI model
P: You can bring your kids to Blue Spice in the riverside area.
H: Blue Spice is a pub.
H: Blue Spice is located in the riverside.
C: 0.87 N: 0.09 E: 0.04 → omission
C: 0.01 N: 0.02 E: 0.97 → OK
P: Blue Spice is a pub. Blue Spice is located in the riverside.
H: You can bring your kids to Blue Spice in the riverside area.
C: 0.72 N: 0.17 E: 0.11 → hallucination
Result
omission +hallucination
OK confidence
0.04
Omitted facts
(Blue Spice | eat_type | pub)

Figure 4.1: An example of evaluating the output from a D2T system with our metric. The generated text is used as a *premise* (P) to check for omissions and as a *hypothesis* (H) to check for hallucinations. The NLI model generates probabilities for *contradiction* (C), *neutral* (N) and *entailment* (E).

do not entail the generated text, we can report a *hallucination.* For this approach, we need only two ingredients: (1) a way to convert individual data items to facts and (2) a model that can assess if one sentence entails another. We formalize our approach in the next section.

4.1.2 Method

We are given a set of RDF triples $x \in X$, where each triple $x = (s, p, o)$ describes the relation p between the entities s and o, and the corresponding natural language description Y. Our task is to assess whether Y mentions all the triples in X. Additionally, we should also check whether the text mentions any extra information.

Data Preprocessing Throughout Chapter 3, we have used simple templates for transforming individual triples to facts, i.e., simple sentences capturing the triple meaning. We use the same method here, considering two cases:

(1) *Default:* We use a specific template for each predicate, using templates that are either handcrafted or extracted from the NLG systems' training data.

(2) *Backoff:* We use only a single, universal "backoff" template for all the facts, in the form: *The <predicate> of <subject> is <object>.*

Natural Language Inference NLI is a sequence classification task that takes two inputs—a *hypothesis* and a *premise*—and produces one of the possible outputs: the hypothesis is *entailed* by (follows from) the premise, *contradicts* the premise, or their relation is *neutral.* Neural models for NLI (Zhang et al., 2020; Liu et al., 2019a,b) have already reached near-human levels of performance, making them suitable for evaluating the output of abstractive summarization systems (Maynez et al., 2020).

Checking Semantic Accuracy with NLI We can use an NLI model for assessing semantic accuracy of generated texts. Consider the two input facts from Figure 4.1: F ={*"Blue spice is a pub"*, *"Blue Spice is located in the riverside"*} and the generated text: Y =*"You can bring your kids to Blue Spice in the riverside area."* We propose using an NLI model for checking if the semantic information implied by F and Y is equal. In this case, the model should detect that the first fact is not entailed by the text (there is no mention of Blue Spice being a pub), but the text is also not entailed by the facts (the information about kids is hallucinated).

We achieve this by using the NLI model to check for entailment in two directions:

(1) To check for omissions, we use the whole generated text as a premise and sequentially feed each fact as a hypothesis to the NLI model. Any failed NLI check is considered an omission.

(2) To check for hallucinations, we concatenate all facts as a premise and feed the generated text as a hypothesis to the NLI model. If this NLI check fails, the text is considered to contain hallucination. This step cannot be split into simpler NLI checks.

The final output of our metric is either 4-way (denoted as Fine) or 2-way (denoted as Rough):

- Fine: We output the probabilities of 4 categories: *OK* (i.e., all NLI checks passed), *omission*, *hallucination*, or *omission+hallucination* (based on the failed checks). The 4-way output is more useful for system evaluation since we can distinguish whether the system tends to hallucinate or omit information.
- Rough: The three failure modes are combined into *not_OK*. The 2-way output corresponds more to usage inside an D2T generation system for output reranking or filtering, where any incorrect *output* should be penalized or filtered out.

Additionally, we compute a *confidence score* of the model as the minimum of all the entailment probabilities.

4.1.3 Experiments

For our NLI model, we use the `roberta-large-mnli`[2] checkpoint of the pretrained RoBERTa model (Liu et al., 2019b), which was finetuned on the MultiNLI dataset (Williams et al., 2018). We use the model *as is*, without any further training on domain-specific data. Given a premise text and a hypothesis text, the NLI model produces a probability distribution over three results: *contradiction*, *neutral*, and *entailment* (see Section 4.1.2). We consider a NLI check as passed if the probability for *entailment* is the highest of the three.

We experiment with the WebNLG and E2E datasets (see Section 2.2.6 for the descriptions of the dataset[illegible]ince both datasets were used in shared tasks, we can compare the outputs of our system with the respective measures of semantic accuracy:

- For WebNLG, we compare our metric with crowdsourced human ratings of semantic adequacy (Shimorina et al., 2019). In particular, we use [illegible] question: *"Does the text correctly represent the meaning in the data?"*, where the human annotators used a three-point Likert scale (1 = Incorrect, 2 = Medium, 3 = Correct). The answers are averaged over multiple annotators. In our experiments, we consider a sentence correct if it achieved a human rating 2.5 or higher.[3]

- For E2E, we compare our metric to the results of the handcrafted automatic script which was used in the E2E challenge (Dušek et al., 2020).[4] We further use small sets of system outputs and human-written texts with expert annotation (provided by Dušek et al., 2019) to evaluate our approach against gold-standard annotation and to compare to existing semantic accuracy classifiers for E2E data.

We evaluate the *Default* and *Backoff* approaches to acquiring templates as described in Section 4.1.2. For WebNLG, we obtained templates by delexicalizing human references for single-triple examples from the WebNLG training data.[5] [illegible]r E2E, we handcrafted eight templates for each predicate in the dataset.

	WebNLG					E2E				
	A	**R**	**P**	**F1**	ρ	**Af**	**Ar**	**R**	**P**	**F1**
Default	0.775	0.772	0.796	0.784	0.628	0.911	0.933	0.895	0.910	0.903
Backoff	0.768	0.760	0.793	0.776	0.637	0.846	0.874	0.913	0.768	0.834

Table 4.1: WebNLG and E2E results, compared to crowdsourced human ratings and the automatic evaluation script, respectively (A = accuracy, Af = FINE accuracy, Ar = ROUGH accuracy, R = recall, P = precision, F1 = F-measure, ρ = Spearman correlation of confidence scores with human scores).

4.1.4 Evaluation

We evaluate our metric in terms of accuracy, precision, recall, and F1-measure (where *not_OK* samples are treated as positive since we focus on detecting errors). The overall scores for both datasets are summarized in Table 4.1. We additionally perform a manual error analysis on a random sample of 100 error examples for each dataset, i.e., examples where our metric gave a different assessment from the ground truth.

WebNLG Analysis The Spearman correlation of our model's confidence scores with the average human scores on the WebNLG dataset is moderate (around 63%; p <1e-10). Our manual error analysis indicates several potential sources of discrepancies:

(1) The human annotation is somewhat noisy—many correctly rendered outputs do not reach the 2.5 threshold, while some incorrect ones do.

(2) The human annotators also tend to give lower scores to accurate but ungrammatical or poorly organized texts, while our metric tends to rate these texts as *OK*.

(3) Imprecise templates can confuse the NLI (e.g., for the predicate *nationality*, our extracted template is *<subj> was <obj>*, which works well with values such as *French*, but not with *United States*). This is a weak point of our metric, as illustrated by the very small performance difference between the *Default* and *Backoff* setups. However, the issue can be mitigated by a better selection of the templates from training data, e.g., using language-model scoring.

[2] `https://huggingface.co/roberta-large-mnli`

[3] We also tried a threshold of 2.0, with slightly worse results.

[4] While the E2E challenge did include crowdsourced evaluation of semantic accuracy, the results were unreliable, overestimating the number of errors (Dušek et al., 2020).

[5] For each predicate, we choose randomly if more templates are found and use the backoff if no templates are found.

Moreover, our re-examination shows that almost half of the error examples (42 out of 100) were in fact correctly classified by our metric (i.e., their crowdsourced human annotation was incorrect), so the true performance is most likely higher than the reported numbers.

E2E Analysis The results for the E2E dataset are very good compared to the WebNLG dataset, with over 90% agreement with the handcrafted script. This can be attributed to lower lexical variability and less noisy texts, as well as to the better quality of the handcrafted templates (the difference between the *Default* and *Backoff* setups is much more pronounced here). The main issues identified by our error analysis are:

(1) Problems in the interpretation of some values, e.g., *price range=less than £20* is verbalized as "cheap" or *family-friendly=no* as "adult-only". These cases are classified as *not_OK* by the NLI model.

(2) Missing or over-greedy patterns in the slot error script, causing annotation errors.

(3) Edge cases: some expressions cannot be interpreted in a straightforward way, e.g., "high restaurant" for *pricerange=high* is deemed OK by the NLI but not by the slot error script.

(4) Expressions in the outputs that do not correspond to input facts, such as "with full service", are considered hallucinations by the NLI but ignored by the slot error script.

Again, we consider about half of the error examples (45 out of 100) as correctly classified by our metric, and thus our metric's performance is probably higher than the reported values.

4.1.5 Discussion

Comparison to Other Metrics The main alternatives to our metric are mostly reference-based, which makes their use-cases different from ours (Zhao et al., 2019; Sellam et al., 2020; Yuan et al., 2021). Except for PARENT (Dhingra et al., 2019) and RoMe (Rony et al., 2022), these metrics are not targetting D2T generation. Our metric thus remains a major alternative for evaluating the semantic accuracy of data-to-text generation, especially with RDF triples on the input. ~~In the future,~~ a more flexible metric for evaluating semantic accuracy can be potentially based on large language models (Zhao et al., 2023b; Sottana et al., 2023; Kocmi and Federmann, 2023b), a topic to which we return in Section 6.2.

Limitations Perhaps surprisingly, the main bottleneck of the metric is not in the capabilities in NLI model. Although the NLI model is not perfect, Chen and Eger (2022) have shown that out-of-the-box NLI models are generally better and more robust metrics than specially trained approaches. In many cases, however, converting the structured data to a format suitable to PLM can be non-trivial. In this respect, see the discussion on automating template generation with PLMs and large language models (LLMs) in Section 3.3.7.

4.2 Token-Level Error Classification

This section is based on the paper *Text-in-Context: Token-Level Error Detection for Table-to-Text Generation* (Kasner et al., 2021), joint work with Simon Mille and Ondřej Dušek, published in the Proceedings of the 14th International Conference on Natural Language Generation (INLG 2021). The work describes our submission to the Shared Task on Evaluating Accuracy in Generated Texts. The project was led by the author of the thesis; Simon Mille provided the rule-based generator and wrote its description.

In this section, we present an automatic metric for D2T generation that can detect semantic accuracy errors in the generated text on the *token level*. Our metric combines a rule-based D2T generation system and PLMs. We first use a rule-based D2T generation system to generate all the facts that can be derived from the input. For each sentence we evaluate, we select a subset of relevant facts by measuring their semantic similarity with the examined sentence. For annotating erroneous tokens, we finetune a pretrained language model for token-level classification, using the annotated data with the relevant facts in the context as the ground truth.

On the test set of the Shared Task on Evaluating Accuracy in Generated Texts (Thomson and Reiter, 2021), we achieve 69% recall and 75% precision with a model trained on a mixture of human-annotated and synthetic data, placing first out of four submissions in the track for automatic metrics. The code for our experiments is available on Github.[6]

4.2.1 Motivation

In Section 4.1, we presented a metric for detecting semantic errors for D2T generation at the level of individual data items. The metric is well-suited for cases where the text should mention *all the data on the input*, as it can report individual missing items (omissions). However, it is less suitable for detecting incorrect information in the text

[6] https://github.com/kasnerz/accuracySharedTask_CUNI-UPF

(hallucinations), as it can give only a single "hallucination score" for the entire text. This is problematic for the texts generated from complex data, where the omissions are not relevant (since we do not verbalize all the input data), but the system can still produce numerous hallucinations.

An example of a dataset with complex data is Rotowire (Wiseman et al., 2017; see Section 2.2.6 for details). In this dataset, the task is to generate basketball match summaries from tabular data. Rotowire poses multiple challenges for neural systems, including the fact that it requires content selection or that its human-written training texts are themselves not always grounded in data, which makes neural models more susceptible to hallucinatio[illegible]he output texts are also usually longer, which makes the hallucinations more common and detecting hallucination errors on a more fine-grained level essential.

There is, however, no established way to check for hallucinations automatically. Specific content-checking metrics mostly remain a domain of handcrafted pattern matching (Wen et al., 2015b; Dušek et al., 2019), which does not scale well to new domains. While human evaluation provides a more reliable alternative, it is costly and difficult to set up (van der Lee et al., 2019; Belz et al., 2020; Thomson and Reiter, 2020). Regarding neural metrics such as BERTScore (Zhang et al., 2019) or BLEURT (Sellam et al., 2020), most of them have not been specifically evaluated (and targeted) for content preservation, especially on the level of individual tokens.

4.2.2 Shared Task in Evaluating Accuracy

The goal of the Shared Task on Evaluating Accuracy in Generated Texts at INLG 2021 was to develop a token-level error annotation metric for complex D2T generation (Reiter and Thomson, 2020). The organizers of the shared task first manually annotated 60 outputs of various neural systems trained on Rotowire, using the error types defined in Thomson and Reiter (2020):

- NUMBER^{N} – Incorrect number (both digits and numerals).
- NAME^{E} – Incorrect named entity (people, places, teams, days of the week).
- WORD^{W} – Incorrect word which is not one of the above.
- CONTEXT^{C} – A phrase inappropriate for the context.
- NOT_CHECKABLE^{NC} – A statement which cannot be checked.
- OTHER^{O} – Any other type of mistake.

The Memphis Grizzlies (5-2^{N}) defeated the Phoenix Suns (3 - 2) **Monday**^{E} 102-91 at the **Talking Stick Resort Arena**^{E} in Phoenix. The Grizzlies had a **strong**^{W} first half where they **out-scored**^{W} the Suns 59^{N}-42^{N}. Marc Gasol scored 18 points, **leading**^{W} the Grizzlies. **Isaiah Thomas added**^{C} 15 points, he is **averaging 19 points on the season so far**^{NC}.

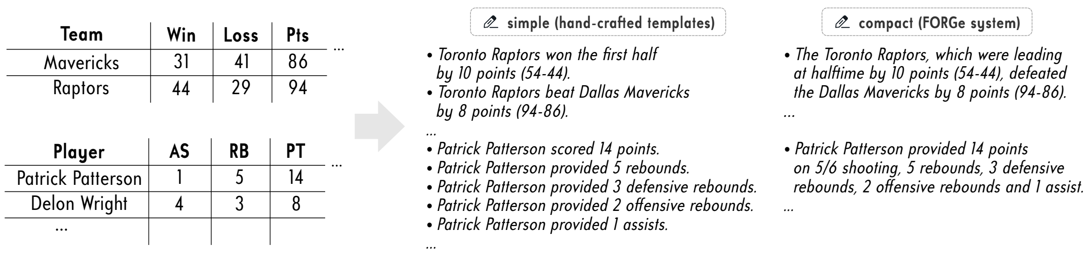
- 2^{N} – Incorrect number, should be 0.
- **Monday**^{E} – Incorrect named entity, should be Wednesday.
- **Talking Stick Resort Arena**^{E} – Incorrect named entity, should be US Airways Center.
- **strong**^{W} – Incorrect word, the Grizzlies did not do well in the first half.
- **out-scored**^{W} – Incorrect word, the Suns had a higher score in first half.
- **59**^{N} – Incorrect number, should be 46.
- **42**^{N} – Incorrect number, should be 52 .
- **leading**^{W} – Incorrect word. Marc Gasol did not lead the Grizzlies, Mike Conley did with 24 points.
- **Isaiah Thomas added**^{C} – Context error. Thomas played for the Suns, but context here implies he played for the Grizzlies and added to their score.
- **averaging 10 points on the season so far**^{NC} – Not checkable. This is very hard to check, since data sources report performance per season and per game, not performance up to a particular point in a season.

Figure 4.2: Example text with error annotations adapted from Thomson and Reiter (2021), using the error marking style from Thomson et al. (2023). The original data for this game is available at `https://www.basketball-reference.com/boxscores/201411050PHO.html` .

An example of an annotated system output is provided in Figure 4.2. The objective of the shared task was to either implement an automatic metric for creating the same type of annotations automatically or to develop a human evaluation scenario capable of producing the same annotations while requiring fewer resources.

4.2.3 Our System

Our submission for the shared task falls into the first category: we developed an automatic metric based on a PLM, which marks each token in the output text for the presence of errors. To make the tabular data understandable for the PLM, we use a rule-based system to exhaustively generate all the facts that can be derived from the data. Since the context window of the language model (LM) is limited, we also use a neural-based retrieval system to retrieve only c relevant facts, which are added into the context window of the LM to support its decisions. We describe individual components of our system below.



Team	Win	Loss	Pts
Mavericks	31	41	86
Raptors	44	29	94

Player	AS	RB	PT
Patrick Patterson	1	5	14
Delon Wright	4	3	8
...			

- *Toronto Raptors won the first half by 10 points (54-44).*
- *Toronto Raptors beat Dallas Mavericks by 8 points (94-86).*

...

- *Patrick Patterson scored 14 points.*
- *Patrick Patterson provided 5 rebounds.*
- *Patrick Patterson provided 3 defensive rebounds.*
- *Patrick Patterson provided 2 offensive rebounds.*
- *Patrick Patterson provided 1 assists.*

...

- *The Toronto Raptors, which were leading at halftime by 10 points (54-44), defeated the Dallas Mavericks by 8 points (94-86).*

...

- *Patrick Patterson provided 14 points on 5/6 shooting, 5 rebounds, 3 defensive rebounds, 2 offensive rebounds and 1 assist.*

...

Figure 4.3: Rule-based NLG which we use to generate facts from the input data. The facts are used as an input to the error checking model (see Figure 4.4). We experiment with (a) simple hand-crafted templates and (b) compact sentences generated by the FORGe system.

Rule-based Fact Descriptions We use a rule-based system to generate facts from the input tables in natural language. For each game, we generate facts about the game (hosting team, visiting team, date converted to weekday), line-score objects (team name and statistics), and box-score objects (player name, player team, player starting position and their personal statistics). We also generate additional facts that can be inferred from the input table, such as which team won and by how much, comparisons between the team and player raw data (e.g., *Team A and Team B committed the same number of fouls*), details based on statistics (e.g., *Player X recorded a double-double*), or an interpretation of some numbers (e.g., *Team A came back in the 4th quarter*).[7]

We experiment with both *simple* descriptions created by filling in sentence templates and *compact* descriptions generated using a grammar-based system:

- **Simple descriptions** are produced by a template-based system, with one template per fact. We handcrafted 129 sentence templates to cover all the facts described above. A sentence template looks like the following: "*[PLAYER_NAME] scored [PTS] points.*", where square brackets indicate variables that are instantiated with the corresponding input values (see Figure 4.3 for sample sentences).
- **Compact descriptions** are produced by the FORGe system (Mille et al., 2019), which allows for the generation of more compact sentences. For instance, the five bottom sentences from the simple system in Figure 4.3 are covered by the single bottom sentence from the compact system. FORGe performs surface

[7]A number of facts frequently mentioned in human-written descriptions could not be obtained from the Rotowire data, as for instance the player stats per quarter, a career-high points total, whether a player is an all-star or not, or if a player scored the winning shot.

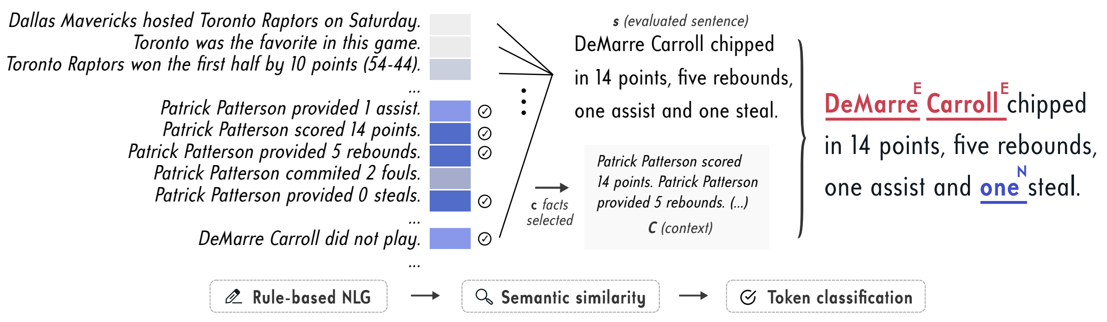

Dallas Mavericks hosted Toronto Raptors on Saturday.
Toronto was the favorite in this game.
Toronto Raptors won the first half by 10 points (54-44).
...
Patrick Patterson provided 1 assist.
Patrick Patterson scored 14 points.
Patrick Patterson provided 5 rebounds.
Patrick Patterson commited 2 fouls.
Patrick Patterson provided 0 steals.
...
DeMarre Carroll did not play.
...
s (evaluated sentence)
DeMarre Carroll chipped in 14 points, five rebounds, one assist and one steal.
c facts selected
Patrick Patterson scored 14 points. Patrick Patterson provided 5 rebounds. (...)
C (context)
DeMarre E Carroll E chipped in 14 points, five rebounds, one assist and one N steal.
Rule-based NLG
Semantic similarity
Token classification

Figure 4.4: An overview of our system. First, we generate the facts from the input table with a rule-based NLG system (see Figure 4.4). For each evaluated sentence s, we select c facts with the highest semantic similarity, getting a context C. The pair (C, s) is given as an input to a pretrained LM for token-level error classification.

realization in several steps, by first aggregating the templates based on the predicate and argument identity and then structuring, linearizing, and inflecting components of the sentences. The FORGe grammars were used off-the-shelf,[8] with additional 98 manually crafted abstract templates.

