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

I declare that I carried out this doctoral thesis independently, and only with the cited sources, literature and other professional sources.

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

Prague, June 16, 2024 Zdeněk Kasner

**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:**

Data-to-text generation systems need to produce texts with high levels of semantic accuracy. Rule-based systems can guarantee this aspect, but their fluency and adaptability to new domains remain limited. Meanwhile, neural language models can easily generate fluent texts and adapt to new domains but are notoriously prone to producing inaccurate outputs. This thesis explores how to efficiently employ neural components in data-to-text generation systems to get the best of both worlds. We focus on approaches based on pretrained transformer language models. Primarily, the models serve as building blocks for data-efficient and robust data-to-text generation systems. Along with that, we introduce model-based evaluation metrics, focusing on detecting errors in data-to-text outputs, and a toolkit for preprocessing and visualizing data-to-text generation datasets. We also analyze the behavior of pretrained and large language models in specific scenarios, including describing individual relations in knowledge graphs and generating texts from standard data formats. We conclude that while employing neural language models in data-to-text generation remains a delicate endeavor, neural components can improve the fluency of the output texts and make the systems adaptable to new domains. At the same time, the semantic accuracy of the outputs can remain high if the models are used for specific, well-defined subtasks for improving text quality. For future research, we emphasize the need for benchmarking with suitable evaluation metrics on real-world use cases.

**Keywords:** data-to-text generation, natural language generation, natural language processing, transformer architecture, pretrained language models, large language models

**Název práce:** Generování textu z dat s neuronovými jazykovými modely

**Autor:** Zdeněk Kasner

**Katedra:** Ústav formální a aplikované lingvistiky

**Vedoucí práce:** Mgr. et Mgr. Ondřej Dušek, Ph.D.,
Ústav formální a aplikované lingvistiky

**Abstrakt:**

Systémy pro generování textu z dat by měly generovat texty odpovídající co nejpřesněji vstupním datům. Pravidlové systémy tento aspekt zaručují, ale zaostávají v plynulosti výstupů a možnostech přizpůsobení pro nové domény. Naopak neuronové jazykové modely zvládají snadno generovat plynulé texty a přizpůsobovat se novým doménám, ale jsou notoricky náchylné k produkci nepřesných výstupů. V této práci zkoumáme, jak efektivně zakomponovat do systémů pro generování textu z dat neuronové modely tak, abychom propojili výhody obou typů systémů. Naše přístupy zakládáme na předtrénovaných jazykových modelech architektury transformer. Tyto modely primárně používáme jako stavební bloky, díky kterým mohou být systémy pro generování textu robustní a efektivně se učit z trénovacích dat. Spolu s tím představujeme automatické evaluační metriky pro odhalování chyb ve výstupech a sadu nástrojů pro předzpracování a vizualizaci datasetů pro generování textu z dat. Analyzujeme také chování předtrénovaných a velkých jazykových modelů ve specifických případech jako je popis jednotlivých relaci ve znalostních grafech a generování textů ze standardních datových formátů. Z našich experimentů vyplývá, že ačkoli k použití neuronových jazykových modelů při generování textu z dat je potřeba přistupovat s rozmyslem, neuronové komponenty mohou zlepšit plynulost výstupních textů a přizpůsobitelnost systémů novým doménám. Přesnost výstupů přitom může zůstat vysoká, pokud jsou modely používány pro konkrétní dílčí úkoly pro zlepšení kvality textu. Cílem budoucího výzkumu by mělo být vyhodnocování systémů pomocí vhodných evaluačních metrik na reálných problémech.

**Klíčová slova:** generování textu z dat, generování přirozeného jazyka, zpracování přirozeného jazyka, architektura transformer, předtrénované jazykové modely, velké jazykové modely

## Acknowledgements

What a ride this has been. When I signed up for the Natural Language Processing course during my Erasmus+ stay at KU Leuven back in 2017, little did I know it would be *the* flap of a butterfly wing. It feels tempting to say that I have been drawn towards language processing since then, propelled by my fondness for reading, writing, and learning languages. But that is just hindsight bias: I am quite sure that if it were not for a couple of other life episodes, I still might have ended up doing something completely different.