The simple system produces about 569 facts for each game. The compact system covers the same amount of information with more syntactically complex sentences, producing about 112 sentences per game, i.e., five times less.

Context Retrieval The maximum length of the input sequence for our error tagger is 512 tokens, which is about 10% of the total length of the generated sentences G. Therefore, we select only a subset of G, which we refer to as *context* C. For each generated sentence $g_i \in G$, we measure semantic similarity between g_i and the evaluated sentence s using Sentence Transformers (Reimers and Gurevych, 2019). In particular, we embed the sentence tokens by applying mean pooling on the output of `paraphrase-distilroberta-base-v2`, getting the embedding vectors e_s and e_{g_i}. Then, we compute the cosine similarity between the embeddings:

$$score = \frac{e_s \cdot e_{g_i}}{\|e_s\|\|e_{g_i}\|}. \tag{4.1}$$

For the context C, we select the top c sentences from G that have the highest cosine similarity to s.

[8]We deactivated cross-sentence referring expression generation so that each generated sentence can be used independently.

LM-based Error Tagger As our error tagger, we use RoBERTa (Liu et al., 2019b) with a token-level classification head. The model receives an input $X = (C, s)$ and is trained to annotate each token in s either with an *OK* label or with a label corresponding to one of the error categories. We experiment with two data sources for training the model:

- *gold-standard annotated data* from the shared task (which contains all error types),
- *synthetic data* created by perturbing the human-written summaries from Rotowire (which contains only **NAME**$^{\mathrm{E}}$ and **NUMBER**$^{\mathrm{N}}$ errors).

Synthetic Data The gold-standard data contains only 60 games, as opposed to 3,395 games in the Rotowire training set. This led us to the idea of using the training set as a source of synthetic data, introducing errors into human-written descriptions. We focus only on the **NAME**$^{\mathrm{E}}$ and **NUMBER**$^{\mathrm{N}}$ errors, i.e., the categories that are the most represented and also easiest to generate. In each sentence, we identify named entities in the text using *spaCy*.[9] We modify only a certain portion of entities according to the *entity modification rate* (EMR), which we treat as a hyperparameter. We introduce the **NAME**$^{\mathrm{E}}$ errors by:

(1) swapping the names of teams with opponent teams,

(2) swapping the names of players with other players in the game,

(3) swapping the names of cities with other cities in the Rotowire dataset,

(4) modifying the days of the week.

For **NUMBER**$^{\mathrm{N}}$ errors, we take an integer n identified in the text, sample a number from a normal distribution with $\mu = n$ and $\sigma = 3$, and truncate it to get an integer. We re-sample if the output equals the original number or for negative outputs. If the number is spelled out, we use `text2num`[10] and `num2words`[11] to convert to digits and back.

[9] `https://spacy.io`

[10] `https://pypi.org/project/text2num/`

[11] `https://pypi.org/project/num2words/`

4.2.4 Experiments

We train a PyTorch version of RoBERTa from the Huggingface Transfomers repository (Wolf et al., 2019) using the AdamW optimizer (Loshchilov and Hutter, 2019), learning rate 5×10^{-5} and linear warmup. We finetune the model for 10 epochs and select the model with the highest validation score. We experiment with several hyperparameters:

- *simple* vs. *compact* sentences in G,
- *number of sentences* retrieved for the context: c = 5, 10, 20 or 40;
- *entity modification rate* (EMR): proportion of entities modified in the synthetic data: 0.25, 0.5, or 0.75.

We evaluate the model using a script provided by the organizers, which computes recall and precision of the model output with respect to the human-annotated data. Since we use the human-annotated data for training, we perform 6-fold cross-validation: in each run, we use 45 games for training, 5 games for validation, and 10 games for evaluation.

Gen.	**Data**	c	**EMR = 0.25**			**EMR = 0.5**			**EMR = 0.75**		
			R	P	F1	R	P	F1	R	P	F1
Simple	s	5	0.123	0.723	0.210	0.165	0.512	0.250	0.310	0.323	0.316
		10	0.138	0.737	0.232	0.181	0.549	0.272	0.328	0.400	0.360
		20	0.137	**0.741**	0.231	0.179	0.559	0.271	0.327	0.433	0.373
		40	0.165	0.712	0.268	0.199	0.560	0.294	0.296	0.436	0.353
	s+h	5	0.422	0.617	0.501	0.414	0.594	0.488	0.401	0.583	0.475
		10	0.467	0.551	0.506	0.438	0.638	0.519	0.428	0.665	0.521
		20	0.518	0.640	0.573	0.544	0.575	0.559	0.509	0.595	0.549
		40	0.584	0.644	**0.613**	**0.595**	0.612	0.603	0.519	0.639	0.573
Compact	s	5	0.151	**0.696**	0.248	0.170	0.617	0.267	0.336	0.427	0.376
		10	0.176	0.663	0.278	0.195	0.624	0.297	0.295	0.486	0.367
		20	0.196	0.672	0.303	0.205	0.635	0.310	0.278	0.552	0.370
		40	0.166	0.643	0.264	0.197	0.595	0.296	0.306	0.530	0.388
	s+h	5	0.600	0.641	0.620	0.552	0.635	0.591	0.588	0.600	0.594
		10	0.583	0.662	0.620	0.629	0.606	0.617	**0.656**	0.606	0.630
		20	0.622	0.647	0.634	0.597	0.688	0.639	0.600	0.660	0.629
		40	0.614	0.690	**0.650***	0.609	0.630	0.619	0.611	0.630	0.620

Table 4.2: Recall (R), precision (P) and F1 scores on development data. s stands for synthetic training data and h for human training data. c indicates the number of sentences in the context provided to the tagger, EMR stands for entity modification rate. Best recall, precision and F1 scores for both generators (simple and compact) are shown in bold, the submitted model is identified by an asterisk (*).

Error Type	Mistake		Token	
	R	P	R	P
NAME^{E}	0.750	0.846	0.759	0.862
NUMBER^{N}	0.777	0.750	0.759	0.752
WORD^{W}	0.514	0.483	0.465	0.529
CONTEXT^{C}	0.000	-	0.000	-
NOT_CHECKABLE^{NC}	0.000	-	0.000	-
OTHER^{O}	0.000	-	0.000	-
Overall	0.691	0.756	0.550	0.769

Table 4.3: Results of our system on test data: recall (R) and precision (P) are shown for individual error types.

Development Results The results of our model on the development data are listed in Table 4.2.[12] For our final submission, we selected the model with the best F1-score overall, which is 0.65 (61% recall and 69% precision). The model uses 40 compact sentences in context, 0.25 EMR, and was trained on both synthetic and human-annotated data. Although compact texts are generally helpful, there are also some well-performing models using simple templates only. A higher number of sentences in context may help to achieve better F1-score, but not always (the longer context is also sometimes cropped to fit the input). Using a higher EMR then generally leads to higher recall, suggesting that the model adapts to the base rate of errors.

Submission Results Table 4.3 shows the results of our model on the official test data of the task, broken down by error types. The overall scores are higher than on the development set – test set recall is 0.691 (vs. 0.614 on the development set), and precision is 0.756 (vs. 0.690). The fact that we used all the available human-annotated data for training the final model may have contributed to the difference, but it is also possible that the test data was somewhat less challenging. We note that our model was able to identify only three types of errors (NAME^{E}, NUMBER^{N}, WORD^{W}), having better results for the NAME^{E} and NUMBER^{N} errors. We believe the explanation is two-fold: the names and numbers are often found verbatim in the input data (and in our generated facts), which makes them easy to detect, and also the corresponding error types were the most represented in the training data. In contrast, the three error types that were not detected are much less represented in the training data and hard to detect in our setup.

[12]Due to space constraints, we do not list the results of model trained only on annotated data. The results were overall in the 0.3-0.5 range for both recall and precision, and no model was the best-performing one in terms of any metric.

4.2.5 Discussion

Limitations Our metric depends on the existence of a rule-based system for generating factual statements from the data. Such a system may be hard to develop, even though we have shown that simple templates can be similarly efficient as more complex approaches. As our approach is data-driven, it also requires ~~the~~ system outputs annotated with errors. This requirement may be partially mitigated by using synthetic data. In our case, using synthetic data only results in low recall (see Table 4.2), but more sophisticated techniques for creating the synthetic data may lead to better results.

How to Improve The Metric Our submission achieved the best results in the automatic metrics category, but there is still a gap with what humans can achieve, as shown by the Laval University submission's overall 0.841 recall and 0.879 precision (Garneau and Lamontagne, 2021). One way to improve our system would be to enrich the reference fact descriptions by either inferring more information from the raw data or by extracting additional data from external databases. Another option would be to add surrounding sentences to the context – this could help to resolve coreferences (e.g., if a player is referred to as *"He"*) and to detect the CONTEXT^{C} errors.

LLM-based Alternatives Recently, the evaluation metrics based on LLMs are starting to provide an alternative, more flexible approach for evaluating generated texts (Zhao et al., 2023b; Sottana et al., 2023; Chiang and Lee, 2023; Fu et al., 2023). An advantage of the LLM-based metrics is the possibility of defining custom error categories without the need for having annotated data for finetuning the model. We explore such an approach in Section 6.2, where we use a LLM-based metric for token-level evaluation of generated text. However, it should be noted that with LLM-based metrics, greater flexibility is traded for lower controllability (especially in the case of closed-source models), making the evaluation potentially biased and hard to reproduce (Stureborg et al., 2024; Koo et al., 2023; Wang et al., 2023c).

4.3 Conclusion

We introduced two metrics for evaluating the semantic accuracy of D2T generation. The metric presented in Section 4.1 targets the cases where all the data items need to be mentioned in the output text. It ~~The metric~~ uses a combination of simple templates and an off-the-shelf neural model, making our approach applicable with minimal additional effort. The metric introduced in Section 4.2 then targets complex data-to-text generation, enabling annotating factual errors on the level of individual tokens.

5
Unified Data Processing

In Section, we present an approach for unified processing of data-to-text (D2T) generation datasets. We first convert the input data in 16 D2T generation datasets, of various formats and provenance, into a standard tabular format. On top of the unified format, we build TabGenie: a toolkit combining web interface, command-line interface and Python bindings for simplifying data visualization and processing. While data visualization helps us to present the contents of individual datasets, a unified format opens up the possibility of stre[illegible]ning datasets into pretrained language models (PLMs) for multi-task training. An ~~I~~nteractive mode, in which the model input can be modified on the fly, then helps us to gain insights into ~~the~~ model? system behavior. At the end of the section, we present multiple real-world use cases of our framework.

5.1 TabGenie Toolkit

This section is based on the paper *TabGenie: A Toolkit for Table-to-Text Generation* (Kasner et al., 2023a), joint work with Ekaterina Garanina, Ondřej Plátek, and Ondřej Dušek. The work was published as a system demonstration in the Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (ACL 2023). The project was led by the author of the thesis; the other authors helped with imp[illegible]enting the framework and writing the paper.

We present TabGenie – a toolkit that enables researchers to explore, preprocess, and analyze a variety of D2T generation datasets as tables with associated metadata. The web interface of TabGenie provides an interactive mode for debugging ~~table-to-text generation~~ models, facilitates side-by-side comparison of generated system outputs, and allows easy exports for manual analysis. TabGenie is also equipped with command line processing tools and Python bindings for unified dataset loading and processing. We release TabGenie as a Python package[1] and provide its open-source code and a live demo through Github.[2]

5.1.1 Motivation

In Table 2.2, we provided a representative, although not comprehensive, overview of research datasets for D2T generation. As the number of datasets keeps growing and research keeps accelerating, researchers need to streamline their interactions with these datasets. However, each dataset comes with a different input format and task description, and the input data may not be easy to access and visualize. Platforms such as HuggingFace Datasets (Lhoest et al., 2021) or the GEM benchmark (Gehrmann et al., 2021) provide a unified way to access the datasets, but they ~~platforms~~ still leave the majority of the data processing load on the user.

A key component missing from current D2T tools is the possibility to visualize ~~the~~ input data and generated outputs. Visualization plays an important role in examining and evaluating scientific data (Kehrer and Hauser, 2013) and can help researchers to make more informed design choices. A suitable interface can also encourage researchers to step away from unreliable automatic metrics (Gehrmann et al., 2022) and focus on manual error analysis (van Miltenburg et al., 2021, 2023).

Along with that, demands for a *unified input data format* have recently been raised with multi-task training for large language models (LLMs) (Sanh et al., 2021; Scao et al., 2022; Ouyang et al., 2022, *inter alia*). Some works have used simple data linearization techniques for converting structured data into a textual format to align it with the format used for other tasks (Xie et al., 2022; Tang et al., 2022). However, they use custom preprocessing code for the linearization, leading to discrepancies between individual works.

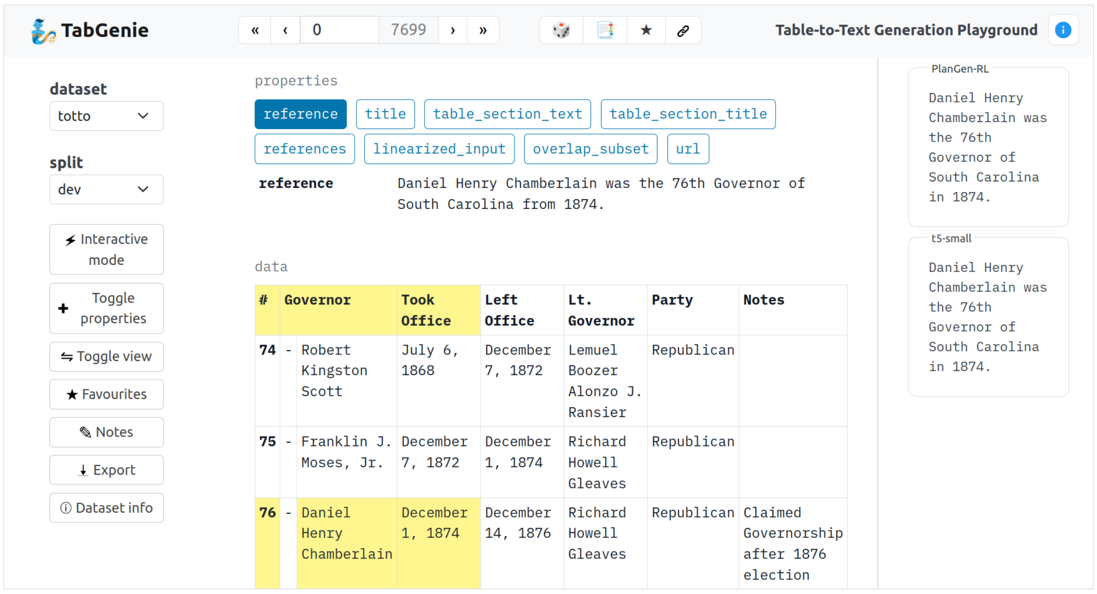
To address these gaps, we present TabGenie – a multi-purpose toolkit for interacting with D2T generation datasets. The cornerstone of TabGenie is a *unified data representation*. Each input is represented as a matrix of m columns and n rows consisting of individual cells accompanied with metadata. Building upon this representation, TabGenie ~~the~~n provides multiple features for unified workflows with table-to-text datasets, including:

(1) visualizing individual dataset examples in ta~~h~~e tabular format,

(2) interacting with table-to-text generation systems in real time,

[1] https://pypi.org/project/tabgenie/

[2] https://github.com/kasnerz/tabgenie


TabGenie

« ‹ 0 7699 › »

Table-to-Text Generation Playground

dataset: totto

split: dev

Interactive mode

Toggle properties

Toggle view

Favourites

Notes

Export

Dataset info

properties

reference | title | table_section_text | table_section_title | references | linearized_input | overlap_subset | url

reference Daniel Henry Chamberlain was the 76th Governor of South Carolina from 1874.

data

#		Governor	Took Office	Left Office	Lt. Governor	Party	Notes
74	-	Robert Kingston Scott	July 6, 1868	December 7, 1872	Lemuel Boozer Alonzo J. Ransier	Republican	
75	-	Franklin J. Moses, Jr.	December 7, 1872	December 1, 1874	Richard Howell Gleaves	Republican	
76	-	Daniel Henry Chamberlain	December 1, 1874	December 14, 1876	Richard Howell Gleaves	Republican	Claimed Governorship after 1876 election

PlanGen-RL

Daniel Henry Chamberlain was the 76th Governor of South Carolina in 1874.

t5-small

Daniel Henry Chamberlain was the 76th Governor of South Carolina in 1874.

Figure 5.1: The web interface of TabGenie. The *left panel* and the top *navigation bar* contain user controls; the *center panel* shows table properties and table content; the *right panel* contains system outputs.

(3) comparing generated system outputs,

(4) loading and preprocessing data for downstream tasks,

(5) exporting examples and generating spreadsheets for manual error analysis.

5.1.2 Data

Input data in TabGenie is in tabular format. We define a *table* as a two-dimensional matrix with m columns and n rows, which together define a grid of $m \times n$ cells. Each cell contains a (possibly empty) text string. A continuous sequence of cells $\{c_i, \ldots, c_{i+k}\}$ from the same row or column may be merged, in which case the values of $\{c_{i+1}, \ldots, c_{i+k}\}$ are linked to the value of c_i. A cell may be optionally marked as a *heading*, which is represented as an additional property of the cell.[3]

To better accommodate the format of datasets such as ToTTo (Parikh et al., 2020) or HiTab (Cheng et al., 2021), we also allow individual cells to be *highlighted*. Highlighted cells are assumed to be preselected for generating ~~the~~ an output description. The tables may be accompanied by an additional set of properties (see Figure 5.1),4 which we These represent as key-value pairs alongside the table. The properties may be used for generating the table description.

[3]The headings are typically located in the first row or column but may also span multiple rows or columns and may not be adjacent.

[4]An example of such a property is a *"title"* of the table in WikiBio (Lebret et al., 2016) or a *"category"* in WebNLG (Gardent et al., 2017b).

Unifying Data Format Our D2T generation datasets contain three high-level input data formats: tables, RDF triples, and key-value pairs. We note that converting the latter two to tabular format requires only minimal changes to the data structure while allowing a unified data representation and visualization. We make a few minor changes to datasets that do not immediately adhere to the tabular format:

- For graph-to-text datasets, we format each triple as a row, using three columns labeled *subject*, *predicate*, and *object*.
- For key-value datasets, we use a two-column format, with keys used as row headings in the first column.
- For SportSett:Basketball (Thomson et al., 2020), we merge the *box score* and *line score* tables and add appropriate headings where necessary.

Datasets We include the 16 datasets listed in Table 2.2 in TabGenie, covering many subtasks of D2T generation. All the datasets are available under a permissive open-source license. To ease the data distribution, we load all the datasets using the Huggingface `datasets` package (Lhoest et al., 2021), which comes equipped with a data downloader. We publicly added to Huggingface `datasets` 9 out of 16 datasets that were not yet available.[5] A custom dataset can be added to TabGenie by simply sub-classing the data loader class and overriding the method for processing individual entries.

5.1.3 Web Interface

TabGenie offers a way to interact with ~~the~~ xxx datasets through ~~the~~ a *web interface*. The interface features a single-page layout with three panels containing user controls, input data, and system outputs (see Figure 5.1).

Content Exploration TabGenie renders input data as HTML tables, providing better visualizations than existing data viewers, especially in the case of large and hierarchical tables.[6] Users can navigate through individual examples in the dataset sequentially, access an example using its index, or go to a random example. Users can add notes to examples and mark examples as favorites to access later. The interface also shows information about the dataset (such as its description, version, homepage, and license) and provides an option to export individual examples.

[5]See `https://huggingface.co/datasets?search=kasnerz`.

[6]Compare, e.g., with the ToTTo dataset ~~in~~ on Huggingface Datasets ~~for which~~, where the table is provided in a single field called *"table"*: `https://huggingface.co/datasets/totto`.

Interactive Mode In the interactive mode, the user can modify the input data and observe how changes influence model outputs. We assume that generation the model runs externally and provides access through a simple an API queried by TabGenie. The user can highlight different cells, edit cell contents, and edit the parameters of the downstream processor.

Pre-generated Outputs TabGenie also allows to visualize static pre-generated outputs, which are loaded in a the JSONL[7] format from a specified directory. Multiple outputs can be displayed alongside a specific example for comparing outputs from multiple systems.

5.1.4 Developer Tools

Besides the web interface, TabGenie also provides developer-friendly access through Python bindings and a command-line interface. Both of these interfaces aim to simplify dataset preprocessing in downstream tasks. The key benefit of using TabGenie is that it provides streamlined access to data in a consistent format, removing the need for dataset-specific code for extracting information such as table properties, references, or individual cell values.

Python Bindings TabGenie can replace custom preprocessing code in Python codebases. With a single unified interface for all the datasets, the TabGenie wrapper class allows to:

- load a dataset from the Huggingface Datasets or a local folder,
- access individual table cells and their properties,
- linearize tables using pre-defined or custom functions,
- prepare xxx the Huggingface `Dataset` objects for downstream processing.

TabGenie can be installed as a Python package, making the integration simple and intuitive.

Command-line Tools TabGenie supports several basic commands via command line:

- **Run** The `tabgenie run` command launches the local web server, mimicking the behavior of `flask run`.

  Example usage:

[7] https://jsonlines.org

```
tabgenie run --port=8890 --host="0.0.0.0"
```

- **Export** The `tabgenie export` command enables batch exporting of the dataset. The supported formats are `xlsx`, `html`, `json`, `txt`, and `csv`. Except for `csv`, table properties can be exported along with the table content.

  Example usage:

  ```
  tabgenie export --dataset "webnlg" \
      --split "dev" \
      --out_dir "export/datasets/webnlg" \
      --export_format "xlsx"
  ```

  Export^{s} can also be done in the web interface.

- **Spreadsheet** For error analysis, it is common to select N random examples from the dataset along with the system outputs and manually annotate them with error categories. The `tabgenie sheet` command generates a suitable spreadsheet for this procedure.

  Example usage:

  ```
  tabgenie sheet --dataset "webnlg" \
      --split "dev" \
      --in_file "out-t5-base.jsonl" \
      --out_file "analysis_webnlg.xlsx" \
      --count 50
  ```

5.1.5 Implementation

TabGenie runs with Python >=3.8 and requires only a few basic packages as dependencies. It can be installed as a stand-alone `tabgenie` module from PyPI or from the project repository.

Backend The web server is based on `Flask`,[8] a popular lightweight Python-based web framework. The server runs locally and can be configured with a YAML[9] configuration file. On startup, the server loads the data using the `datasets`[10] package. To render web pages, the server uses the `tinyhtml`[11] package and the Jinja[12] templating language.

[8] https://pypi.org/project/Flask/
[9] https://yaml.org
[10] https://pypi.org/project/datasets/
[11] https://pypi.org/project/tinyhtml/
[12] https://jinja.palletsprojects.com/

Frontend The web frontend is built on HTML5, CSS, Bootstrap,[13] JavaScript, and jQuery.[14] We additionally use the D3.js[15] library for visualizing the structure of data in graph-to-text datasets. To keep the project simple, we do not use any other major external libraries.

5.1.6 Case Studies

We present several case studies for using TabGenie in D2T generation research. The instructions and code samples for these tasks are available in the project repository.

Table-To-Text Generation TabGenie can serve for finetuning a sequence-to-sequence language model for table-to-text generation in PyTorch (Paszke et al., 2019) using the Huggingface Transformers (Wolf et al., 2019) framework. In a typical finetuning procedure, the user needs to prepare a `Dataset` object with tokenized input and output sequences. Using TabGenie, preprocessing a specific dataset is simplified to the following:

```
from transformers import AutoTokenizer
import tabgenie as tg

# instantiate a tokenizer
tokenizer = AutoTokenizer.from_pretrained(...)

# load the dataset
tg_dataset = tg.load_dataset(
    dataset_name="totto"
)

# preprocess the dataset
hf_dataset = tg_dataset.get_hf_dataset(
    split="train",
    tokenizer=tokenizer
)
```

[13]https://getbootstrap.com/
[14]https://jquery.com
[15]https://d3js.org

The customizable function `get_hf_dataset()` linearizes the tables and tokenizes the inputs and references. For training a single model on multiple datasets in a multi-task learning setting (Xie et al., 2022), the user may preprocess each dataset individually, prepending a dataset-specific task description to each example. The datasets may then be combined using the methods provided by the `datasets` package.

Interactive Prompting TabGenie can be used for observing the impact of various inputs on the outputs of a large language model (LLM) ~~prompted for~~ a table-to-text generation. The user customizes the provided `model_api` pipeline to communicate with an a LLM through an API. The API can communicate either with an external model (using e.g. OpenAI API[16]), or with a model running locally (using libraries such as FastAPI[17]). The user then interacts with the model through the TabGenie web interface, where they are able to modify prompts or table contents as well as highlight specific cells. modifying the prompts, highlighted cells, and table content.

Error Analysis TabGenie can help with annotating error categories in the outputs of from a table-to-text generation model. The user generates xx the system outputs and saves them. ~~the outputs for a particular dataset split~~ in a JSONL format. Through the command-line interface, the user will then generate a XLSX file which can be imported in any suitable office software and distributed to annotators for performing error analysis.

5.1.7 Discussion

Limitations For some D2T inputs, the tabular structure may be inappropriate, such as for tree-based structures (Balakrishnan et al., 2019), bag-of-words (Lin et al., 2019), or multimodal inputs (Krishna et al., 2017). It is also not suitable for the JSON format, which we explore as the format of the input data in Section 6.2. As the framework targets the research community, it requires some programming skills (e.g., for integrating the model API).

Extending the Framework Adding new datasets to TabGenie is straightforward as long as the dataset is convertible to the specified format. Due to deployment issues, TabGenie does not include large synthetic datasets (Agarwal et al., 2021; Jin et al., 2020), but these datasets could be added locally.

[16] https://openai.com/api/
[17] https://fastapi.tiangolo.com

5.2 Conclusion

We presented a toolkit for unified processing of D2T generation datasets. The toolkit enables researchers to gain insights into the datasets by visualizing their contents in the web interface. The toolkit also allows to pre-process the datasets in a unified format, facilitating their processing with language models (LMs). On top of that, the toolkit provides various practical methods for in D2T generation research, such as sending the inputs to a D2T generation models via an API or generating error analysis spreadsheets. As such, the framework promotes more informed and principled D2T generation research.

6 Examining Model Behavior

In this chapter, we observe behaviors of neural language models (LMs) in specific data-to-text (D2T) generation scenarios, building upon custom datasets.

In Section 6.1, we examine the capabilities of pretrained language models (PLMs) to describe unseen relations between entities in knowledge graphs. For this problem, existing D2T datasets are not able to discern memorization from generalization. We thus collect a custom dataset with a large variety of relation labels, including unseen labels in the test set. Using our dataset, we investigate whether the models can correctly describe the relations they have not seen in the training data. We find out that the models can generalize unseen labels as long as the labels are human-readable and unambiguous, which is often (but not always) fulfilled in real-world data.

In Section 6.2, we investigate the abilities of open large language models (LLMs) for D2T generation from common formats such as JSON, CSV, and Markdown. To prevent data contamination, we scrape unlabeled data from public sources across five domains. Using LLM-based referenceless metric and human annotators, we quantify the semantic accuracy of the generated texts with respect to the input data. We find out that although the descriptions are fluent, most of them contain semantic errors.

6.1 Describing Relations in Knowledge Graphs

This section is based on the paper *Mind the Labels: Describing Relations in Knowledge Graphs With Pretrained Models* (Kasner et al., 2023b), joint work with Ioannis Konstas and Ondřej Dušek. The work was published in the Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics (EACL 2023). The project was led by the author of the thesis; Ioannis Konstas and Ondřej Dušek supervised the project.

We investigate how human-readable data labels help PLMs with D2T generation. We start by noticing that PLMs can use labels such as column headings, keys, or relation names to generalize to out-of-domain examples. The question is (a) whether this ability is robust enough and (b) how accurate the outputs are in cases where these labels are ambiguous or incomplete. To answer this question, we focus on describing a relation between two entities.

For our experiments, we collect Rel2Text: a novel dataset for verbalizing a diverse set of 1,522 unique relations from three large-scale knowledge graphs (Wikidata, DBPedia, YAGO). We evaluate model outputs on unseen relations using a combination of automatic metrics and manual analysis. We find that although PLMs for D2T generation expectedly fail on unclear cases, models trained with a large variety of relation labels are surprisingly robust in verbalizing novel, unseen relations. We argue that using data with a diverse set of clear and meaningful labels is key to training D2T generation systems capable of generalizing to novel domains. We release the code and data for our experiments on Github.[1]

6.1.1 Motivation

D2T generation systems need to accurately capture the semantics of relations between values in the data. However, the data labels such as relation names (Färber et al., 2018; Haller et al., 2022), table headings (Parikh et al., 2020), or meaning representation keys (Dušek et al., 2020) may provide only superficial or—if the labels are abbreviations, such as in the Rotowire dataset (Wiseman et al., 2017)—no usable hints about the data semantics.

PLMs such as BART (Lewis et al., 2019) or T5 (Raffel et al., 2019) can quickly adapt to new domains and exhibit robustness to out-of-domain inputs. We investigate to what extent are PLMs limited by the expressivity of the data labels. A suitable testing ground is the task of describing individual RDF triples in a knowledge graph (KG), as shown in Table 6.1. In this task, there is a wide range of lexical choices for the relation

[1] `https://github.com/kasnerz/rel2text`

label	property id	verbalization	note
part of	P361	s is part of o.	can be used verbatim
duration	P2047	s lasted for o.	unambiguous verbalization
platform	P400	s is available on o. s runs on o.	multiple equivalent lexical choices
occupant	P466	o is occupied by s. s plays at o.	semantics depends on entities
parent	P8810	s is the parent of o. o is the parent of s.	ambiguous relation direction

Table 6.1: Example relation labels and the variability in their verbalizations. s and o denote subject and object in the triple, respectively. The Wikidata page for each relation is available at `https://www.wikidata.org/wiki/Property:<property_id>`.

label, while the entities can be copied verbatim or with only minor morphological changes. To illustrate the problem, consider the last example in Table 6.1: the model can use its representation of *"parent"* to understand there is a *"is-a-parent-of"* relation between the entities, but it has to infer (or guess) who is the parent of whom. Even in less ambiguous cases, the model still has to correctly capture the intended semantics of the relation (e.g. *"occupant"* meaning *"home team"*).

Current human-annotated datasets for D2T generation are not suitable for investigating this problem, as they contain only a small number of relations and rarely contain any unseen relations in the test set (Mille et al., 2021). The only existing datasets covering verbalizations of a wider range of KG relations are based on *model-generated outputs* (Agarwal et al., 2021; Amaral et al., 2022). For this reason, we collect a novel human-annotated dataset for the task.

Our aim is also to investigate whether incorporating long-form *descriptions* of data labels helps improve model outputs. Previous works have reached contradictory conclusions: Wang et al. (2021a) use descriptions of relations instead of their labels for relation embeddings, concluding that it results in worse performance on downstream tasks. Conversely, Kale and Rastogi (2020) and Lee et al. (2021) improve the performance of their systems by including schema descriptions on the input for the dialogue state tracking and dialogue response generation systems.

Lastly, we investigate *verbalizing single triples* as a stand-alone task. As we have shown (Sections 3.2, 3.3, and 4.1), in line with other works (Xiang et al., 2022; Kale and Rastogi, 2020; Gupta et al., 2020; Neeraja et al., 2021), transforming triples to text helps for PLMs-based data processing. The works above use various methods for converting individual triples to text, ranging from simple templates and rule-based systems to prompting PLMs; however, none of them investigate how to apply these approaches to novel relations.

6.1.2 Rel2Text dataset

For our experiments, we need data with diverse labels and their human verbalizations. We start by collecting a large set of relations from three large-scale KGs (Wikidata, DBPedia, and YAGO). For each relation, we collect its label, textual description, and up to five triples in which the relation occurs in the KG. We then use human annotators to collect a *verbalization* for each triple, i.e., a short sentence capturing the meaning of the triple. After filtering, our dataset—which we call Rel2Text (Re-writing edge labels to Text)[2]—contains 4,097 single triples covering 1,522 unique relations. We describe the data collection process in the following paragraphs.

Input Data An RDF triple is a tuple $t = (s, r, o)$, where r denotes the relation[3] between the subject s and the object o. We retrieve triples from three open large-scale KGs encoding factual knowledge:

- **Wikidata** (Vrandečić and Krötzsch, 2014) is a large-scale Wikipedia-based KG created using collaborative editing. With approximately 10,000 human-created relations equipped with descriptions[4], it is by far the largest source of variety in relation labels.
- **YAGO** (Tanon et al., 2020) is a KG which builds upon factual knowledge from Wikidata, but uses a limited set of 116 pre-defined relations from `schema.org` (Guha et al., 2016) mapped to a subset of Wikidata relations.
- **DBPedia** (Auer et al., 2007; Lehmann et al., 2015) is a KG that maps Wikipedia infotables to a predefined ontology containing 1,355 relations, about 350 of which are accompanied by a description.

[2]Or simply "Relations-to-Text".
[3]In previous sections, we have also called this constituent a *predicate*; these notions are equivalent.
[4]https://www.wikidata.org/wiki/Wikidata:Database_reports/List_of_properties/all

We query all KGs using their openly available endpoints to retrieve a list of relations in each KG. For each relation, we retrieve up to five *triples* that use this relation and the relation *description*, i.e., a short explanatory text. If present, we also retrieve descriptions for the subject and the object. We apply a set of filtering heuristics, leaving out, e.g., relations describing KG metadata or identification numbers.[5] In this way, we collect 7,334 triples with 1,716 relations in total.

Annotation Process We collect human-written verbalizations for all input triples using Prolific.[6] We built a web interface where human annotators are shown a single triple t and asked to describe it in a single sentence. The annotators are encouraged to re-use the entities in their original form, but they can change the form if necessary. The annotators can also report noisy inputs. We employed 420 annotators in total, each of which annotated 20 examples. We set the average reward per hour according to the platform recommendations to £7.29 per hour, and we accepted all the inputs that passed our built-in checks.

Postprocessing the Data A considerable portion of the collected verbalizations contain typos and grammatical errors, misunderstood meaning of the relation, or extra information in the input. To ensure the high quality of our data, we manually examined all crowdsourced examples and annotated them as *OK*, *noisy*, *corrupted*, or *containing extra information*. For our experiments, we only use the subset of our dataset with *OK* annotations, one per input triple (4,097 examples, 1,522 distinct relations).

6.1.3 Analysis and Experiments

In our analysis, we are interested in the following research questions:

- **RQ1:** Are the PLMs finetuned for D2T generation able to describe relations *not present in the finetuning corpus*?
- **RQ2:** How many *training examples* do the PLMs need to generate satisfactory outputs?
- **RQ3:** How do the PLMs behave when provided *limited lexical cues* about the relation?
- **RQ4:** Can relation *descriptions* help to clarify ambiguous cases and improve the semantic accuracy of the outputs?

[5]Relations describing various IDs make up a large portion of relations in Wikidata. Since we focus on diversity instead of coverage, we decided not to include these relations in our dataset.

[6]`https://www.prolific.co/`

Datasets We experiment with the following datasets, all of which focus on verbalizing factual information from KGs and use the same triple-based input data format:

- REL2TEXT: Our dataset with single triples from three KGs with 4,097 examples, 1,522 relations, and *human-annotated* outputs.
- WebNLG (Ferreira et al., 2020; Gardent et al., 2017b): A DBPedia-based triple-to-text dataset with 38k examples, 411 relations, up to 7 triples per example, and *human-annotated* outputs. We use the English part of version 3.0 from HuggingFace.[7]
- KeLM (Agarwal et al., 2021): A Wikidata-based dataset with 11M examples, 1,519 relations, up to 13 triples per example, and *model-generated* outputs. We use the dataset released by the authors, splitting it into a 1:100 ratio for validation and training data.

To answer the research questions, we divide our REL2TEXT dataset into a training and test split. We then use the REL2TEXT *test set* to evaluate a finetuned BART model (Lewis et al., 2019). To answer *RQ1*, we compare the performance of BART finetuned on the REL2TEXT training set with BART finetuned on WebNLG and KeLM. Using REL2TEXT only, we then prepare various setups for answering *RQ2*, *RQ3*, and *RQ4*.

Rel2Text Data Split We use approximately 15% of the REL2TEXT examples for the *test set*. To ensure maximum fairness and focus on model generalization to unseen relations, we do not include in the REL2TEXT test set any relations that have an exact string match with a relation in KeLM, WebNLG, or the REL2TEXT training set. We also exclude any relations for which the maximum semantic similarity[8] to any KeLM/WebNLG/REL2TEXT training relation exceeds a threshold of 0.9. We set this threshold empirically to exclude relations that are almost synonymous, but slightly lexically different. We use 90% of the remaining examples for the training set and 10% for the validation set.

Experimental Setup We split the camel case in the relation labels. For finetuning the models, we linearize the input triples by marking the triple constituents with special tokens that we add to the model vocabulary. In a default scenario, we finetune BART-base (Lewis et al., 2019) for 10 epochs and select the best checkpoint using validation BLEU score, then use greedy decoding to produce outputs. We repeat each experiment with five random seeds, averaging the results.

[7]`https://huggingface.co/datasets/web_nlg`

[8]Computed as cosine similarity between embeddings of the labels, which are encoded using `all-distilroberta-v1` from SBERT (Reimers and Gurevych, 2019).

Compared Systems We compare the following systems:

- **Copy Baseline**: We introduce a simple *copy* baseline by outputting the triple constituents separated by space: "$s\ r\ o$".
- **Full Training Data**: We use the default setup on full REL2TEXT and WebNLG training sets. For KeLM (which is about 300× larger than WebNLG), we finetune the model only for one epoch. We denote the trained models *full-rel2text*, *full-webnlg*, and *full-kelm*, respectively.
- **Limited Training Data**: For the limited training data setup, we prepare few-shot splits from REL2TEXT as subsets containing $N = \{25, 50, 100, 200\}$ relations with a single example per relation. We select examples at random, ensuring that each few-shot split is a subset of the larger splits. We finetune the *fewshot-N* models for 10 epochs without validation, using the last checkpoint.
- **Limited Lexical Cues**: We investigate how the models behave if we do not include the relation labels at all. We consider three scenarios:
  - *mask-test* – We train the model on REL2TEXT in the standard training setup. For testing, we replace the relation labels in REL2TEXT with the <*mask*> token.
  - *mask-train* – For training, we replace the relation labels in REL2TEXT with the <*mask*> token. We test the model on REL2TEXT in the standard evaluation setup.
  - *mask-all* – We replace the relation labels in REL2TEXT with the <*mask*> token for both training and testing.
- **Incorporating Descriptions**: Our dataset contains short textual descriptions of the relations, which may be useful to disambiguate its meaning and provide additional cues to the model. We consider two scenarios:
  - *desc-repl* – We replace the relation label with its description.
  - *desc-cat* – We concatenate the relation description with the input, separated using the special token <*rel_desc*>.

6.1.4 Evaluation Setup

Automatic Metrics We evaluate generated outputs using an extensive set of automatic metrics from the GEM-metrics[9] package (Gehrmann et al., 2021). We use BLEU (Papineni et al., 2002), METEOR (Banerjee and Lavie, 2005), and BLEURT (Sellam et al., 2020) for measuring lexical similarity. For semantic similarity, we use NUBIA

[9]https://github.com/GEM-benchmark/GEM-metrics

	Label	Example inputs and outputs (✗ incorrect, ✓ correct)
model	Sem	*(Yousra Matine, sport country, Morocco)* ✗ Yousra Matine was born in Morocco. [*mask-mask*] ✓ Yousra Matine plays for Morocco. [*full-rel2text*]
	Dir	*(Kentucky Channel, former broadcast network, KET ED)* ✗ KET ED was broadcast on Kentucky Channel ED. [*fewshot-100*] ✓ The Kentucky Channel was broadcast on KET ED. [*full-rel2text*]
	Lit	*(Vietnam Television, first air date, 1970-09-07)* ✗ The first air date of Vietnam Television was 1970-09-07. [*full-kelm*] ✓ Vietnam Television first aired on 1970-09-07. [*full-rel2text*]
	Lex	*(RPG-43, used in war , The Troubles)* ✗ RPG-43 was used in the The Troubles. [*full-rel2text*] ✓ The RPG-43 was used in the Troubles. [*full-kelm*]
data	Ent	*(The Age of Entitlement, by artist, The Basics)* ✗ The Age of Entitlement was written by The Basics. [*full-kelm*] ✓ The Age of Entitlement was recorded by The Basics. [*full-rel2text*]
	Lbl	*(General Motors Epsilon platform, vehicle, Cadillac XTS)* ✗ General Motors Epsilon is a vehicle similar to the Cadillac XTS. [*full-webnlg*] ✓ General Motors Epsilon platform is used in the Cadillac XTS. [*desc-cat*]

Table 6.2: Error categories used in manual analysis, with examples of errors found and corresponding correct verbalizations (square brackets denote the model). Model error types (top): Sem – The output is semantically incorrect, Dir – The direction of the relation is swapped, Lit – The verbalization is too literal, Lex – There is a lexical error in the output. Input data error types (bottom): Ent – The verbalization may depend on the entities, Lbl – The relation label is not clear.

(Kane et al., 2020) along with its individual features: the semantic similarity score (SS) on a 0-5 scale, the contradiction (C), neutral (N), and entailment (E) probabilities, and the perplexity score (PPL). To assess lexical diversity, we measure the number of unique n-grams (U-1), conditional entropy of bi-grams (CE2), and the mean segmental type-token ratio over segment lengths of 100 (MSTTR). We also measure the average output length in tokens (len). See Section 2.2.7 for a detailed description of the metrics.