One such episode brought me to the Institute of Formal and Applied Linguistics—ÚFAL. There, I met Jindřich Helcl and Jindřich Libovický, my current colleagues and friends. Thanks to your course, I learned how research feels like, how strong is the link between AI and NLP, and most importantly, how friendly are the people at ÚFAL. I could not help it, interacting with you just felt right. After Jindřich H. became my Master's thesis supervisor, I started to consider doing research in NLP—but for that, I first needed a PhD supervisor.

I once read that the PhD supervisor/student relationship is akin to marriage, as it leads to intense personal interactions over several years. If that is the case, what are the chances that you find a perfect person for that during a random encounter in the hallway? However, that is essentially what happened to me. I sincerely could not have wished for a better advisor than Ondřej Dušek. Thank you for teaching me how to think and write properly, for all the discussions during our meetings about the quirks of doing research (which were often off-topic but always insightful and fun), for the intense exchanges before paper deadlines, that made me feel supported and motivated, and in general for always being there, from the very beginning up till the write-up of this thesis. Without you, I would not be where I am now.

Looking back at my five years at ÚFAL, my initial instincts were correct – you became not only my colleagues but my friends. As colleagues, you inspired me with your approach to research: working on what feels exciting and meaningful instead of chasing research throughput, and staying playful while producing top-notch research. At the same time, life at ÚFAL meant more to me than research. To put it briefly: right after our first autumn retreat—where we cycled eighty kilometers, climbed to the peak of Sněžka, and then played guitar the whole evening—I knew I was in the right place. Naming just a few out of many: thank you, Vojta Hudeček for becoming my friend and outdoor buddy, Rudolf Rosa for inviting me to your wedding, Tomáš Musil for our co-teaching and train rides, people from our NG-NLG team (Simone Balloccu, Mateusz Lango, Sourabrata Mukherjee, Ondřej Plátek, Patrícia Schmidtová) for our

lively discussions full of jokes, Tomasz Limisiewicz for our hallway encounters and conference travels, Tea Vojtěchová for taking part in my ventures to Irish dancing, and all of you that made our monthly "community activity outside of the workplace" a pleasant experience.

After two years of covid, my three months at Heriot-Watt University in Edinburgh felt almost miraculous. Thank you, Ioannis Konstas, for volunteering to be my mentor, and Xinnuo Xu, for being my local guide and friend. Yet another dream of mine became true a year later when I was accepted for a research internship at the Mila - AI Research Institute in Montréal. I am thankful to Siva Reddy and the rest of the team for cordially inviting me to the group and keeping in touch later on. Special thanks to Xing Han Lù for your enthusiasm, cool ideas, and being my friend in Montréal and abroad. Our pair programming was an enriching experience, especially with your thoughtful process of making coffee.

If I were not able to enjoy my life outside of research, none of this would matter. Luckily, I was blessed to have many friends that made these years a real joyride. Thanks to my close friends (Dan, Tomáš, Michal, Jan, and others) for our ongoing interactions, the X-Challenge community for teaching me—nothing less than—how to live, Vlakfest and Expedition Club for taking me to unorthodox adventures, and people from the Board of European Students of Technology for staying in my life way beyond our student years. And thanks to all the other people that I did not get a chance to mention but your name belongs here.

The list would not be complete without you, Lenka. You became such an essential element in my life that I now cannot imagine it any other way. Thank you for all the affection and support that I got from you while writing this thesis, the time spent together, our internal jokes, moments of understanding, and most of all, our common horizons I am looking forward to. You give sense to all of this.

All those years that got me here, I have been constantly supported by my family. Thanks to my parents, Zdeněk and Dana, for always welcoming me with open arms and caring about my life. Thanks to my brother Marek for our discussions, friendship, and many adventures. Lastly, thanks to my grandmother Zdeňka, who—besides raising me as a kid—supported my vague childhood idea of becoming a university teacher "since I will have three months of holidays". Learning about the reality of academia, I am sure she would be proud nevertheless.

If you are the village, I am the child. And out of all of you, there must have been somebody who made me enjoy writing so much that I barely know when to stop...

The work has been supported by the European Union (ERC, NG-NLG, 101039303) and by the Charles University projects GAUK 140320 and SVV 260 698. 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 understand 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 successfully 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., 2019; Brown et al., 2020). In the end, it took us over fifty years to build such systems. Similarly to other tasks in artificial intelligence, 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, redefining 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, 2008) and healthcare report generators (Portet et al., 2009)—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., 2023). 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 field of machine learning—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 a 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 texts 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 How 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 a 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 generalization 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 related to neural language models (LMs) and data-to-text generation. The chapter serves as the main point of reference for the concepts and related work referenced throughout the thesis; we will only briefly revisit the most relevant concepts in the respective chapters.