Manual Error Analysis Based on our preliminary observations, we identify four sources of model errors: semantic errors (Sem), with a swap of the relation direction (Dir) as a special case, too literal (Lit), i.e., containing awkward or misleading phrasing, and grammar/lexical errors (Lex). We further identified two types of input data errors: ambiguous relations (Ent) and relations with unclear labels (Lbl). Examples are shown in Table 6.2. For error analysis, we select 100 random examples together with their corresponding outputs from the *full-rel2text*, *full-webnlg*, *full-kelm*, *fewshot-100*, *mask-all* and *desc-cat* models. Without revealing the output sources, we mark all error categories that apply.

	Lexical			Semantics					Referenceless				
	BLEU	**MET**	**BLR**	**SS**	**C**	**N**	**E**	**NB**	**U-1**	**CE-2**	**TTR**	**PPL**	**len**
human	-	-	-	-	-	-	-	-	1785	2.13	0.62	5.88	9.55
copy	29.04	37.52	0.09	4.79	1.22	7.57	91.21	0.74	1606	1.17	0.7	7.55	6.72
full-rel2text	52.54	44.86	0.54	4.72	3.50	4.65	91.85	0.88	1661	1.96	0.58	5.89	9.16
full-webnlg	41.99	41.59	0.41	4.65	3.68	6.93	89.39	0.86	1651	2.54	0.56	5.65	10.29
full-kelm	46.74	42.94	0.46	4.70	3.95	5.29	90.77	0.86	1652	2.32	0.56	5.83	9.71
fewshot-25	31.13	35.52	-0.02	3.94	8.35	27.26	64.39	0.65	1445	2.93	0.52	5.34	10.67
fewshot-50	40.60	40.05	0.25	4.44	8.04	13.12	78.84	0.76	1536	2.31	0.55	5.79	9.90
fewshot-100	45.88	42.38	0.38	4.53	6.34	10.60	83.06	0.81	1600	2.13	0.57	5.85	9.57
fewshot-200	48.67	43.34	0.44	4.58	5.40	9.03	85.57	0.83	1626	2.04	0.58	5.89	9.36
mask-test	42.45	38.52	0.25	3.99	14.91	18.47	66.62	0.65	1669	1.96	0.61	5.69	8.96
mask-train	46.90	43.15	0.43	4.55	5.85	11.55	82.61	0.81	1646	2.00	0.57	5.91	9.74
mask-all	42.53	38.49	0.24	3.85	17.58	25.15	57.26	0.61	1677	1.96	0.61	5.66	9.16
desc-repl	49.35	42.85	0.47	4.57	5.78	8.80	85.42	0.82	1693	1.94	0.59	5.86	9.18
desc-cat	53.07	45.04	0.55	4.72	3.46	4.66	91.88	0.87	1668	1.91	0.59	5.92	9.11

Table 6.3: The summary of evaluation using automatic metrics on REL2TEXT test set. **MET** = METEOR, **BLR** = BLEURT, **TTR** = MSTTR. See Section 6.1.3 for the descriptions of the models and metrics.

6.1.5 Findings from Automatic Metrics

Table 6.3 shows automatic scores for all our models.

Lexical and Semantic Similarity *full-rel2text* is the best among the fully trained models regarding lexical overlap metrics, which is expected, as it is trained on the most similar reference distribution. *full-webnlg* and *full-kelm* models are almost equal in terms of semantic consistency, achieving around 90% average entailment probability, which is on par with the copy baseline.

Few-shot Models For the few-shot models, semantic similarity is much lower (e.g., the average entailment probability is between 65% and 85%), showing that there is a certain minimum amount of data needed to achieve consistent outputs. As we show in Figure 6.1, using more examples to train the model generally helps decrease variance and increase performance across various metrics.

Masked Relations Interestingly, the models which do not see the relations during test time (*mask-test* and *mask-all*) still achieve around 60% average entailment probability, similarly to the worst few-shot model. Although their rate of contradictions is higher than for other models, the results suggest that in many cases, the guessed

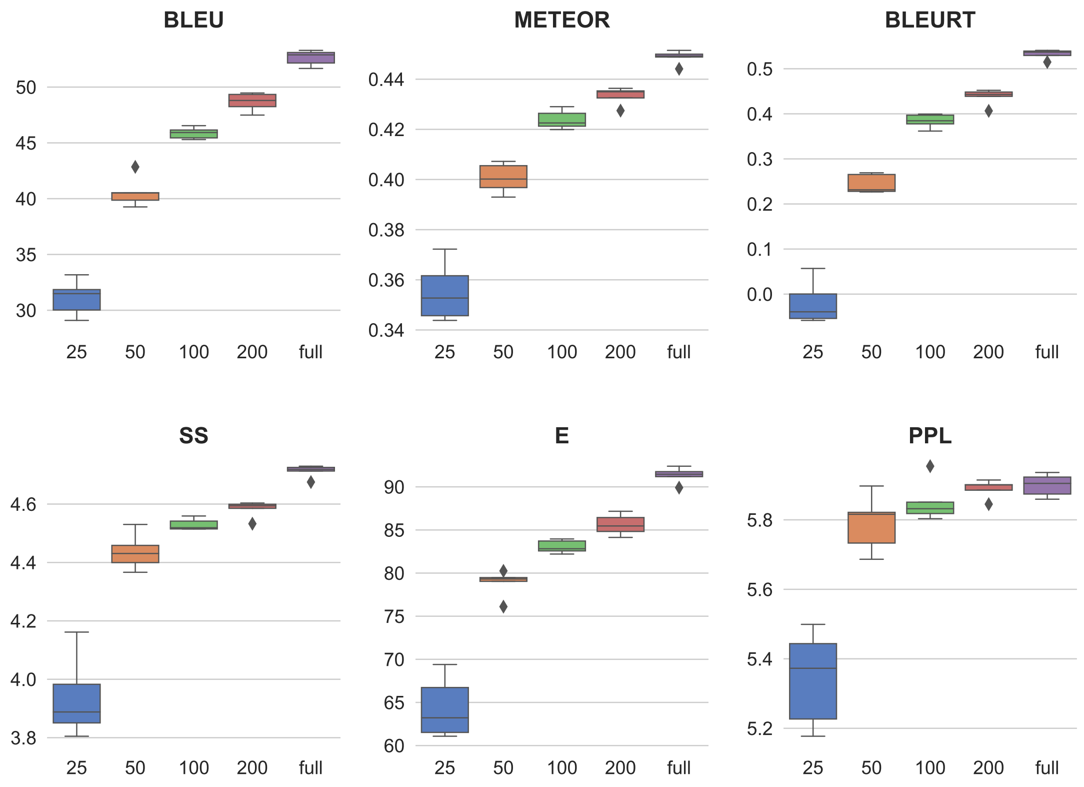


Figure 6.1: Boxplots for selected metrics from Table 6.3 w.r.t. the number of examples (displayed on the x-axis, *full* $= 1522$), taking into account variance from individual random seeds.

relation is semantically consisten with the correct relation. Another interesting observation is that the *mask-train* model (trained not to use the labels) can use the labels provided at test time to improve the outputs considerably (contradiction rate drops from 17% to 5% compared to *mask-all*).

Influence of Descriptions The fact that the short labels are both sufficient and necessary for successful verbalization is emphasized by the fact that the *desc-repl* model is worse than *full-rel2text* (although the descriptions are longer and supposedly explain the relation semantics). Moreover, the benefits of concatenating the descriptions alongside the relation labels (*desc-cat*) are negligible, only slightly improving lexical similarity metrics (0.5 BLEU point gain over *full-rel2text*).

Lexical Diversity In terms of lexical diversity, human references use more unique n-grams, but the model outputs are very similar in other aspects. It remains to be seen if the model outputs can stay semantically consistent with diversity-focused decoding techniques such as nucleus sampling (Holtzman et al., 2019).

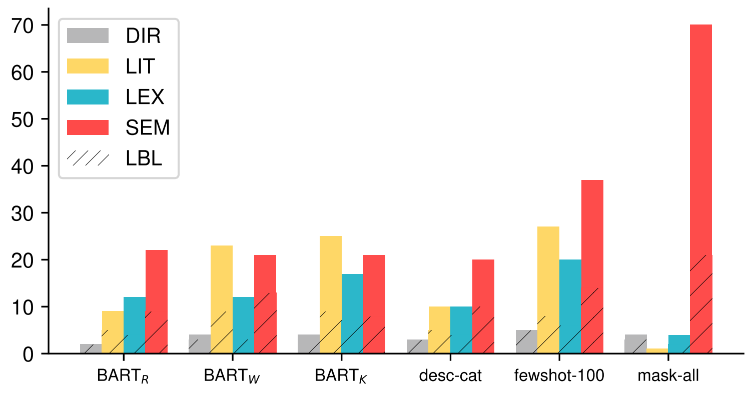


Figure 6.2: Number of annotated errors per model (see Table 6.2 for the description of error categories and Section 6.1.3 for the models). The striped part signifies that the label of the input was marked as unclear.

6.1.6 Findings from Manual Error Analysis

Results are summarized in Figure 6.2; examples of model outputs for each error type are shown in Table 6.2.

Naturalness of Expressions The *full-kelm* and *full-webnlg* models use expressions that are too literal (LIT) in 23 and 29 cases, respectively, while the *full-rel2text* and *desc-cat* models do the same only in 11 cases (5 out of which are marked as LBL, i.e., with an unclear label). This suggests that the variability of our dataset helps models to apply more natural expressions, especially if the relation is understandable from its label.

Semantic Errors There is a near-constant portion of examples where the models make a semantic error (SEM) *and* the input is marked as needing an extra description (LBL). The models also make relatively many semantic errors, most prominently in the case of the *fewshot-100* and the *mask-all* models. The *mask-all* model made a semantic error in 78 cases, suggesting that guessing the exact relation just from the entities is difficult (although still possible in 22 cases). Moreover, the outcomes from this model are fluent (only 4 LEX errors), making it hard to detect faulty cases. The case of swapping the relation direction (DIR) is surprisingly not that common, which is probably down to having only a few examples in our dataset prone to this kind of error.

premise repr.	**dev**	α_1	α_2	α_3
OPR (Gupta et al., 2020)	76.78	75.30	68.46	64.63
BPR (Neeraja et al., 2021)	77.04	74.44	67.46	63.17
full-rel2text (ours)	74.44	74.31	64.59	63.46

Table 6.4: Accuracy for the dev set and test sets $\alpha_{1,2,3}$ from the InfoTabS dataset. The results are averaged over 3 random seeds.

Additional Clues There were only 12 out of 100 examples annotated as Ent, which suggests that the verbalization of the relation can be mostly decided irrespective of the entities in the triple. Notably, the results for *full-rel2text* and *desc-cat* are very similar, rendering the impact of extra descriptions negligible.

6.1.7 Applications to Downstream Tasks

Given that the *full-rel2text* model can describe relations from their labels with high accuracy, we investigate if we can use the model to replace manually created templates in downstream tasks. We select two qualitatively different tasks, both using the idea of transforming individual input triples to simple sentences as a preprocessing step: tabular reasoning and zero-shot data-to-text generation.

Tabular Reasoning For the task of natural language inference (NLI) from a table, Gupta et al. (2020) represent each table cell using a simple template "The *key* of *title* are *value*.". Neeraja et al. (2021) extend their approach by preparing a fine-grained set of rules[10] for individual entity categories. We replicate the setup of Neeraja et al. (2021) for the original (OPR) and better (BPR) paragraph representation using their public codebase. We then replace their templates with our *full-rel2text* model, verbalizing the triple (*title*, *key*, *value*). The results are summarized in Table 6.4. Our manual evaluation suggests that the sentences from our model are indeed more grammatical (even compared to BPR), but we observe comparable performance across all three test sets. In line with McCoy et al. (2019), we conclude that for classification tasks such as NLI, the input content appears to be more important than the input form.

Zero-shot Data-to-Text Generation In Section 3.3, we described our approach for zero-shot D2T generation which requires transforming individual triples into text (Kasner and Dušek, 2022). Here, we replicate the setup on the WebNLG dataset, applying the *full-rel2text* model instead of the templates. The results are summarized

[10]Formalized using more than 250 lines of Python code: `https://github.com/utahnlp/knowledge_infotabs/blob/main/scripts/preprocess/bpr.py#L120`

dataset	model	BLEU	METEOR	O	H
filtered	orig	43.19	39.13	0.152	0.073
	full-rel2text	45.39	38.97	0.056	0.161
full	orig	42.92	39.07	0.051	0.148
	full-rel2text	44.63	38.93	0.058	0.166

Table 6.5: Lexical similarity metrics (BLEU, METEOR) and ommission (O) and hallucinaton (H) rate; following the setup in Kasner and Dušek (2022).

in Table 6.5. We note that the pipeline using our model for preprocessing is able to achieve improvements of ∼2 BLEU points, at the cost of a slightly higher omission and hallucination rate, but crucially without needing the manual effort to create templates. A cursory examination shows that sentences produced by our model are qualitatively similar to the manual templates but more varied. Unlike the templates, our model may verbalize a relation differently depending on the context. Overall, we argue that training a PLM on verbalizing individual relations can potentially replace the manual effort of creating simple templates, which will have a notable impact on scaling similar approaches to larger datasets.

6.1.8 Discussion

Takeaways We showed that PLMs can verbalize novel relations as long as the relation label is human-readable and unambiguous. However, when the cues about the relation are limited, the model will resolve to guessing. A takeaway for datasets that do not follow standard naming conventions, such as the Rotowire dataset (Wiseman et al., 2017), which uses abbreviations for column headers (e.g., FG3A stands for *"the number of shots the player attempted beyond the arc"*), is that rephrasing the labels to natural language may increase the quality of outputs from neural systems. It is still an open question how to handle input data with noisy labels. We suggest detecting and fixing these cases prior to generation, for example, with a human-in-the-loop setup.

Implications for Large Language Models We focused on PLMs, which require at least several hundred examples to produce satisfactory results. With in-context learning, LLMs may bring down the number of examples required close to zero. In this case, the models cannot learn the correct verbalizations from the training data, which makes using clear and unambiguous labels even more important. A follow-up work has shown that prompting LLMs can bring comparable results to using a finetuned PLM in our scenario but requires a more complex setup for controlling the model outputs (Vejvar and Fujimoto, 2023).

Efficient Use of Relation Descriptions To achieve more notable improvements with long-form descriptions of relations, it may be necessary to include a more detailed specification regarding the relation direction, type, or acceptable values. The model also needs to be able to reason about this specification, which could be achieved with the help of LLMs and chain-of-thought reasoning (Wei et al., 2022b; Zhao et al., 2023b).

Limitations Our analysis is limited to verbalizing single triples, which is only a trivial case of graph-to-text generation. Nevertheless, we believe that this simplified setting allows us to distill insights that are still applicable to graph-to-text generation in general. We also focus only on the English part of the KGs: for more morphologically rich languages, an extra effort must be put into correctly inflecting the entities in the generated text.

6.2 Data-to-Text Generation with Large Language Models

This section is based on the paper *Beyond Traditional Benchmarks: Evaluating Open LLMs on Data-to-Text Generation* (Kasner and Dušek, 2024), joint work with Ondřej Dušek. The work ZK: *TODO*

7 Conclusions

Bibliography

Agarwal, A. – Lavie, A. Meteor, m-bleu and m-ter: Evaluation metrics for high-correlation with human rankings of machine translation output. In *Proceedings of the Third Workshop on Statistical Machine Translation*, p. 115–118, 2008.

Agarwal, O. – Kale, M. – Ge, H. – Shakeri, S. – Al-Rfou, R. Machine Translation Aided Bilingual Data-to-Text Generation and Semantic Parsing. 2020, p. 6.

Agarwal, O. – Ge, H. – Shakeri, S. – Al-Rfou, R. Knowledge Graph Based Synthetic Corpus Generation for Knowledge-Enhanced Language Model Pre-training. *arXiv:2010.12688 [cs]*. 2021. Available at: `http://arxiv.org/abs/2010.12688`.

Amaral, G. – Rodrigues, O. – Simperl, E. WDV: A Broad Data Verbalisation Dataset Built from Wikidata. In *International Semantic Web Conference*, p. 556–574. Springer, 2022.

Angeli, G. – Liang, P. – Klein, D. A Simple Domain-Independent Probabilistic Approach to Generation. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, EMNLP 2010, 9-11 October 2010, MIT Stata Center, Massachusetts, USA, A meeting of SIGDAT, a Special Interest Group of the ACL*, p. 502–512, 2010. Available at: `https://aclanthology.org/D10-1049/`.

Aoki, T. – Miyazawa, A. – Ishigaki, T. – Goshima, K. – Aoki, K. – Kobayashi, I. – Takamura, H. – Miyao, Y. Generating market comments referring to external resources. In *Proceedings of the 11th International Conference on Natural Language Generation*, p. 135–139, 2018.

Aroca-Ouellette, S. – Rudzicz, F. On losses for modern language models. *arXiv preprint arXiv:2010.01694*. 2020.

Attardi, G. WikiExtractor. `https://github.com/attardi/wikiextractor`, 2015.

Auer, S. – Bizer, C. – Kobilarov, G. – Lehmann, J. – Cyganiak, R. – Ives, Z. DBpedia: A Nucleus for a Web of Open Data. In *International Semantic Web Conference*, p. 722–735, 2007.

Axelsson, A. – Skantze, G. Using Large Language Models for Zero-Shot Natural Language Generation from Knowledge Graphs, 2023. Available at: `http://arxiv.org/abs/2307.07312`.

Ba, J. L. – Kiros, J. R. – Hinton, G. E. Layer normalization. *arXiv preprint arXiv:1607.06450*. 2016.

Bahdanau, D. – Cho, K. – Bengio, Y. Neural machine translation by jointly learning to align and translate. *ICLR*. 2015.

Balakrishnan, A. – Rao, J. – Upasani, K. – White, M. – Subba, R. Constrained Decoding for Neural NLG from Compositional Representations in Task-Oriented Dialogue. In Korhonen, A. – Traum, D. R. – Màrquez, L. (Ed.) *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Volume 1: Long Papers*, p. 831–844, Florence, Italy, 2019. doi: 10.18653/V1/P19-1080. Available at: `https://doi.org/10.18653/v1/p19-1080`.

Balloccu, S. – Reiter, E. Comparing informativeness of an NLG chatbot vs graphical app in diet-information domain. In Shaikh, S. – Ferreira, T. – Stent, A. (Ed.) *Proceedings of the 15th International Conference on Natural Language Generation*, p. 156–185, Waterville, Maine, USA and virtual meeting, July 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.inlg-main.13. Available at: `https://aclanthology.org/2022.inlg-main.13`.

Balloccu, S. – Schmidtová, P. – Lango, M. – Dušek, O. Leak, Cheat, Repeat: Data Contamination and Evaluation Malpractices in Closed-Source LLMs. *arXiv preprint arXiv:2402.03927*. 2024.

Banarescu, L. – Bonial, C. – Cai, S. – Georgescu, M. – Griffitt, K. – Hermjakob, U. – Knight, K. – Koehn, P. – Palmer, M. – Schneider, N. Abstract meaning representation for sembanking. In *Proceedings of the 7th linguistic annotation workshop and interoperability with discourse*, p. 178–186, 2013.

Banerjee, S. – Lavie, A. 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*, p. 65–72, Ann Arbor, Michigan, jun 2005. Association for Computational Linguistics. Available at: `https://www.aclweb.org/anthology/W05-0909`.

Bangalore, S. – Rambow, O. Corpus-based lexical choice in natural language generation. In *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics*, p. 464–471, 2000.

Bao, J. – Tang, D. – Duan, N. – Yan, Z. – Lv, Y. – Zhou, M. – Zhao, T. Table-to-text: Describing table region with natural language. In *Proceedings of the AAAI conference on artificial intelligence*, 32, 2018.

Barzilay, R. – Lee, L. Catching the drift: Probabilistic content models, with applications to generation and summarization. *arXiv preprint cs/0405039*. 2004.

Barzilay, R. – McKeown, K. R. Sentence Fusion for Multidocument News Summarization. *Computational Linguistics*. 2005, 31, 3, p. 297–328. doi: 10.1162/089120105774321091. Available at: `https://www.aclweb.org/anthology/J05-3002`.

Barzilay, R. – Elhadad, N. – McKeown, K. R. Sentence Ordering in Multidocument Summarization. In *Proceedings of the First International Conference on Human Language Technology Research, HLT 2001*, San Diego, CA, USA, 2001. Available at: https://aclanthology.org/H01-1065/.

Bateman, J. A. Enabling technology for multilingual natural language generation: the KPML development environment. *Natural Language Engineering*. 1997, 3, 1, p. 15–55.

Baum, L. E. – Petrie, T. Statistical inference for probabilistic functions of finite state Markov chains. *The annals of mathematical statistics*. 1966, 37, 6, p. 1554–1563.

Belz, A. Corpus-driven generation of weather forecasts. In *Proc. 3rd Corpus Linguistics Conference*, 2005.

Belz, A. Automatic generation of weather forecast texts using comprehensive probabilistic generation-space models. *Nat. Lang. Eng.* 2008a, 14, 4, p. 431–455. doi: 10.1017/S1351324907004664. Available at: https://doi.org/10.1017/S1351324907004664.

Belz, A. Automatic Generation of Weather Forecast Texts Using Comprehensive Probabilistic Generation-Space Models. *Nat. Lang. Eng.* 2008b, 14, p. 431–455. ISSN 1351-3249, 1469-8110. doi: 10.1017/S1351324907004664. Available at: https://www.cambridge.org/core/product/identifier/S1351324907004664/type/journal_article.

Belz, A. – Mille, S. – Howcroft, D. M. Disentangling the Properties of Human Evaluation Methods: A Classification System to Support Comparability, Meta-Evaluation and Reproducibility Testing. In *Proceedings of the 13th International Conference on Natural Language Generation*, p. 183–194, Dublin, Ireland, 2020. Association for Computational Linguistics. Available at: https://www.aclweb.org/anthology/2020.inlg-1.24.

Bird, S. NLTK: The Natural Language Toolkit. In Calzolari, N. – Cardie, C. – Isabelle, P. (Ed.) *ACL 2006, 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference*, Sydney, Australia, 2006. doi: 10.3115/1225403.1225421. Available at: https://aclanthology.org/P06-4018/.

Bishop, C. M. Pattern recognition and machine learning. *Springer google schola*. 2006, 2, p. 5–43.

Botha, J. A. – Faruqui, M. – Alex, J. – Baldridge, J. – Das, D. Learning to Split and Rephrase From Wikipedia Edit History. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, p. 732–737, Brussels, Belgium, 2018. doi: 10.18653/v1/D18-1080. Available at: https://www.aclweb.org/anthology/D18-1080.

Brown, T. – Mann, B. – Ryder, N. – Subbiah, M. – Kaplan, J. D. – Dhariwal, P. – Neelakantan, A. – Shyam, P. – Sastry, G. – Askell, A. – others. Language models are few-shot learners. *Advances in neural information processing systems*. 2020, 33, p. 1877–1901.

Bubeck, S. – Chandrasekaran, V. – Eldan, R. – Gehrke, J. – Horvitz, E. – Kamar, E. – Lee, P. – Lee, Y. T. – Li, Y. – Lundberg, S. – others. Sparks of artificial general intelligence: Early experiments with gpt-4. *arXiv preprint arXiv:2303.12712*. 2023.

Budzianowski, P. – Wen, T.-H. – Tseng, B.-H. – Casanueva, I. – Ultes, S. – Ramadan, O. – Gašić, M. MultiWOZ – A Large-Scale Multi-Domain Wizard-of-Oz Dataset for Task-Oriented Dialogue Modelling. *arXiv:1810.00278 [cs]*. 2020. Available at: http://arxiv.org/abs/1810.00278.

Calizzano, R. – Ostendorff, M. – Rehm, G. Ordering sentences and paragraphs with pre-trained encoder-decoder transformers and pointer ensembles. In Healy, P. – Bilauca, M. – Bonnici, A. (Ed.) *DocEng '21: ACM Symposium on Document Engineering 2021*, p. 10:1–10:9, Limerick, Ireland, 2021. doi: 10.1145/3469096.3469874. Available at: https://doi.org/10.1145/3469096.3469874.

Castro Ferreira, T. – Moussallem, D. – Krahmer, E. – Wubben, S. Enriching the WebNLG Corpus. In *Proceedings of the 11th International Conference on Natural Language Generation*, p. 171–176, Tilburg University, The Netherlands, 2018. Association for Computational Linguistics. doi: 10.18653/v1/W18-6521. Available at: https://www.aclweb.org/anthology/W18-6521.

Celikyilmaz, A. – Clark, E. – Gao, J. Evaluation of Text Generation: A Survey. *arXiv:2006.14799 [cs]*. 2021. Available at: http://arxiv.org/abs/2006.14799.

Cer, D. – Diab, M. – Agirre, E. – Lopez-Gazpio, I. – Specia, L. SemEval-2017 Task 1: Semantic Textual Similarity Multilingual and Crosslingual Focused Evaluation. In Bethard, S. – Carpuat, M. – Apidianaki, M. – Mohammad, S. M. – Cer, D. – Jurgens, D. (Ed.) *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, p. 1–14, Vancouver, Canada, August 2017. Association for Computational Linguistics. doi: 10.18653/v1/S17-2001. Available at: https://aclanthology.org/S17-2001.

Chang, E. – Shen, X. – Marin, A. – Demberg, V. The SelectGen Challenge: Finding the Best Training Samples for Few-Shot Neural Text Generation. In *Proceedings of the 14th International Conference on Natural Language Generation*, p. 325–330, Aberdeen, Scotland, UK, 2021a. Association for Computational Linguistics. Available at: https://aclanthology.org/2021.inlg-1.36.

Chang, E. – Shen, X. – Zhu, D. – Demberg, V. – Su, H. Neural Data-to-Text Generation with LM-based Text Augmentation. *arXiv:2102.03556 [cs]*. 2021b. Available at: http://arxiv.org/abs/2102.03556.

Chapman, C. L. – Hillebrand, L. – Stenzel, M. R. – Deußer, T. – Biesner, D. – Bauckhage, C. – Sifa, R. Towards generating financial reports from tabular data using transformers. In *International Cross-Domain Conference for Machine Learning and Knowledge Extraction*, p. 221–232. Springer, 2022.

Chen, J. – Xu, R. – Zeng, W. – Sun, C. – Li, L. – Xiao, Y. Converge to the truth: Factual error correction via iterative constrained editing. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 37, p. 12616–12625, 2023.

Chen, W. – Chen, J. – Su, Y. – Chen, Z. – Wang, W. Y. Logical Natural Language Generation from Open-Domain Tables. 2020a. Available at: https://arxiv.org/abs/2004.10404v2.

Chen, W. – Su, Y. – Yan, X. – Wang, W. Y. KGPT: Knowledge-Grounded Pre-Training for Data-to-Text Generation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, p. 8635–8648, Online, 2020b. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.697. Available at: https://www.aclweb.org/anthology/2020.emnlp-main.697.

Chen, Y. – Eger, S. MENLI: Robust Evaluation Metrics from Natural Language Inference, 2022. Available at: http://arxiv.org/abs/2208.07316.

Chen, Z. – Eavani, H. – Chen, W. – Liu, Y. – Wang, W. Y. Few-Shot NLG with Pre-Trained Language Model. *arXiv:1904.09521 [cs]*. 2019. Available at: http://arxiv.org/abs/1904.09521.

Chen, Z. – Chen, W. – Zha, H. – Zhou, X. – Zhang, Y. – Sundaresan, S. – Wang, W. Y. Logic2Text: High-Fidelity Natural Language Generation from Logical Forms. *arXiv:2004.14579 [cs]*. 2020c. Available at: http://arxiv.org/abs/2004.14579.

Cheng, J. – Dong, L. – Lapata, M. Long Short-Term Memory-Networks for Machine Reading. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, p. 551–561, 2016.

Cheng, Z. – Dong, H. – Wang, Z. – Jia, R. – Guo, J. – Gao, Y. – Han, S. – Lou, J.-G. – Zhang, D. HiTab: A Hierarchical Table Dataset for Question Answering and Natural Language Generation. *arXiv:2108.06712 [cs]*. 2021. Available at: http://arxiv.org/abs/2108.06712.

Chiang, D. C. – Lee, H. Can Large Language Models Be an Alternative to Human Evaluations? In Rogers, A. – Boyd-Graber, J. L. – Okazaki, N. (Ed.) *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2023*, p. 15607–15631, Toronto, Canada, 2023. doi: 10.18653/V1/2023.ACL-LONG.870. Available at: https://doi.org/10.18653/v1/2023.acl-long.870.

Cho, K. – Merrienboer, B. – Gulcehre, C. – Bahdanau, D. – Bougares, F. – Schwenk, H. – Bengio, Y. 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)*, p. 1724–1734, Doha, Qatar, October 2014. Association for Computational Linguistics. doi: 10.3115/v1/D14-1179. Available at: https://www.aclweb.org/anthology/D14-1179.

Chomsky, N. *Syntactic Structures.* The Hague/Paris: Mouton, 1957. ISBN 978-3-11-021832-9.

Clark, E. – August, T. – Serrano, S. – Haduong, N. – Gururangan, S. – Smith, N. A. All That's 'Human' Is Not Gold: Evaluating Human Evaluation of Generated Text. *arXiv:2107.00061 [cs]*. 2021. Available at: http://arxiv.org/abs/2107.00061.

Colas, A. – Sadeghian, A. – Wang, Y. – Wang, D. Z. EventNarrative: A Large-scale Event-centric Dataset for Knowledge Graph-to-Text Generation. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 1)*, 2021.

Cotterell, R. – Svete, A. – Meister, C. – Liu, T. – Du, L. Formal Aspects of Language Modeling, 2024.

Cripwell, L. – Belz, A. – Gardent, C. – Gatt, A. – Borg, C. – Borg, M. – Judge, J. – Lorandi, M. – Nikiforovskaya, A. – Soto Martinez, W. The 2023 WebNLG Shared Task on Low Resource Languages. Overview and Evaluation Results (WebNLG 2023). In Gatt, A. – Gardent, C. – Cripwell, L. – Belz, A. – Borg, C. – Erdem, A. – Erdem, E. (Ed.) *Proceedings of the Workshop on Multimodal, Multilingual Natural Language Generation and Multilingual WebNLG Challenge (MM-NLG 2023)*, p. 55–66, Prague, Czech Republic, 2023. Association for Computational Linguistics. Available at: https://aclanthology.org/2023.mmnlg-1.6.

Dale, R. Natural Language Generation: The Commercial State of the Art in 2020. *Natural Language Engineering*. 2020, 26, p. 481–487. ISSN 1351-3249, 1469-8110. doi: 10.1017/S135132492000025X. Available at: https://www.cambridge.org/core/product/identifier/S135132492000025X/type/journal_article.

Dale, R. Navigating the Text Generation Revolution: Traditional Data-to-Text NLG Companies and the Rise of ChatGPT. *Natural Language Engineering*. 2023, 29, p. 1188–1197. ISSN 1351-3249, 1469-8110. doi: 10.1017/S1351324923000347. Available at: https://www.cambridge.org/core/journals/natural-language-engineering/article/navigating-the-text-generation-revolution-traditional-datatotext-nlg-companies-and-the-rise-of-chatgpt/F43278ED38F2F7F709488625CDAF5829.

Demir, S. – Carberry, S. – McCoy, K. Generating Textual Summaries of Bar Charts. In *Proceedings of the Fifth International Natural Language Generation Conference*, p. 7–15, Salt Fork, Ohio, USA, 2008. Association for Computational Linguistics. Available at: https://aclanthology.org/W08-1103.

Demir, S. – Carberry, S. – McCoy, K. F. Summarizing Information Graphics Textually. *Computational Linguistics*. 2012, 38, p. 527–574. ISSN 0891-2017. doi: 10.1162/COLI_a_00091. Available at: https://doi.org/10.1162/COLI_a_00091.

Devlin, J. – Chang, M.-W. – Lee, K. – Toutanova, K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *arXiv:1810.04805 [cs]*. 2019. Available at: http://arxiv.org/abs/1810.04805.

Dhingra, B. – Faruqui, M. – Parikh, A. – Chang, M.-W. – Das, D. – Cohen, W. W. Handling Divergent Reference Texts When Evaluating Table-to-Text Generation. *arXiv:1906.01081 [cs]*. 2019. Available at: http://arxiv.org/abs/1906.01081.

Doddington, G. Automatic evaluation of machine translation quality using n-gram co-occurrence statistics. In *Proceedings of the second international conference on Human Language Technology Research*, p. 138–145, 2002.

Dong, Q. – Li, L. – Dai, D. – Zheng, C. – Wu, Z. – Chang, B. – Sun, X. – Xu, J. – Sui, Z. A survey on in-context learning. *arXiv preprint arXiv:2301.00234*. 2022.

Driverless Future. Autonomous Car Forecasts. https://www.driverless-future.com/?page_id=384, 2017. Accessed on March 08, 2024.

Dubey, S. R. – Singh, S. K. – Chaudhuri, B. B. Activation Functions in Deep Learning: A Comprehensive Survey and Benchmark, 2022.

Duboue, P. – Mckeown, K. Statistical Acquisition of Content Selection Rules for Natural Language Generation. In *Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing*, p. 121–128, 2003.

Dufter, P. – Schmitt, M. – Schütze, H. Position Information in Transformers: An Overview. *Computational Linguistics*. 2022, 48, 3, p. 733–763.

Dušek, O. – Jurčíček, F. Sequence-to-Sequence Generation for Spoken Dialogue via Deep Syntax Trees and Strings. *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. 2016, p. 45–51. doi: 10.18653/v1/P16-2008. Available at: http://arxiv.org/abs/1606.05491.

Dušek, O. – Jurcicek, F. Training a Natural Language Generator From Unaligned Data. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, p. 451–461, Beijing, China, 2015. Association for Computational Linguistics. doi: 10.3115/v1/P15-1044. Available at: http://aclweb.org/anthology/P15-1044.

Dušek, O. – Jurčíček, F. Neural Generation for Czech: Data and Baselines. In *Proceedings of the 12th International Conference on Natural Language Generation*, p. 563–574, 2019.

Dušek, O. – Kasner, Z. Evaluating Semantic Accuracy of Data-to-Text Generation with Natural Language Inference. In *Proceedings of the 13th International Conference on Natural Language Generation*, p. 131–137, Dublin, Ireland, 2020. Association for Computational Linguistics. Available at: https://www.aclweb.org/anthology/2020.inlg-1.19.

Dušek, O. – Howcroft, D. M. – Rieser, V. Semantic Noise Matters for Neural Natural Language Generation. *arXiv:1911.03905 [cs]*. 2019. Available at: http://arxiv.org/abs/1911.03905.

Dušek, O. – Novikova, J. – Rieser, V. Evaluating the State-of-the-Art of End-to-End Natural Language Generation: The E2E NLG Challenge. *Computer Speech & Language*. 2020, 59, p. 123–156. ISSN 08852308. doi: 10.1016/j.csl.2019.06.009. Available at: https://linkinghub.elsevier.com/retrieve/pii/S0885230819300919.

Elhadad, M. – Robin, J. SURGE: a comprehensive plug-in syntactic realization component for text generation. *Computational Linguistics*. 1997, 99, 4.

Färber, M. – Bartscherer, F. – Menne, C. – Rettinger, A. Linked data quality of DBpedia, Freebase, OpenCyc, Wikidata, and YAGO. *Semantic Web*. 2018, 9, 1, p. 77–129. doi: 10.3233/SW-170275. Available at: https://doi.org/10.3233/SW-170275.

Ferreira, T. C. – van der Lee, C. – van Miltenburg, E. – Krahmer, E. Neural Data-to-Text Generation: A Comparison between Pipeline and End-to-End Architectures. *arXiv:1908.09022 [cs]*. 2019. Available at: http://arxiv.org/abs/1908.09022.

Ferreira, T. C. – Gardent, C. – Ilinykh, N. – Van Der Lee, C. – Mille, S. – Moussallem, D. – Shimorina, A. The 2020 bilingual, bi-directional webnlg+ shared task overview and evaluation results (webnlg+ 2020). In *Proceedings of the 3rd International Workshop on Natural Language Generation from the Semantic Web (WebNLG+)*, 2020.

Firth, J. A synopsis of linguistic theory, 1930-1955. *Studies in linguistic analysis*. 1957, p. 10–32.

Freitag, M. – Roy, S. Unsupervised Natural Language Generation with Denoising Autoencoders. *arXiv:1804.07899 [cs]*. 2018. Available at: http://arxiv.org/abs/1804.07899.

Fu, J. – Ng, S. – Jiang, Z. – Liu, P. GPTScore: Evaluate as You Desire. *CoRR*. 2023, abs/2302.04166. doi: 10.48550/ARXIV.2302.04166. Available at: https://doi.org/10.48550/arXiv.2302.04166.

Gao, L. – Biderman, S. – Black, S. – Golding, L. – Hoppe, T. – Foster, C. – Phang, J. – He, H. – Thite, A. – Nabeshima, N. – others. The pile: An 800gb dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027*. 2020.

Gardent, C. – Perez-Beltrachini, L. A statistical, grammar-based approach to microplanning. *Computational Linguistics*. 2017, 43, 1, p. 1–30.

Gardent, C. – Shimorina, A. – Narayan, S. – Perez-Beltrachini, L. Creating Training Corpora for NLG Micro-Planners. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, p. 179–188, Vancouver, Canada, 2017a. Association for Computational Linguistics. doi: 10.18653/v1/P17-1017. Available at: https://www.aclweb.org/anthology/P17-1017.

Gardent, C. – Shimorina, A. – Narayan, S. – Perez-Beltrachini, L. The WebNLG Challenge: Generating Text from RDF Data. In *Proceedings of the 10th International Conference on Natural Language Generation*, p. 124–133, Santiago de Compostela, Spain, 2017b. Association for Computational Linguistics. doi: 10.18653/v1/W17-3518. Available at: http://aclweb.org/anthology/W17-3518.

Gardner, M. – Grus, J. – Neumann, M. – Tafjord, O. – Dasigi, P. – Liu, N. F. – Peters, M. E. – Schmitz, M. – Zettlemoyer, L. AllenNLP: A Deep Semantic Natural Language Processing Platform. *CoRR*. 2018, abs/1803.07640. Available at: http://arxiv.org/abs/1803.07640.

Garneau, N. – Lamontagne, L. Shared Task in Evaluating Accuracy: Leveraging Pre-Annotations in the Validation Process. In *Proceedings of the 14th International Conference on Natural Language Generation*, Aberdeen, Scotland, UK, 2021.

Gatt, A. – Krahmer, E. Survey of the State of the Art in Natural Language Generation: Core Tasks, Applications and Evaluation. *jair*. 2018, 61, p. 65–170. ISSN 1076-9757. doi: 10.1613/jair.5477. Available at: https://jair.org/index.php/jair/article/view/11173.

Gatt, A. – Reiter, E. SimpleNLG: A realisation engine for practical applications. In *Proceedings of the 12th European workshop on natural language generation (ENLG 2009)*, p. 90–93, 2009.

Gehrmann, S. – Dai, F. Z. – Elder, H. – Rush, A. M. End-to-End Content and Plan Selection for Data-to-Text Generation. *arXiv:1810.04700 [cs]*. 2018. Available at: http://arxiv.org/abs/1810.04700.

Gehrmann, S. et al. The GEM Benchmark: Natural Language Generation, Its Evaluation and Metrics. *arXiv:2102.01672 [cs]*. 2021. Available at: http://arxiv.org/abs/2102.01672.

Gehrmann, S. – Clark, E. – Sellam, T. Repairing the Cracked Foundation: A Survey of Obstacles in Evaluation Practices for Generated Text. *arXiv:2202.06935 [cs]*. 2022. Available at: http://arxiv.org/abs/2202.06935.

Geva, M. – Malmi, E. – Szpektor, I. – Berant, J. DiscoFuse: A Large-Scale Dataset for Discourse-Based Sentence Fusion. 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)*, p. 3443–3455, Minneapolis, Minnesota, jun 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1348. Available at: https://www.aclweb.org/anthology/N19-1348.

Gkatzia, D. Content Selection in Data-to-Text Systems: A Survey. *arXiv:1610.08375 [cs]*. 2016. Available at: http://arxiv.org/abs/1610.08375.

Gkatzia, D. – Mahamood, S. A snapshot of NLG evaluation practices 2005-2014. In *Proceedings of the 15th European Workshop on Natural Language Generation (ENLG)*, p. 57–60, 2015.

Goldberg, E. – Driedger, N. – Kittredge, R. I. Using natural-language processing to produce weather forecasts. *IEEE Expert*. 1994, 9, 2, p. 45–53.

Goodfellow, I. – Bengio, Y. – Courville, A. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

Grajcar, P. Data-to-text generation with text-editing models. 2023.

Grusky, M. Rogue Scores. In Rogers, A. – Boyd-Graber, J. – Okazaki, N. (Ed.) *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, p. 1914–1934, Toronto, Canada, 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-long.107. Available at: https://aclanthology.org/2023.acl-long.107.

Gu, J. – Lu, Z. – Li, H. – Li, V. O. Incorporating Copying Mechanism in Sequence-to-Sequence Learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, p. 1631–1640, 2016.

Guha, R. V. – Brickley, D. – Macbeth, S. Schema.org: evolution of structured data on the web. *Commun. ACM*. 2016, 59, 2, p. 44–51. doi: 10.1145/2844544. Available at: https://doi.org/10.1145/2844544.

Gupta, V. – Mehta, M. – Nokhiz, P. – Srikumar, V. INFOTABS: Inference on Tables as Semi-structured Data. In Jurafsky, D. – Chai, J. – Schluter, N. – Tetreault, J. R. (Ed.) *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, p. 2309–2324. Association for Computational Linguistics, 2020. doi: 10.18653/v1/2020.acl-main.210. Available at: https://doi.org/10.18653/v1/2020.acl-main.210.

Gururaja, S. – Bertsch, A. – Na, C. – Widder, D. – Strubell, E. To Build Our Future, We Must Know Our Past: Contextualizing Paradigm Shifts in Natural Language Processing. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, p. 13310–13325, 2023.

Haller, A. – Polleres, A. – Dobriy, D. – Ferranti, N. – Méndez, S. J. R. An Analysis of Links in Wikidata. In Groth, P. – Vidal, M. – Suchanek, F. M. – Szekely, P. A. – Kapanipathi, P. – Pesquita, C. – Skaf-Molli, H. – Tamper, M. (Ed.) *The Semantic Web - 19th International Conference, ESWC 2022, Hersonissos, Crete, Greece, May 29 - June 2, 2022, Proceedings*, 13261 / *Lecture Notes in Computer Science*, p. 21–38. Springer, 2022. doi: 10.1007/978-3-031-06981-9_2. Available at: https://doi.org/10.1007/978-3-031-06981-9_2.