In Section 2.1, we first cover *neural 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.

## 2.1 Neural Language Models

In this section, we work our way towards neural LMs. We start with the mathematical foundations of neural networks on which 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] In our case, we are interested the most in learning language patterns from large-scale textual data, which will in turn help us with generating text.

To begin, let us say that our goal is to predict a real-number output $y \in \mathbb{R}$ for a given vector of real numbers $\mathbf{x} = (x_1, \ldots, x_d) \in \mathbb{R}^d$. We assume that the $\mathbf{x} \to y$ mapping is not arbitrary—if it were, it 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 topics.

In machine learning, we approximate the mapping using mathematical models designed to capture the patterns in their parameters. The models estimate the parameters from a set of $(\mathbf{x}, y)$ examples called the *training data* and use these parameters to predict the outputs on the *test data*, which is a new set of examples generally coming from the same distribution.

**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 $\mathbf{w} = (w_1, \ldots, w_d) \in \mathbb{R}^d$ 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 $y_i$:

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

$$\mathbf{w} = \mathbf{w} + (y - \hat{y})\mathbf{x} \qquad \triangleright \text{ Weights update} \tag{2.2}$$

$$b = b + y - \hat{y} \qquad \triangleright \text{ Bias update} \tag{2.3}$$

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

The perceptron algorithm is guaranteed to converge if (and only if) a hyperplane exists that separates the data belonging to one class from another (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, an 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 $f : \mathbb{R} \rightarrow \mathbb{R}$ 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 $\mathbf{b}_i \in \mathbb{R}^n$. The layer produces an 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. Similarly as with the perceptron, we first do a *feed-forward pass* for each training example $(\mathbf{x}, y)$: 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* $\mathcal{L}(y, \hat{y}) \rightarrow \mathbb{R}$. 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—with respect to the loss function 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. As the speed and robustness of convergence depend on the learning rate, more advanced optimizers—such as Adam (Kingma and Ba, 2015)—adapt the learning rate for each parameter based on the history of the gradients.

**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 process can be thought of as repeatedly applying the same feed-forward layer to each element of the input and updated hidden state from the previous step (hence the recurrence). 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), which we will not cover in detail here.

RNNs had a lot of success across NLP on sequence processing tasks (Karpathy, 2015; Salehinejad et al., 2018). At the same time, RNNs turned out to have various shortcomings, such as the vanishing gradient problem (which arises due to repeated gradient updates with limited numerical precision), the fixed size of the hidden state (which limits the amount of information about the sequence that can be stored during the computation), 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 they became a predecessor to the transformer architecture.

### 2.1.2 Text Representation

Until now, we have assumed that our inputs are numerical vectors. However, it is yet not clear how to represent *text* using these vectors – that is something we will look into in this section.

**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 the set of all possible tokens (a *vocabulary* $V$) and assign each token an integer index $i \in \{0, \ldots, |V| - 1\}$.

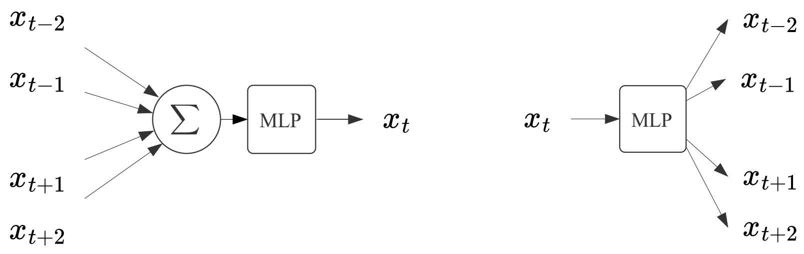
The naive way to represent each token would be using its integer value. However, this would misleadingly suggest linear dependence between tokens. A better way is to use the index $i$ for constructing a *one-hot* vector $\mathbf{x} \in \{0, 1\}^{|V|}$ for each token:

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

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 one where tokens with similar meanings have similar representations. To get this kind of 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—predicting 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* $\mathbf{x} \in \mathbb{R}^d$.