Halliday, M. A. Systemic background. *Systemic perspectives on discourse*. 1985, 1, p. 1–15.

Han, J. – Beck, D. – Cohn, T. Generating Diverse Descriptions from Semantic Graphs. *arXiv:2108.05659 [cs]*. 2021. Available at: http://arxiv.org/abs/2108.05659.

Harkous, H. – Groves, I. – Saffari, A. Have Your Text and Use It Too! End-to-End Neural Data-to-Text Generation with Semantic Fidelity. *arXiv:2004.06577 [cs]*. 2020. Available at: http://arxiv.org/abs/2004.06577.

Harris, Z. S. Distributional structure. *Word*. 1954, 10, 2-3, p. 146–162.

Hedderich, M. A. – Lange, L. – Adel, H. – Strötgen, J. – Klakow, D. A Survey on Recent Approaches for Natural Language Processing in Low-Resource Scenarios. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, p. 2545–2568, 2021.

Heidari, P. – Einolghozati, A. – Jain, S. – Batra, S. – Callender, L. – Arun, A. – Mei, S. – Gupta, S. – Donmez, P. – Bhardwaj, V. – Kumar, A. – White, M. Getting to Production with Few-shot Natural Language Generation Models. In Li, H. – Levow, G. – Yu, Z. – Gupta, C. – Sisman, B. – Cai, S. – Vandyke, D. – Dethlefs, N. – Wu, Y. – Li, J. J. (Ed.) *Proceedings of the 22nd Annual Meeting of the Special Interest Group on Discourse and Dialogue, SIGdial 2021*, p. 66–76, Singapore/Online, 2021. Available at: https://aclanthology.org/2021.sigdial-1.8.

Hendrycks, D. – Gimpel, K. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*. 2016.

Hochreiter, S. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*. 1998, 6, 02, p. 107–116.

Hochreiter, S. – Schmidhuber, J. Long short-term memory. *Neural computation*. 1997, 9, 8, p. 1735–1780.

Hockenmaier, J. – Steedman, M. CCGbank: a corpus of CCG derivations and dependency structures extracted from the Penn Treebank. *Computational Linguistics*. 2007, 33, 3, p. 355–396.

Hoffmann, J. – Borgeaud, S. – Mensch, A. – Buchatskaya, E. – Cai, T. – Rutherford, E. – Casas, D. d. L. – Hendricks, L. A. – Welbl, J. – Clark, A. – others. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*. 2022.

Holtzman, A. – Buys, J. – Du, L. – Forbes, M. – Choi, Y. The Curious Case of Neural Text Degeneration. In *International Conference on Learning Representations*, 2019.

Holtzman, A. – West, P. – Zettlemoyer, L. Generative Models as a Complex Systems Science: How Can We Make Sense of Large Language Model Behavior? 2023. Available at: https://arxiv.org/pdf/2308.00189.pdf.

Hooker, S. The hardware lottery. *Communications of the ACM*. 2021, 64, 12, p. 58–65.

Hornik, K. – Stinchcombe, M. – White, H. Multilayer feedforward networks are universal approximators. *Neural networks*. 1989, 2, 5, p. 359–366.

Hoyle, A. M. – Marasović, A. – Smith, N. A. Promoting Graph Awareness in Linearized Graph-to-Text Generation. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, p. 944–956, 2021.

Jiang, A. Q. et al. Mistral 7B. *CoRR*. 2023, abs/2310.06825. doi: 10.48550/ARXIV.2310.06825. Available at: https://doi.org/10.48550/arXiv.2310.06825.

Jin, Z. – Guo, Q. – Qiu, X. – Zhang, Z. GenWiki: A Dataset of 1.3 Million Content-Sharing Text and Graphs for Unsupervised Graph-to-Text Generation. In *Proceedings of the 28th International Conference on Computational Linguistics*, p. 2398–2409, Barcelona, Spain (Online), 2020. International Committee on Computational Linguistics. Available at: https://www.aclweb.org/anthology/2020.coling-main.217.

Johnson, W. Studies in language behavior: A program of research. *Psychological Monographs*. 1944, 56, 2, p. 1–15.

Jurafsky, D. – Martin, J. H. *Speech and Language Processing (3rd ed. draft)*. Draft available online, 2024. https://web.stanford.edu/~jurafsky/slp3/.

Juraska, J. – Karagiannis, P. – Bowden, K. – Walker, M. A Deep Ensemble Model with Slot Alignment for Sequence-to-Sequence Natural Language Generation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, p. 152–162, New Orleans, LA, USA, jun 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-1014. Available at: https://www.aclweb.org/anthology/N18-1014.

Kale, M. Text-to-Text Pre-Training for Data-to-Text Tasks. *arXiv:2005.10433 [cs]*. 2020. Available at: http://arxiv.org/abs/2005.10433.

Kale, M. – Rastogi, A. Template Guided Text Generation for Task-Oriented Dialogue. In Webber, B. – Cohn, T. – He, Y. – Liu, Y. (Ed.) *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020*, p. 6505–6520, Online, 2020. doi: 10.18653/v1/2020.emnlp-main.527. Available at: https://doi.org/10.18653/v1/2020.emnlp-main.527.

Kane, H. – Kocyigit, M. Y. – Abdalla, A. – Ajanoh, P. – Coulibali, M. NUBIA: NeUral Based Interchangeability Assessor for Text Generation, 2020. Available at: http://arxiv.org/abs/2004.14667.

Kann, K. – Ebrahimi, A. – Koh, J. – Dudy, S. – Roncone, A. Open-domain dialogue generation: What we can do, cannot do, and should do next. In *Proceedings of the 4th Workshop on NLP for Conversational AI*, p. 148–165, 2022.

Kantharaj, S. – Leong, R. T. K. – Lin, X. – Masry, A. – Thakkar, M. – Hoque, E. – Joty, S. Chart-to-Text: A Large-Scale Benchmark for Chart Summarization, 2022. Available at: http://arxiv.org/abs/2203.06486.

Kaplan, J. – McCandlish, S. – Henighan, T. – Brown, T. B. – Chess, B. – Child, R. – Gray, S. – Radford, A. – Wu, J. – Amodei, D. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*. 2020.

Karpathy, A. The unreasonable effectiveness of recurrent neural networks. http://karpathy.github.io/2015/05/21/rnn-effectiveness/, 2015. Accessed on April 20, 2024.

Kasner, Z. – Dušek, O. Data-to-Text Generation with Iterative Text Editing. In Davis, B. – Graham, Y. – Kelleher, J. – Sripada, Y. (Ed.) *Proceedings of the 13th International Conference on Natural Language Generation*, p. 60–67, Dublin, Ireland, December 2020a. Association for Computational Linguistics. doi: 10.18653/v1/2020.inlg-1.9. Available at: https://aclanthology.org/2020.inlg-1.9.

Kasner, Z. – Dušek, O. Beyond Traditional Benchmarks: Analyzing Behaviors of Open LLMs on Data-to-Text Generation. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2024. Available at: http://arxiv.org/abs/2401.10186. To appear.

Kasner, Z. – Dušek, O. Train Hard, Finetune Easy: Multilingual Denoising for RDF-to-Text Generation. In Castro Ferreira, T. – Gardent, C. – Ilinykh, N. – Lee, C. – Mille, S. – Moussallem, D. – Shimorina, A. (Ed.) *Proceedings of the 3rd International Workshop on Natural Language Generation from the Semantic Web (WebNLG+)*, p. 171–176, Dublin, Ireland (Virtual), 12 2020b. Association for Computational Linguistics. Available at: https://aclanthology.org/2020.webnlg-1.20.

Kasner, Z. – Dušek, O. Neural Pipeline for Zero-Shot Data-to-Text Generation. In Muresan, S. – Nakov, P. – Villavicencio, A. (Ed.) *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, p. 3914–3932, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.acl-long.271. Available at: https://aclanthology.org/2022.acl-long.271.

Kasner, Z. – Mille, S. – Dušek, O. Text-in-Context: Token-Level Error Detection for Table-to-Text Generation. In *Proceedings of the 14th International Conference on Natural Language Generation*, p. 259–265, Aberdeen, Scotland, UK, 2021. Association for Computational Linguistics. Available at: https://aclanthology.org/2021.inlg-1.25.

Kasner, Z. – Garanina, E. – Plátek, O. – Dušek, O. TabGenie: A Toolkit for Table-to-Text Generation. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*, p. 444–455, 2023a.

Kasner, Z. – Konstas, I. – Dušek, O. Mind the Labels: Describing Relations in Knowledge Graphs With Pretrained Models. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, p. 2398–2415, 2023b.

Ke, P. – Ji, H. – Ran, Y. – Cui, X. – Wang, L. – Song, L. – Zhu, X. – Huang, M. JointGT: Graph-Text Joint Representation Learning for Text Generation from Knowledge Graphs. *arXiv:2106.10502 [cs]*. 2021. Available at: http://arxiv.org/abs/2106.10502.

Kedzie, C. – McKeown, K. A Good Sample is Hard to Find: Noise Injection Sampling and Self-Training for Neural Language Generation Models. In *Proceedings of the 12th International Conference on Natural Language Generation*, p. 584–593, Tokyo, Japan, oct 2019. Association for Computational Linguistics. doi: 10.18653/v1/W19-8672. Available at: https://www.aclweb.org/anthology/W19-8672.

Kehrer, J. – Hauser, H. Visualization and Visual Analysis of Multifaceted Scientific Data: A Survey. *IEEE Trans. Vis. Comput. Graph.* 2013, 19, 3, p. 495–513. doi: 10.1109/TVCG.2012.110. Available at: https://doi.org/10.1109/TVCG.2012.110.

Kelley, H. J. Gradient theory of optimal flight paths. *Ars Journal*. 1960, 30, 10, p. 947–954.

Kingma, D. P. – Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*. 2014.

Kirkpatrick, J. – Pascanu, R. – Rabinowitz, N. – Veness, J. – Desjardins, G. – Rusu, A. A. – Milan, K. – Quan, J. – Ramalho, T. – Grabska-Barwinska, A. – others. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*. 2017, 114, 13, p. 3521–3526.

Kocmi, T. – Federmann, C. GEMBA-MQM: Detecting Translation Quality Error Spans with GPT-4. In Koehn, P. – Haddon, B. – Kocmi, T. – Monz, C. (Ed.) *Proceedings of the Eighth Conference on Machine Translation, WMT 2023*, p. 768–775, Singapore, 2023a. Available at: https://aclanthology.org/2023.wmt-1.64.

Kocmi, T. – Federmann, C. Large Language Models Are State-of-the-Art Evaluators of Translation Quality. In Nurminen, M. et al. (Ed.) *Proceedings of the 24th Annual Conference of the European Association for Machine Translation, EAMT 2023*, p. 193–203, Tampere, Finland, 2023b. Available at: https://aclanthology.org/2023.eamt-1.19.

Kocmi, T. – Federmann, C. – Grundkiewicz, R. – Junczys-Dowmunt, M. – Matsushita, H. – Menezes, A. To Ship or Not to Ship: An Extensive Evaluation of Automatic Metrics for Machine Translation. *arXiv:2107.10821 [cs]*. 2021. Available at: http://arxiv.org/abs/2107.10821.

Koncel-Kedziorski, R. – Bekal, D. – Luan, Y. – Lapata, M. – Hajishirzi, H. Text Generation from Knowledge Graphs with Graph Transformers. *arXiv:1904.02342 [cs]*. 2019. Available at: http://arxiv.org/abs/1904.02342.

Konstas, I. – Lapata, M. Concept-to-text generation via discriminative reranking. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, p. 369–378, 2012.

Koo, R. – Lee, M. – Raheja, V. – Park, J. I. – Kim, Z. M. – Kang, D. Benchmarking Cognitive Biases in Large Language Models as Evaluators, 2023. Available at: http://arxiv.org/abs/2309.17012.

Koto, F. – Lau, J. H. – Baldwin, T. Can Pretrained Language Models Generate Persuasive, Faithful, and Informative Ad Text for Product Descriptions? In *Proceedings of the Fifth Workshop on E-Commerce and NLP (ECNLP 5)*, p. 234–243, Dublin, Ireland, 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.ecnlp-1.27. Available at: https://aclanthology.org/2022.ecnlp-1.27.

Krishna, R. – Zhu, Y. – Groth, O. – Johnson, J. – Hata, K. – Kravitz, J. – Chen, S. – Kalantidis, Y. – Li, L. – Shamma, D. A. – Bernstein, M. S. – Fei-Fei, L. Visual Genome: Connecting Language and Vision Using Crowdsourced Dense Image Annotations. *Int. J. Comput. Vis.* 2017, 123, 1, p. 32–73. doi: 10.1007/s11263-016-0981-7. Available at: https://doi.org/10.1007/s11263-016-0981-7.

Kudo, T. – Richardson, J. 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*, p. 66–71, Brussels, Belgium, 2018. doi: 10.18653/v1/D18-2012.

Laha, A. – Jain, P. – Mishra, A. – Sankaranarayanan, K. Scalable Micro-planned Generation of Discourse from Structured Data. *Comput. Linguistics.* 2019, 45, 4, p. 737–763. doi: 10.1162/coli_a_00363. Available at: https://doi.org/10.1162/coli_a_00363.

Langkilde, I. Forest-based statistical sentence generation. In *1st Meeting of the North American Chapter of the Association for Computational Linguistics*, 2000.

Lapata, M. Probabilistic Text Structuring: Experiments with Sentence Ordering. In Hinrichs, E. W. – Roth, D. (Ed.) *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, p. 545–552, Sapporo, Japan, 2003. doi: 10.3115/1075096.1075165. Available at: https://aclanthology.org/P03-1069/.

Lebret, R. – Grangier, D. – Auli, M. Neural Text Generation from Structured Data with Application to the Biography Domain. In *EMNLP*, 2016. doi: 10.18653/v1/D16-1128.

LeCun, Y. – Bengio, Y. – Hinton, G. Deep learning. *nature.* 2015, 521, 7553, p. 436–444.

Lee, C. – Cheng, H. – Ostendorf, M. Dialogue State Tracking with a Language Model using Schema-Driven Prompting. In Moens, M. – Huang, X. – Specia, L. – Yih, S. W. (Ed.) *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021*, p. 4937–4949. Association for Computational Linguistics, 2021. doi: 10.18653/v1/2021.emnlp-main.404. Available at: https://doi.org/10.18653/v1/2021.emnlp-main.404.

Lee, K. – He, L. – Zettlemoyer, L. Higher-Order Coreference Resolution with Coarse-to-Fine Inference. In Walker, M. A. – Ji, H. – Stent, A. (Ed.) *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT, Volume 2 (Short Papers)*, p. 687–692, New Orleans, LA, USA, 2018. doi: 10.18653/v1/n18-2108. Available at: https://doi.org/10.18653/v1/n18-2108.

Lee, N. – Ping, W. – Xu, P. – Patwary, M. – Shoeybi, M. – Catanzaro, B. Factuality Enhanced Language Models for Open-Ended Text Generation, 2022. Available at: http://arxiv.org/abs/2206.04624.

Lehmann, J. – Isele, R. – Jakob, M. – Jentzsch, A. – Kontokostas, D. – Mendes, P. N. – Hellmann, S. – Morsey, M. – Kleef, P. – Auer, S. – Bizer, C. DBpedia - A large-scale, multilingual knowledge base extracted from Wikipedia. *Semantic Web*. 2015, 6, 2, p. 167–195. doi: 10.3233/SW-140134. Available at: https://doi.org/10.3233/SW-140134.

Leppänen, L. – Munezero, M. – Granroth-Wilding, M. – Toivonen, H. Data-driven news generation for automated journalism. In *Proceedings of the 10th international conference on natural language generation*, p. 188–197, 2017.

Lewis, M. – Liu, Y. – Goyal, N. – Ghazvininejad, M. – Mohamed, A. – Levy, O. – Stoyanov, V. – Zettlemoyer, L. BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension. *arXiv:1910.13461 [cs, stat]*. 2019. Available at: http://arxiv.org/abs/1910.13461.

Lhoest, Q. et al. Datasets: A Community Library for Natural Language Processing. In Adel, H. – Shi, S. (Ed.) *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, EMNLP 2021, Online and Punta Cana, Dominican Republic, 7-11 November, 2021*, p. 175–184, 2021. doi: 10.18653/v1/2021.emnlp-demo.21. Available at: https://doi.org/10.18653/v1/2021.emnlp-demo.21.

Li, J. – Galley, M. – Brockett, C. – Gao, J. – Dolan, W. B. A Diversity-Promoting Objective Function for Neural Conversation Models. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, p. 110–119, 2016.

Li, S. – Han, C. – Yu, P. – Edwards, C. – Li, M. – Wang, X. – Fung, Y. – Yu, C. – Tetreault, J. – Hovy, E. – others. Defining a New NLP Playground. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, p. 11932–11951, 2023.

Liang, P. – Jordan, M. I. – Klein, D. Learning Semantic Correspondences with Less Supervision. In Su, K. – Su, J. – Wiebe, J. (Ed.) *ACL 2009, Proceedings of the 47th Annual Meeting of the Association for Computational Linguistics and the 4th International Joint Conference on Natural Language Processing of the AFNLP, 2-7 August 2009*, p. 91–99, Singapore, 2009. Available at: `https://aclanthology.org/P09-1011/`.

Lin, B. Y. – Shen, M. – Xing, Y. – Zhou, P. – Ren, X. CommonGen: A Constrained Text Generation Dataset Towards Generative Commonsense Reasoning. *CoRR*. 2019, abs/1911.03705. Available at: `http://arxiv.org/abs/1911.03705`.

Lin, C.-Y. ROUGE: A Package for Automatic Evaluation of Summaries. In *Text Summarization Branches Out*, p. 74–81, Barcelona, Spain, jul 2004. Association for Computational Linguistics. Available at: `https://www.aclweb.org/anthology/W04-1013`.

Lin, Y. – Ruan, T. – Liu, J. – Wang, H. A survey on neural data-to-text generation. *IEEE Transactions on Knowledge and Data Engineering*. 2023.

Liu, X. – He, P. – Chen, W. – Gao, J. Multi-Task Deep Neural Networks for Natural Language Understanding. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, p. 4487–4496, Florence, Italy, July 2019a. doi: 10.18653/v1/P19-1441. Available at: `https://www.aclweb.org/anthology/P19-1441`.

Liu, Y. – Ott, M. – Goyal, N. – Du, J. – Joshi, M. – Chen, D. – Levy, O. – Lewis, M. – Zettlemoyer, L. – Stoyanov, V. RoBERTa: A Robustly Optimized BERT Pretraining Approach. *arXiv:1907.11692 [cs]*. 2019b. Available at: `http://arxiv.org/abs/1907.11692`.

Liu, Y. – Gu, J. – Goyal, N. – Li, X. – Edunov, S. – Ghazvininejad, M. – Lewis, M. – Zettlemoyer, L. Multilingual Denoising Pre-training for Neural Machine Translation. *arXiv:2001.08210 [cs]*. 2020. Available at: `http://arxiv.org/abs/2001.08210`.

Loshchilov, I. – Hutter, F. Fixing Weight Decay Regularization in Adam. In *International Conference on Learning Representations (ICLR)*, New Orleans, LA, USA, 2019. Available at: `https://openreview.net/forum?id=Bkg6RiCqY7`.

Mairesse, F. – Gasic, M. – Jurcicek, F. – Keizer, S. – Thomson, B. – Yu, K. – Young, S. Phrase-based statistical language generation using graphical models and active learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, p. 1552–1561, 2010.

Mallinson, J. – Severyn, A. – Malmi, E. – Garrido, G. Felix: Flexible Text Editing Through Tagging and Insertion. *arXiv:2003.10687 [cs]*. 2020. Available at: `http://arxiv.org/abs/2003.10687`.

Malmi, E. – Krause, S. – Rothe, S. – Mirylenka, D. – Severyn, A. Encode, Tag, Realize: High-Precision Text Editing. *CoRR*. 2019, abs/1909.01187. Available at: `http://arxiv.org/abs/1909.01187`.

Malmi, E. – Dong, Y. – Mallinson, J. – Chuklin, A. – Adamek, J. – Mirylenka, D. – Stahlberg, F. – Krause, S. – Kumar, S. – Severyn, A. Text generation with text-editing models. *arXiv preprint arXiv:2206.07043*. 2022.

Mann, W. Text Generation. *American Journal of Computational Linguistics*. 1982, 8, 2, p. 62–69. Available at: https://aclanthology.org/J82-2003.

Marcheggiani, D. – Perez-Beltrachini, L. Deep Graph Convolutional Encoders for Structured Data to Text Generation. *arXiv:1810.09995 [cs]*. 2018. Available at: http://arxiv.org/abs/1810.09995.

Martin, A. – Przybocki, M. The NIST 1999 speaker recognition evaluation—An overview. *Digital signal processing*. 2000, 10, 1-3, p. 1–18.

Mathur, N. – Baldwin, T. – Cohn, T. Tangled up in BLEU: Reevaluating the Evaluation of Automatic Machine Translation Evaluation Metrics. *arXiv:2006.06264 [cs]*. 2020. Available at: http://arxiv.org/abs/2006.06264.

Matthiessen, C. Lexico (grammatical) choice in text generation. 1991, p. 249–292.

Maynez, J. – Narayan, S. – Bohnet, B. – McDonald, R. On Faithfulness and Factuality in Abstractive Summarization. *arXiv:2005.00661 [cs]*. 2020. Available at: http://arxiv.org/abs/2005.00661.

McCloskey, M. – Cohen, N. J. *Catastrophic interference in connectionist networks: The sequential learning problem*. 24. Elsevier, 1989.

McCoy, T. – Pavlick, E. – Linzen, T. Right for the Wrong Reasons: Diagnosing Syntactic Heuristics in Natural Language Inference. In Korhonen, A. – Traum, D. R. – Màrquez, L. (Ed.) *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, p. 3428–3448. Association for Computational Linguistics, 2019. doi: 10.18653/v1/p19-1334. Available at: https://doi.org/10.18653/v1/p19-1334.

McDonald, D. A Framework for Writing Generation Grammars for Interactive Computer Programs. *American Journal of Computational Linguistics*. November 1975, p. 4–17. Available at: https://aclanthology.org/J75-4016. Microfiche 33.

McKeown, K. *Text generation*. Cambridge University Press, 1985.

Meehan, J. R. Using Planning Structures to Generate Stories. *American Journal of Computational Linguistics*. November 1975, p. 78–94. Available at: https://aclanthology.org/J75-4021. Microfiche 33.

Mehta, S. V. – Rao, J. – Tay, Y. – Kale, M. – Parikh, A. – Zhong, H. – Strubell, E. Improving Compositional Generalization with Self-Training for Data-to-Text Generation. *CoRR*. 2021, abs/2110.08467. Available at: https://arxiv.org/abs/2110.08467.

Mei, H. – Bansal, M. – Walter, M. R. What to talk about and how? Selective Generation using LSTMs with Coarse-to-Fine Alignment. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, p. 720–730, 2016.

Mel'cuk, I. A. – others. *Dependency syntax: theory and practice.* SUNY press, 1988.

Michael, J. – Holtzman, A. – Parrish, A. – Mueller, A. – Wang, A. – Chen, A. – Madaan, D. – Nangia, N. – Pang, R. Y. – Phang, J. – others. What Do NLP Researchers Believe? Results of the NLP Community Metasurvey. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, p. 16334–16368, 2023.

Mikolov, T. – Sutskever, I. – Chen, K. – Corrado, G. S. – Dean, J. Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems.* 2013, 26.

Mille, S. – Dasiopoulou, S. – Fisas, B. – Wanner, L. Teaching FORGe to verbalize DBpedia properties in Spanish. In *Proceedings of the 12th International Conference on Natural Language Generation*, p. 473–483, 2019.

Mille, S. – Dhole, K. D. – Mahamood, S. – Perez-Beltrachini, L. – Gangal, V. – Kale, M. S. – Miltenburg, E. – Gehrmann, S. Automatic Construction of Evaluation Suites for Natural Language Generation Datasets. In Vanschoren, J. – Yeung, S. (Ed.) *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks 1, NeurIPS Datasets and Benchmarks 2021, December 2021, virtual*, 2021. Available at: https://datasets-benchmarks-proceedings.neurips.cc/paper/2021/hash/ec8956637a99787bd197eacd77acce5e-Abstract-round1.html.

Mille, S. – Lareau, F. – Dasiopoulou, S. – Belz, A. Mod-D2T: A Multi-layer Dataset for Modular Data-to-Text Generation. In *Proceedings of the 16th International Natural Language Generation Conference*, p. 455–466, Prague, Czechia, 2023. Association for Computational Linguistics. Available at: https://aclanthology.org/2023.inlg-main.36.

Mittal, V. O. – Moore, J. D. – Carenini, G. – Roth, S. Describing Complex Charts in Natural Language: A Caption Generation System. *Computational Linguistics.* 1998, 24, p. 431–467. Available at: https://aclanthology.org/J98-3004.

Moosavi, N. S. – Rücklé, A. – Roth, D. – Gurevych, I. Learning to Reason for Text Generation from Scientific Tables. *arXiv:2104.08296 [cs].* 2021. Available at: http://arxiv.org/abs/2104.08296.

Moryossef, A. – Goldberg, Y. – Dagan, I. Improving Quality and Efficiency in Plan-based Neural Data-to-text Generation. In Deemter, K. – Lin, C. – Takamura, H. (Ed.) *Proceedings of the 12th International Conference on Natural Language Generation, INLG 2019*, p. 377–382, Tokyo, Japan, 2019a. doi: 10.18653/v1/W19-8645. Available at: https://aclanthology.org/W19-8645/.

MORYOSSEF, A. – GOLDBERG, Y. – DAGAN, I. Step-by-Step: Separating Planning from Realization in Neural Data-to-Text Generation. In BURSTEIN, J. – DORAN, C. – SOLORIO, T. (Ed.) *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Volume 1 (Long and Short Papers)*, p. 2267–2277, Minneapolis, MN, USA, 2019b. doi: 10.18653/v1/n19-1236. Available at: https://doi.org/10.18653/v1/n19-1236.

MURAKAMI, S. – WATANABE, A. – MIYAZAWA, A. – GOSHIMA, K. – YANASE, T. – TAKAMURA, H. – MIYAO, Y. Learning to generate market comments from stock prices. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, p. 1374–1384, 2017.

NAIR, V. – HINTON, G. E. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, p. 807–814, 2010.

NAN, L. – RADEV, D. – ZHANG, R. – RAU, A. – SIVAPRASAD, A. – HSIEH, C. – TANG, X. – VYAS, A. – VERMA, N. – KRISHNA, P. – OTHERS. DART: Open-Domain Structured Data Record to Text Generation. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, p. 432–447, 2021.

NARAYAN, S. – GARDENT, C. – COHEN, S. B. – SHIMORINA, A. Split and Rephrase. In PALMER, M. – HWA, R. – RIEDEL, S. (Ed.) *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017*, p. 606–616, Copenhagen, Denmark, 2017. doi: 10.18653/v1/d17-1064. Available at: https://doi.org/10.18653/v1/d17-1064.

NEERAJA, J. – GUPTA, V. – SRIKUMAR, V. Incorporating External Knowledge to Enhance Tabular Reasoning. In TOUTANOVA, K. – RUMSHISKY, A. – ZETTLEMOYER, L. – HAKKANI-TÜR, D. – BELTAGY, I. – BETHARD, S. – COTTERELL, R. – CHAKRABORTY, T. – ZHOU, Y. (Ed.) *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2021, Online, June 6-11, 2021*, p. 2799–2809. Association for Computational Linguistics, 2021. doi: 10.18653/v1/2021.naacl-main.224. Available at: https://doi.org/10.18653/v1/2021.naacl-main.224.

NIE, F. – YAO, J.-G. – WANG, J. – PAN, R. – LIN, C.-Y. A Simple Recipe towards Reducing Hallucination in Neural Surface Realisation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, p. 2673–2679, Florence, Italy, 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1256. Available at: https://www.aclweb.org/anthology/P19-1256.

NOVIKOFF, A. B. On convergence proofs on perceptrons. In *Proceedings of the Symposium on the Mathematical Theory of Automata*, 12, p. 615–622. New York, NY, 1962.

Obeid, J. – Hoque, E. Chart-to-Text: Generating Natural Language Descriptions for Charts by Adapting the Transformer Model. *arXiv:2010.09142 [cs]*. 2020. Available at: http://arxiv.org/abs/2010.09142.

Oh, A. – Rudnicky, A. Stochastic language generation for spoken dialogue systems. In *ANLP-NAACL 2000 Workshop: Conversational Systems*, 2000.

OpenAI. Introducing ChatGPT. https://openai.com/blog/chatgpt, 2023a. Accessed on April 20, 2024.

OpenAI. GPT-4 Technical Report. *CoRR*. 2023b, abs/2303.08774. doi: 10.48550/ARXIV.2303.08774. Available at: https://doi.org/10.48550/arXiv.2303.08774.

Oremus, W. The first news report on the LA earthquake was written by a robot. *Slate. com*. 2014, 17.

Ott, M. – Edunov, S. – Baevski, A. – Fan, A. – Gross, S. – Ng, N. – Grangier, D. – Auli, M. fairseq: A Fast, Extensible Toolkit for Sequence Modeling. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, p. 48–53, Minneapolis, MN, USA, 2019. doi: 10.18653/v1/N19-4009.

Ouyang, L. et al. Training Language Models to Follow Instructions With Human Feedback. In *NeurIPS*, 2022. Available at: http://papers.nips.cc/paper_files/paper/2022/hash/b1efde53be364a73914f58805a001731-Abstract-Conference.html.

Papert, S. A. The summer vision project. *Massachusetts Institute of Technology, Project MAC*. 1966.

Papineni, K. – Roukos, S. – Ward, T. – Zhu, W.-J. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, p. 311–318, 2002.

Parikh, A. P. – Wang, X. – Gehrmann, S. – Faruqui, M. – Dhingra, B. – Yang, D. – Das, D. ToTTo: A Controlled Table-To-Text Generation Dataset. *arXiv:2004.14373 [cs]*. 2020. Available at: http://arxiv.org/abs/2004.14373.

Pascanu, R. – Mikolov, T. – Bengio, Y. On the difficulty of training recurrent neural networks. In *International conference on machine learning*, p. 1310–1318. Pmlr, 2013.

Paszke, A. et al. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In Wallach, H. M. – Larochelle, H. – Beygelzimer, A. – d'Alché-Buc, F. – Fox, E. B. – Garnett, R. (Ed.) *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, p. 8024–8035, 2019. Available at: https://proceedings.neurips.cc/paper/2019/hash/bdbca288fee7f92f2bfa9f7012727740-Abstract.html.

Perlitz, Y. – Ein-Dor, L. – Sheinwald, D. – Slonim, N. – Shmueli-Scheuer, M. Diversity Enhanced Table-to-Text Generation via Type Control, 2022. Available at: http://arxiv.org/abs/2205.10938.

Peters, M. E. – Neumann, M. – Iyyer, M. – Gardner, M. – Clark, C. – Lee, K. – Zettlemoyer, L. Deep contextualized word representations, 2018.

Petroni, F. – Rocktäschel, T. – Riedel, S. – Lewis, P. – Bakhtin, A. – Wu, Y. – Miller, A. Language Models as Knowledge Bases? In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, p. 2463–2473, 2019.

Ponnamperuma, K. – Siddharthan, A. – Zeng, C. – Mellish, C. – Van Der Wal, R. Tag2blog: Narrative generation from satellite tag data. In *Proceedings of the 51st annual meeting of the association for computational linguistics: System demonstrations*, p. 169–174, 2013.

Popović, M. chrF: character n-gram F-score for automatic MT evaluation. In Bojar, O. – Chatterjee, R. – Federmann, C. – Haddow, B. – Hokamp, C. – Huck, M. – Logacheva, V. – Pecina, P. (Ed.) *Proceedings of the Tenth Workshop on Statistical Machine Translation*, p. 392–395, Lisbon, Portugal, September 2015. Association for Computational Linguistics. doi: 10.18653/v1/W15-3049. Available at: https://aclanthology.org/W15-3049.

Popović, M. chrF++: words helping character n-grams. In *Proceedings of the second conference on machine translation*, p. 612–618, 2017.

Portet, F. – Reiter, E. – Gatt, A. – Hunter, J. – Sripada, S. – Freer, Y. – Sykes, C. Automatic generation of textual summaries from neonatal intensive care data. *Artificial Intelligence*. 2009a, 173, 7-8, p. 789–816.

Portet, F. – Reiter, E. – Gatt, A. – Hunter, J. – Sripada, S. – Freer, Y. – Sykes, C. Automatic Generation of Textual Summaries from Neonatal Intensive Care Data. *Artificial Intelligence*. 2009b, 173, p. 789–816. ISSN 00043702. doi: 10.1016/j.artint.2008.12.002. Available at: https://linkinghub.elsevier.com/retrieve/pii/S0004370208002117.

Post, M. A Call for Clarity in Reporting BLEU Scores. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, p. 186. Association for Computational Linguistics, 2018.

Puduppully, R. – Lapata, M. Data-to-Text Generation with Macro Planning. *arXiv:2102.02723 [cs]*. 2021. Available at: http://arxiv.org/abs/2102.02723.

Puduppully, R. – Dong, L. – Lapata, M. Data-to-Text Generation with Entity Modeling. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, p. 2023–2035, Florence, Italy, 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1195. Available at: https://www.aclweb.org/anthology/P19-1195.

Radford, A. – Wu, J. – Child, R. – Luan, D. – Amodei, D. – Sutskever, I. Language Models Are Unsupervised Multitask Learners. 2019a, p. 24.

Radford, A. – Wu, J. – Child, R. – Luan, D. – Amodei, D. – Sutskever, I. – others. Language models are unsupervised multitask learners. *OpenAI blog*. 2019b, 1, 8, p. 9.

Raffel, C. – Shazeer, N. – Roberts, A. – Lee, K. – Narang, S. – Matena, M. – Zhou, Y. – Li, W. – Liu, P. J. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *arXiv:1910.10683 [cs, stat]*. 2019. Available at: http://arxiv.org/abs/1910.10683.

Rastogi, A. – Zang, X. – Sunkara, S. – Gupta, R. – Khaitan, P. Towards Scalable Multi-Domain Conversational Agents: The Schema-Guided Dialogue Dataset. *Proceedings of the AAAI Conference on Artificial Intelligence*. 2020, 34, p. 8689–8696. ISSN 2374-3468. doi: 10.1609/aaai.v34i05.6394. Available at: https://ojs.aaai.org/index.php/AAAI/article/view/6394.

Ratnaparkhi, A. Trainable Methods for Surface Natural Language Generation. In *1st Meeting of the North American Chapter of the Association for Computational Linguistics*, 2000.

Rebuffel, C. – Roberti, M. – Soulier, L. – Scoutheeten, G. – Cancelliere, R. – Gallinari, P. Controlling hallucinations at word level in data-to-text generation. *Data Mining and Knowledge Discovery*. 2021, p. 1–37.

Reimers, N. – Gurevych, I. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, p. 3982–3992, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1410. Available at: https://www.aclweb.org/anthology/D19-1410.

Reiter, E. NLG vs Templates: Levels of Sophistication in Generating Text. *URL: https://ehu-dreiter. com/2016/12/18/nlg-vs-templates*. 2016.

Reiter, E. Academic NLG should not fixate on end-to-end neural. https://ehudreiter.com/2020/12/01/dont-fixate-on-end-to-end-neural/, 2020. Accessed on March 08, 2024.

Reiter, E. An Architecture for Data-to-Text Systems. In *Proceedings of the Eleventh European Workshop on Natural Language Generation (ENLG 07)*, p. 97–104, Saarbrücken, Germany, 2007. DFKI GmbH. Available at: https://aclanthology.org/W07-2315.

Reiter, E. – Dale, R. Building Applied Natural Language Generation Systems. *Nat. Lang. Eng*. 1997, 3, p. 57–87. ISSN 13513249. doi: 10.1017/S1351324997001502. Available at: http://www.journals.cambridge.org/abstract_S1351324997001502.

Reiter, E. – Thomson, C. Shared Task on Evaluating Accuracy. In *Proceedings of the 13th International Conference on Natural Language Generation*, p. 227–231, Dublin, Ireland, 2020. Association for Computational Linguistics. Available at: https://www.aclweb.org/anthology/2020.inlg-1.28.

Reiter, E. – Robertson, R. – Osman, L. M. Lessons from a failure: Generating tailored smoking cessation letters. *Artificial Intelligence*. 2003, 144, 1-2, p. 41–58.

Rony, M. R. A. H. – Kovriguina, L. – Chaudhuri, D. – Usbeck, R. – Lehmann, J. RoMe: A Robust Metric for Evaluating Natural Language Generation. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, p. 5645–5657, Dublin, Ireland, 2022. Association for Computational Linguistics. Available at: https://aclanthology.org/2022.acl-long.387.

Rosenblatt, F. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*. 1958, 65, 6, p. 386.

Rumelhart, D. E. – Hinton, G. E. – Williams, R. J. Learning representations by back-propagating errors. *nature*. 1986, 323, 6088, p. 533–536.

Saha, S. – Yu, X. V. – Bansal, M. – Pasunuru, R. – Celikyilmaz, A. MURMUR: Modular Multi-Step Reasoning for Semi-Structured Data-to-Text Generation, 2022. Available at: http://arxiv.org/abs/2212.08607.

Salehinejad, H. – Sankar, S. – Barfett, J. – Colak, E. – Valaee, S. Recent advances in recurrent neural networks. *arXiv preprint arXiv:1801.01078*. 2017.

Sanh, V. – Webson, A. – Raffel, C. – Bach, S. H. – Sutawika, L. – Alyafeai, Z. – Chaffin, A. – Stiegler, A. – Scao, T. L. – Raja, A. – others. Multitask prompted training enables zero-shot task generalization. *arXiv preprint arXiv:2110.08207*. 2021.

Scao, T. L. et al. BLOOM: A 176b-Parameter Open-Access Multilingual Language Model. *CoRR*. 2022, abs/2211.05100. doi: 10.48550/arXiv.2211.05100. Available at: https://doi.org/10.48550/arXiv.2211.05100.

Scott, D. – Hallett, C. – Fettiplace, R. Data-to-text summarisation of patient records: Using computer-generated summaries to access patient histories. *Patient education and counseling*. 2013, 92, 2, p. 153–159.

See, A. – Liu, P. J. – Manning, C. D. Get To The Point: Summarization with Pointer-Generator Networks. *arXiv:1704.04368 [cs]*. 2017. Available at: http://arxiv.org/abs/1704.04368.

Sellam, T. – Das, D. – Parikh, A. BLEURT: Learning Robust Metrics for Text Generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, p. 7881–7892, 2020.

Sennrich, R. – Haddow, B. – Birch, A. 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)*, p. 1715–1725, 2016.

Shannon, C. E. A mathematical theory of communication. *The Bell system technical journal*. 1948, 27, 3, p. 379–423.

Shao, H. – Wang, J. – Lin, H. – Zhang, X. – Zhang, A. – Ji, H. – Abdelzaher, T. F. Controllable and Diverse Text Generation in E-commerce. In Leskovec, J. – Grobelnik, M. – Najork, M. – Tang, J. – Zia, L. (Ed.) *WWW '21: The Web Conference 2021*, p. 2392–2401, Virtual Event / Ljubljana, Slovenia, 2021. doi: 10.1145/3442381.3449838. Available at: https://doi.org/10.1145/3442381.3449838.

Shao, Z. – Huang, M. – Wen, J. – Xu, W. – Zhu, X. Long and Diverse Text Generation with Planning-based Hierarchical Variational Model. In Inui, K. – Jiang, J. – Ng, V. – Wan, X. (Ed.) *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019*, p. 3255–3266, Hong Kong, 2019. doi: 10.18653/v1/D19-1321. Available at: https://doi.org/10.18653/v1/D19-1321.

Sharma, M. – Gogineni, A. – Ramakrishnan, N. Innovations in Neural Data-to-text Generation, 2022. Available at: http://arxiv.org/abs/2207.12571.

Shen, X. – Chang, E. – Su, H. – Niu, C. – Klakow, D. Neural Data-to-Text Generation via Jointly Learning the Segmentation and Correspondence. In Jurafsky, D. – Chai, J. – Schluter, N. – Tetreault, J. R. (Ed.) *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020*, p. 7155–7165, Online, 2020. doi: 10.18653/v1/2020.acl-main.641. Available at: https://doi.org/10.18653/v1/2020.acl-main.641.

Sheridan, P. Research in language translation on the IBM type 701. *IBM Technical Newsletter*. 1955, 9, p. 5–24.

Shimorina, A. – Gardent, C. Handling Rare Items in Data-to-Text Generation. In *Proceedings of the 11th International Conference on Natural Language Generation*, p. 360–370, Tilburg University, The Netherlands, 2018. Association for Computational Linguistics. doi: 10.18653/v1/W18-6543. Available at: https://aclanthology.org/W18-6543.

Shimorina, A. – Gardent, C. – Narayan, S. – Perez-Beltrachini, L. WebNLG Challenge: Human Evaluation Results. 2019, p. 16.

Siddharthan, A. – Green, M. J. – Van Deemter, K. – Mellish, C. – Van Der Wal, R. Blogging birds: Generating narratives about reintroduced species to promote public engagement. In *INLG 2012 Proceedings of the Seventh International Natural Language Generation Conference*, p. 120–124, 2012.

Sottana, A. – Liang, B. – Zou, K. – Yuan, Z. Evaluation Metrics in the Era of GPT-4: Reliably Evaluating Large Language Models on Sequence to Sequence Tasks. In Bouamor, H. – Pino, J. – Bali, K. (Ed.) *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, p. 8776–8788, Singapore, 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.543. Available at: https://aclanthology.org/2023.emnlp-main.543.

Steedman, M. *The syntactic process.* MIT press, 2001.

Stefanini, M. – Cornia, M. – Baraldi, L. – Cascianelli, S. – Fiameni, G. – Cucchiara, R. From show to tell: A survey on deep learning-based image captioning. *IEEE transactions on pattern analysis and machine intelligence.* 2022, 45, 1, p. 539–559.