(a) Continuous Bag-of-Words

(b) Skip-gram

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) *continuous bag-of-words* algorithm, we sum the embeddings of $k - 1$ surrounding tokens and predict the original token. In the (b) *skip-gram* algorithm, we use the original token for predicting the $k - 1$ surrounding tokens.

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

**Tokenization** For processing text as a sequence, we need to have a way of tokenizing the text, i.e. splitting it into discrete units. The most straightforward way would be to tokenize the text into either words or characters. Unfortunately, both of these approaches have major shortcomings. With word-level tokenization, we are not able to represent *unknown* words, i.e., the words not seen in the training corpus. Word-level tokenization also considers morphologically similar words as independent units, forcing the model to learn their representation separately. Moreover, word-level tokenization becomes more difficult for languages such as Chinese, which does not separate words with spaces. In contrast, character-level tokenization uses a small and well-defined set of tokens, but the tokens are less meaningful and the resulting sequence is much longer, making the approach computationally inefficient. (Jurafsky and Martin, 2024, p.19)

*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. (Jurafsky and Martin, 2024, p.21)

A 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.

There are also alternative sub-word tokenization algorithms, differing in their approach to constructing the vocabulary. WordPiece (Wu et al., 2016) works similarly to BPE, but instead of the most frequent token it chooses the token that maximizes the likelihood of the training data. Unigram (Kudo, 2018) proceeds—unlike the previous algorithms—top-down, starting with a large vocabulary and progressively reducing the number of tokens to minimize unigram loss over the training data.

### 2.1.3 Language Modeling

After introducing the main principles 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 the 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):

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

$n$-gram LMs store the counts of *n*-gram occurrences over a training corpus in a look-up table. The probabilities are then estimated using these counts, interpolating over lower-order $n$-grams in case the specific $n$-gram did not occur in the training corpus. 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 (Bengio et al., 2003).

**Neural Language Model** A neural LM is a language model that estimates the text probability $P_\theta(X)$ using a neural network with parameters $\theta$. In contrast to *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: $P_\theta(x_i|x_{<i})$ given preceding words in the context $x_{<i}$. If we express the gap between the model distribution $P_\theta$ 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 naturally occurring in the corpus serves as the ground-truth label that the model aims to predict.

### 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), originally introduced on top of RNNs, 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]

[2]We will later adapt the idea also for the transformer architecture.

(1) The first 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 second 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, which is fed back as input in the next step. 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 used in every step has a fixed size, which limits the amount of information the network can capture about a sequence. The *attention mechanism* (Bahdanau et al., 2015; Luong et al., 2015) bypasses this bottleneck by enabling the decoder to extract information dynamically from the whole encoded sequence.

At each step $j$, the decoder first computes a weight vector $\boldsymbol{\alpha}_j$: a probability distribution over the encoder's hidden states. The coefficients $\alpha_{ji}$ are then used as weights in computing a context vector $\mathbf{c}_j$, which incorporates information from every hidden state of the encoder proportionally to its weight. The context vector is used as an additional input for the decoder:

$$\mathbf{h}_d^{(j)}, y_j = \mathcal{D}(\mathbf{h}_d^{(j-1)}, y_{j-1}, \mathbf{c}_j). \tag{2.13}$$

**Transformer Architecture** The *transformer*[3] (Vaswani et al., 2017) is a neural sequence processing architecture. Similarly as with RNNs, the input of the transformer is a sequence $\mathbf{X} \in \mathbb{R}^{n,d}$ and the output is the series of hidden states $\mathbf{H} \in \mathbb{R}^{n,d}$. Unlike RNNs, the transformer can process the sequence efficiently in parallel. To achieve that, the transformer replaces the RNN hidden state—used previously as a mechanism for sharing information among tokens within a sequence—with the *self-attention mechanism.*

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

Specifically, the transformer processes the input in a series of blocks. Each block is composed of two layers: (a) the *self-attention layer* and (b) the *MLP layer*. The layers serve a different purpose: while the MLP layer computes element-wise operations over each token, the self-attention layer enables sharing the information among tokens.