Stureborg, R. – Alikaniotis, D. – Suhara, Y. Large Language Models Are Inconsistent and Biased Evaluators, 2024. Available at: http://arxiv.org/abs/2405.01724.

Su, Y. – Meng, Z. – Baker, S. – Collier, N. Few-Shot Table-to-Text Generation with Prototype Memory. *arXiv:2108.12516 [cs].* 2021a. Available at: http://arxiv.org/abs/2108.12516.

Su, Y. – Vandyke, D. – Wang, S. – Fang, Y. – Collier, N. Plan-then-Generate: Controlled Data-to-Text Generation via Planning. In Moens, M. – Huang, X. – Specia, L. – Yih, S. W. (Ed.) *Findings of the Association for Computational Linguistics: EMNLP 2021*, p. 895–909, Online/Punta Cana, Dominican Republic, 2021b. doi: 10.18653/v1/2021.findings-emnlp.76. Available at: https://doi.org/10.18653/v1/2021.findings-emnlp.76.

Su, Y. – Vandyke, D. – Wang, S. – Fang, Y. – Collier, N. Plan-Then-Generate: Controlled Data-to-Text Generation via Planning. *arXiv:2108.13740 [cs].* 2021c. Available at: http://arxiv.org/abs/2108.13740.

Suadaa, L. H. – Kamigaito, H. – Funakoshi, K. – Okumura, M. – Takamura, H. Towards Table-to-Text Generation with Numerical Reasoning. In *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)*, p. 1451–1465, Online, 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-long.115. Available at: https://aclanthology.org/2021.acl-long.115.

Sun, X. – Mellish, C. Domain Independent Sentence Generation from RDF Representations for the Semantic Web. 2006, p. 7.

Sutskever, I. – Vinyals, O. – Le, Q. V. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, p. 3104–3112, 2014.

Tang, T. – Li, J. – Zhao, W. X. – Wen, J. MVP: Multi-Task Supervised Pre-Training for Natural Language Generation. *CoRR.* 2022, abs/2206.12131. doi: 10.48550/arXiv.2206.12131. Available at: https://doi.org/10.48550/arXiv.2206.12131.

Tanon, T. P. – Weikum, G. – Suchanek, F. M. YAGO 4: A Reason-able Knowledge Base. In Harth, A. – Kirrane, S. – Ngomo, A. N. – Paulheim, H. – Rula, A. – Gentile, A. L. – Haase, P. – Cochez, M. (Ed.) *The Semantic Web - 17th International Conference, ESWC 2020, Heraklion, Crete, Greece, May 31-June 4, 2020, Proceedings*, 12123 / *Lecture Notes in Computer Science*, p. 583–596. Springer, 2020. doi: 10.1007/978-3-030-49461-2_34. Available at: https://doi.org/10.1007/978-3-030-49461-2_34.

Taylor, W. L. “Cloze procedure”: A new tool for measuring readability. *Journalism quarterly*. 1953, 30, 4, p. 415–433.

Thomson, C. – Reiter, E. Generation Challenges: Results of the Accuracy Evaluation Shared Task. *arXiv:2108.05644 [cs]*. 2021. Available at: http://arxiv.org/abs/2108.05644.

Thomson, C. – Reiter, E. A Gold Standard Methodology for Evaluating Accuracy in Data-To-Text Systems. *arXiv:2011.03992 [cs]*. 2020. Available at: http://arxiv.org/abs/2011.03992.

Thomson, C. – Reiter, E. – Sripada, S. SportSett: basketball-a robust and maintainable data-set for natural language generation. In *Proceedings of the Workshop on Intelligent Information Processing and Natural Language Generation*, p. 32–40, 2020.

Thomson, C. – Reiter, E. – Sundararajan, B. Evaluating Factual Accuracy in Complex Data-to-Text. *Computer Speech & Language*. 2023, 80, p. 101482. ISSN 0885-2308. doi: 10.1016/j.csl.2023.101482. Available at: https://www.sciencedirect.com/science/article/pii/S0885230823000013.

Tian, R. – Narayan, S. – Sellam, T. – Parikh, A. P. Sticking to the Facts: Confident Decoding for Faithful Data-to-Text Generation. 2020, p. 16.

Touvron, H. et al. Llama 2: Open Foundation and Fine-Tuned Chat Models. *CoRR*. 2023, abs/2307.09288. doi: 10.48550/ARXIV.2307.09288. Available at: https://doi.org/10.48550/arXiv.2307.09288.

Trisedya, B. D. – Qi, J. – Zhang, R. Sentence Generation for Entity Description with Content-plan Attention. 2020, p. 8.

Tunstall, L. – Beeching, E. – Lambert, N. – Rajani, N. – Rasul, K. – Belkada, Y. – Huang, S. – Werra, L. – Fourrier, C. – Habib, N. – Sarrazin, N. – Sanseviero, O. – Rush, A. M. – Wolf, T. Zephyr: Direct Distillation of LM Alignment. *CoRR*. 2023, abs/2310.16944. doi: 10.48550/ARXIV.2310.16944. Available at: https://doi.org/10.48550/arXiv.2310.16944.

Lee, C. – Krahmer, E. – Wubben, S. PASS: A Dutch data-to-text system for soccer, targeted towards specific audiences. In *Proceedings of the 10th International Conference on Natural Language Generation*, p. 95–104, 2017.

van der Lee, C. – Krahmer, E. – Wubben, S. Automated Learning of Templates for Data-to-Text Generation: Comparing Rule-Based, Statistical and Neural Methods. In *Proceedings of the 11th International Conference on Natural Language Generation*, p. 35–45, Tilburg University, The Netherlands, 2018. Association for Computational Linguistics. doi: 10.18653/v1/W18-6504. Available at: https://aclanthology.org/W18-6504.

Lee, C. – Gatt, A. – Van Miltenburg, E. – Wubben, S. – Krahmer, E. Best practices for the human evaluation of automatically generated text. In *Proceedings of the 12th International Conference on Natural Language Generation*, p. 355–368, 2019.

van der Lee, C. – Emmery, C. – Wubben, S. – Krahmer, E. The CACAPO Dataset: A Multilingual, Multi-Domain Dataset for Neural Pipeline and End-to-End Data-to-Text Generation. In *Proceedings of the 13th International Conference on Natural Language Generation*, p. 68–79, Dublin, Ireland, 2020. Association for Computational Linguistics. Available at: https://www.aclweb.org/anthology/2020.inlg-1.10.

van Miltenburg, E. – Elliott, D. – Vossen, P. Measuring the Diversity of Automatic Image Descriptions. In *Proceedings of the 27th International Conference on Computational Linguistics*, p. 1730–1741, Santa Fe, New Mexico, USA, 2018. Association for Computational Linguistics. Available at: https://aclanthology.org/C18-1147.

Miltenburg, E. – Clinciu, M. – Dusek, O. – Gkatzia, D. – Inglis, S. – Leppänen, L. – Mahamood, S. – Manning, E. – Schoch, S. – Thomson, C. – Wen, L. Underreporting of Errors in NLG Output, and What to Do About It. In Belz, A. – Fan, A. – Reiter, E. – Sripada, Y. (Ed.) *Proceedings of the 14th International Conference on Natural Language Generation, INLG 2021*, p. 140–153, Aberdeen, Scotland, UK, 2021. Available at: https://aclanthology.org/2021.inlg-1.14.

Miltenburg, E. – Clinciu, M. – Dušek, O. – Gkatzia, D. – Inglis, S. – Leppänen, L. – Mahamood, S. – Schoch, S. – Thomson, C. – Wen, L. Barriers and Enabling Factors for Error Analysis in NLG Research. *Northern European Journal of Language Technology*. February 2023, 9, 1. ISSN 2000-1533. doi: 10.3384/nejlt.2000-1533.2023.4529. Available at: https://nejlt.ep.liu.se/article/view/4529. Number: 1.

Vaswani, A. – Shazeer, N. – Parmar, N. – Uszkoreit, J. – Jones, L. – Gomez, A. N. – Kaiser, Ł. – Polosukhin, I. Attention is All You Need. In *Advances in Neural Information Processing Systems 30*, p. 6000–6010, Long Beach, CA, USA, December 2017. Curran Associates, Inc.

Vejvar, M. – Fujimoto, Y. ASPIRO: Any-shot Structured Parsing-error-Induced ReprOmpting for Consistent Data-to-Text Generation, 2023. Available at: http://arxiv.org/abs/2310.17877.

Veselovsky, V. – Ribeiro, M. H. – West, R. Artificial Artificial Artificial Intelligence: Crowd Workers Widely Use Large Language Models for Text Production Tasks, 2023. Available at: http://arxiv.org/abs/2306.07899.

Vrandečić, D. – Krötzsch, M. Wikidata: a free collaborative knowledgebase. *Commun. ACM*. 2014, 57, 10, p. 78–85. doi: 10.1145/2629489. Available at: https://doi.org/10.1145/2629489.

Wang, J. – Liang, Y. – Meng, F. – Sun, Z. – Shi, H. – Li, Z. – Xu, J. – Qu, J. – Zhou, J. Is ChatGPT a Good NLG Evaluator? A Preliminary Study, 2023a. Available at: http://arxiv.org/abs/2303.04048.

Wang, J.-t. On Computational Sentence Generation From Logical Form. In *COLING 1980 Volume 1: The 8th International Conference on Computational Linguistics*, 1980. Available at: https://aclanthology.org/C80-1061.

Wang, L. – Lyu, C. – Ji, T. – Zhang, Z. – Yu, D. – Shi, S. – Tu, Z. Document-Level Machine Translation with Large Language Models. In Bouamor, H. – Pino, J. – Bali, K. (Ed.) *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, p. 16646–16661, Singapore, December 2023b. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.1036. Available at: https://aclanthology.org/2023.emnlp-main.1036.

Wang, P. – Li, L. – Chen, L. – Zhu, D. – Lin, B. – Cao, Y. – Liu, Q. – Liu, T. – Sui, Z. Large Language Models are not Fair Evaluators. *CoRR*. 2023c, abs/2305.17926. doi: 10.48550/ARXIV.2305.17926. Available at: https://doi.org/10.48550/arXiv.2305.17926.

Wang, T. – Wan, X. Hierarchical Attention Networks for Sentence Ordering. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019*, p. 7184–7191, Honolulu, HI, USA, 2019. doi: 10.1609/aaai.v33i01.33017184. Available at: https://doi.org/10.1609/aaai.v33i01.33017184.

Wang, X. – Gao, T. – Zhu, Z. – Zhang, Z. – Liu, Z. – Li, J. – Tang, J. KEPLER: A Unified Model for Knowledge Embedding and Pre-trained Language Representation. *Trans. Assoc. Comput. Linguistics*. 2021a, 9, p. 176–194. doi: 10.1162/tacl_a_00360. Available at: https://doi.org/10.1162/tacl_a_00360.

Wang, Z. – Zhang, G. – Yang, K. – Shi, N. – Zhou, W. – Hao, S. – Xiong, G. – Li, Y. – Sim, M. Y. – Chen, X. – others. Interactive natural language processing. *arXiv preprint arXiv:2305.13246*. 2023d.

Wang, Z. – Dong, H. – Jia, R. – Li, J. – Fu, Z. – Han, S. – Zhang, D. Tuta: Tree-based transformers for generally structured table pre-training. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, p. 1780–1790, 2021b.

Wei, J. – Tay, Y. – Bommasani, R. – Raffel, C. – Zoph, B. – Borgeaud, S. – Yogatama, D. – Bosma, M. – Zhou, D. – Metzler, D. – others. Emergent Abilities of Large Language Models. *Transactions on Machine Learning Research*. 2022a.

WEI, J. – WANG, X. – SCHUURMANS, D. – BOSMA, M. – CHI, E. – LE, Q. – ZHOU, D. Chain of Thought Prompting Elicits Reasoning in Large Language Models, 2022b. Available at: http://arxiv.org/abs/2201.11903.

WEN, T.-H. – YOUNG, S. Recurrent neural network language generation for spoken dialogue systems. *Computer Speech & Language*. 2020, 63, p. 101017.

WEN, T.-H. – GAŠIC, M. – MRKŠIC, N. – ROJAS-BARAHONA, L. M. – SU, P.-H. – VANDYKE, D. – YOUNG, S. Toward multi-domain language generation using recurrent neural networks. In *NIPS Workshop on Machine Learning for Spoken Language Understanding and Interaction*, 2015a.

WEN, T.-H. – GASIC, M. – MRKŠIĆ, N. – SU, P.-H. – VANDYKE, D. – YOUNG, S. Semantically Conditioned LSTM-based Natural Language Generation for Spoken Dialogue Systems. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, p. 1711–1721, 2015b.

WEN, T. – GASIC, M. – MRKSIC, N. – ROJAS-BARAHONA, L. M. – SU, P. – VANDYKE, D. – YOUNG, S. J. Multi-domain Neural Network Language Generation for Spoken Dialogue Systems. In KNIGHT, K. – NENKOVA, A. – RAMBOW, O. (Ed.) *NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, p. 120–129, San Diego California, USA, 2016. doi: 10.18653/V1/N16-1015. Available at: https://doi.org/10.18653/v1/n16-1015.

WENZEK, G. – LACHAUX, M.-A. – CONNEAU, A. – CHAUDHARY, V. – GUZMÁN, F. – JOULIN, A. – GRAVE, É. CCNet: Extracting High Quality Monolingual Datasets from Web Crawl Data. In *Proceedings of The 12th Language Resources and Evaluation Conference (LREC)*, p. 4003–4012, Marseille, France, 2020. Available at: https://www.aclweb.org/anthology/2020.lrec-1.494/.

WHITE, M. – RAJKUMAR, R. – MARTIN, S. Towards broad coverage surface realization with CCG. In *Proceedings of the Workshop on Using corpora for natural language generation*, 2007.

WIHER, G. – MEISTER, C. – COTTERELL, R. On decoding strategies for neural text generators. *Transactions of the Association for Computational Linguistics*. 2022, 10, p. 997–1012.

WILCOXON, F. Individual comparisons by ranking methods. In *Breakthroughs in statistics*. : Springer, 1992. p. 196–202.

WILLIAMS, A. – NANGIA, N. – BOWMAN, S. A Broad-Coverage Challenge Corpus for Sentence Understanding through Inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, p. 1112–1122. Association for Computational Linguistics, 2018. Available at: http://aclweb.org/anthology/N18-1101.

Wiseman, S. – Shieber, S. M. – Rush, A. M. Challenges in Data-to-Document Generation. In Palmer, M. – Hwa, R. – Riedel, S. (Ed.) *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017*, p. 2253–2263, Copenhagen, Denmark, 2017. doi: 10.18653/V1/D17-1239. Available at: `https://doi.org/10.18653/v1/d17-1239`.

Wiseman, S. – Shieber, S. M. – Rush, A. M. Learning Neural Templates for Text Generation. In Riloff, E. – Chiang, D. – Hockenmaier, J. – Tsujii, J. (Ed.) *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, p. 3174–3187, Brussels, Belgium, 2018. doi: 10.18653/v1/d18-1356. Available at: `https://doi.org/10.18653/v1/d18-1356`.

Wolf, T. – Debut, L. – Sanh, V. – Chaumond, J. – Delangue, C. – Moi, A. – Cistac, P. – Rault, T. – Louf, R. – Funtowicz, M. – Brew, J. HuggingFace's Transformers: State-of-the-art Natural Language Processing. *ArXiv*. 2019, abs/1910.03771.

Woolley, G. H. Automatic Text Generation. In *International Conference on Computational Linguistics COLING 1969: Preprint No. 37*, Sånga Säby, Sweden, September 1969. Available at: `https://aclanthology.org/C69-3701`.

Xiang, J. – Liu, Z. – Zhou, Y. – Xing, E. P. – Hu, Z. ASDOT: Any-Shot Data-to-Text Generation with Pretrained Language Models, 2022. Available at: `http://arxiv.org/abs/2210.04325`.

Xie, T. et al. UnifiedSKG: Unifying and Multi-Tasking Structured Knowledge Grounding with Text-to-Text Language Models, 2022. Available at: `http://arxiv.org/abs/2201.05966`.

Xie, Z. – Cohn, T. – Lau, J. H. The Next Chapter: A Study of Large Language Models in Storytelling. In *Proceedings of the 16th International Natural Language Generation Conference*, p. 323–351, 2023.

Xu, X. – Dušek, O. – Rieser, V. – Konstas, I. AGGGEN: Ordering and Aggregating While Generating. *arXiv:2106.05580 [cs]*. 2021. Available at: `http://arxiv.org/abs/2106.05580`.

Xu, X. – Titov, I. – Lapata, M. Compositional Generalization for Data-to-Text Generation, 2023. Available at: `http://arxiv.org/abs/2312.02748`.

Xue, L. – Constant, N. – Roberts, A. – Kale, M. – Al-Rfou, R. – Siddhant, A. – Barua, A. – Raffel, C. mT5: A Massively Multilingual Pre-Trained Text-to-Text Transformer. *arXiv:2010.11934 [cs]*. 2021. Available at: `http://arxiv.org/abs/2010.11934`.

Yang, J. – Gupta, A. – Upadhyay, S. – He, L. – Goel, R. – Paul, S. TableFormer: Robust Transformer Modeling for Table-Text Encoding. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, p. 528–537, Dublin, Ireland, 2022. Association for Computational Linguistics. Available at: `https://aclanthology.org/2022.acl-long.40`.

Yang, Z. – Einolghozati, A. – Inan, H. – Diedrick, K. – Fan, A. – Donmez, P. – Gupta, S. 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+)*, p. 107–116, 2020.

Yngve, V. H. Random generation of English sentences. In *Proceedings of the International Conference on Machine Translation and Applied Language Analysis*, 1961.

Yuan, S. – Färber, M. Evaluating Generative Models for Graph-to-Text Generation, 2023. Available at: http://arxiv.org/abs/2307.14712.

Yuan, W. – Neubig, G. – Liu, P. BARTScore: Evaluating Generated Text as Text Generation. In *Advances in Neural Information Processing Systems*, 34, p. 27263–27277. Curran Associates, Inc., 2021. Available at: https://proceedings.neurips.cc/paper/2021/hash/e4d2b6e6fdeca3e60e0f1a62fee3d9dd-Abstract.html.

Zarrieß, S. – Voigt, H. – Schüz, S. Decoding methods in neural language generation: a survey. *Information*. 2021, 12, 9, p. 355.

Zhang, T. – Kishore, V. – Wu, F. – Weinberger, K. Q. – Artzi, Y. BERTScore: Evaluating Text Generation with BERT. In *International Conference on Learning Representations*, 2019.

Zhang, T. – Ladhak, F. – Durmus, E. – Liang, P. – McKeown, K. – Hashimoto, T. B. Benchmarking large language models for news summarization. *Transactions of the Association for Computational Linguistics*. 2024, 12, p. 39–57.

Zhang, Z. – Wu, Y. – Zhao, H. – Li, Z. – Zhang, S. – Zhou, X. – Zhou, X. Semantics-aware BERT for language understanding. In *Thirty-Fourth AAAI Conference on Artificial Intelligence (AAAI-2020)*, 2020. Available at: https://arxiv.org/abs/1909.02209.

Zhao, W. X. – Zhou, K. – Li, J. – Tang, T. – Wang, X. – Hou, Y. – Min, Y. – Zhang, B. – Zhang, J. – Dong, Z. – others. A survey of large language models. *arXiv preprint arXiv:2303.18223*. 2023a.

Zhao, W. – Peyrard, M. – Liu, F. – Gao, Y. – Meyer, C. M. – Eger, S. MoverScore: Text Generation Evaluating with Contextualized Embeddings and Earth Mover Distance. *arXiv:1909.02622 [cs]*. 2019. Available at: http://arxiv.org/abs/1909.02622.

Zhao, Y. – Zhang, H. – Si, S. – Nan, L. – Tang, X. – Cohan, A. Investigating Table-to-Text Generation Capabilities of LLMs in Real-World Information Seeking Scenarios, 2023b. Available at: http://arxiv.org/abs/2305.14987.

Zhong, V. – Xiong, C. – Socher, R. Seq2sql: Generating structured queries from natural language using reinforcement learning. *arXiv preprint arXiv:1709.00103*. 2017.

Zhou, H. – Bradley, A. – Littwin, E. – Razin, N. – Saremi, O. – Susskind, J. – Bengio, S. – Nakkiran, P. What Algorithms Can Transformers Learn? A Study in Length Generalization, 2023. Available at: `http://arxiv.org/abs/2310.16028`.

List of Abbreviations

This document is incomplete. The external file associated with the glossary ‘acronym’ (which should be called `thesis.acr`) hasn’t been created.

Check the contents of the file `thesis.acn`. If it’s empty, that means you haven’t indexed any of your entries in this glossary (using commands like `\gls` or `\glsadd`) so this list can’t be generated. If the file isn’t empty, the document build process hasn’t been completed.

Try one of the following:

- Add `automake` to your package option list when you load `glossaries-extra.sty`. For example:

  ```
  \usepackage[automake]{glossaries-extra}
  ```

- Run the external (Lua) application:

  ```
  makeglossaries-lite.lua "thesis"
  ```

- Run the external (Perl) application:

  ```
  makeglossaries "thesis"
  ```

Then rerun LaTeX on this document.

This message will be removed once the problem has been fixed.

List of Tables

List of Figures

List of Publications