- **Self-attention layer**: Self-attention (Cheng et al., 2016; Vaswani et al., 2017) is a variant of the attention mechanism in which the source and the target states come from the same sequence. Given the input $\mathbf{X} \in \mathbb{R}^{n,d}$, the *self-attention* produces the output $\mathbf{A} \in \mathbb{R}^{n,d}$ of the same size. For the state $\mathbf{x}_j \in \mathbf{X}$, the self-attention mechanism computes the vector $\mathbf{a}_j \in \mathbf{A}$ as a weighted combination of the value vectors $\mathbf{v}_i$ corresponding to the states $\mathbf{x}_i \in \mathbf{X}$:

$$\mathbf{a}_j = \sum_{i \in 1..n} \alpha_{ji} \mathbf{v}_i, \tag{2.14}$$

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

$$\mathbf{v}_i = \mathbf{x}_i \mathbf{W_v}. \tag{2.15}$$

To get the attention weights $\alpha_{ji}$, we first compute *query* and *key* vectors for each state 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:

$$\mathbf{q}_i = \mathbf{x}_i \mathbf{W_q}, \tag{2.16}$$

$$\mathbf{k}_i = \mathbf{x}_i \mathbf{W_k}, \tag{2.17}$$

$$\alpha_{ji} = \mathrm{softmax}\left(\frac{\mathbf{q}_j \mathbf{k}_i}{\sqrt{d}}\right), \tag{2.18}$$

where $\mathrm{softmax}(\mathbf{x}) = \frac{\exp(\mathbf{x})}{\sum_i \exp(\mathbf{x}_i)}$. The operations can be efficiently parallelized using matrix multiplication:

$$\mathbf{Q} = \mathbf{X}\mathbf{W_q}, \quad \mathbf{K} = \mathbf{X}\mathbf{W_k}, \quad \mathbf{V} = \mathbf{X}\mathbf{W_v}, \tag{2.19}$$

$$\mathrm{attn}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \mathrm{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d}}\right)\mathbf{V}. \tag{2.20}$$

To capture different aspects of the input sequence, the transformer uses $k$ *attention heads*. Each head $\mathcal{H}_i$ is parametrized by a set of attention matrices $\mathbf{W}_\mathbf{q}^{(\mathcal{H}_i)}$,$\mathbf{W}_\mathbf{k}^{(\mathcal{H}_i)}$, and $\mathbf{W}_\mathbf{v}^{(\mathcal{H}_i)}$, computing the self-attention as described above. To compute the output of the self-attention layer, the output of each head is concatenated and linearly transformed using the trainable output matrix $\mathbf{W}_o$:

$$\mathbf{A} = \mathrm{concat}(\mathrm{attn}^{(\mathcal{H}_1)}, \ldots, \mathrm{attn}^{(\mathcal{H}_k)})\mathbf{W}_o. \tag{2.21}$$

- **MLP layer:** The MLP layer processes the outputs of the self-attention layer with a two-layer MLP. Specifically, it applies two linear transformations with a non-linear activation function $f$ in between:

$$\mathbf{H} = f(\mathbf{a}_j\mathbf{W}_1 + \mathbf{b}_1)\mathbf{W}_2 + \mathbf{b}_2, \tag{2.22}$$

  where $\mathbf{W}_1 \in \mathbb{R}^{d,d_{\mathrm{ff}}}$, $\mathbf{W}_2 \in \mathbb{R}^{d_{\mathrm{ff}},d}$, $\mathbf{b}_1 \in \mathbb{R}^{d_{\mathrm{ff}}}$, and $\mathbf{b}_2 \in \mathbb{R}^{d}$ are the trainable parameters and $d_{\mathrm{ff}}$ is the dimensionality of the hidden layer. Note that the operations in the MLP layer are element-wise (applied separately to each $\mathbf{a}_j$), so the transformation can be computed efficiently in parallel.

To stabilize the training process, the input of each layer (or output, depending on the architecture variant) is normalized using *layer normalization* (Ba et al., 2016). Yet another feature that helps to stabilize training is the *residual connection*: the fact that the output of the $i$-th layer is summed with its original input $\mathbf{H}^{(i)}$:

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

This way, the model can learn to adjust the input representation rather than completely replacing it.

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 would otherwise get lost in parallelized processing.[4]

As shown in Figure 2.2, which summarizes the architecture details discussed so far, the original transformer architecture is based on the encoder-decoder framework. The decoder blocks implement two additional features:

- Each block contains an additional layer called the *encoder-decoder attention*. In contrast to the self-attention mechanism, the *keys* and *values* come from the last block 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.

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

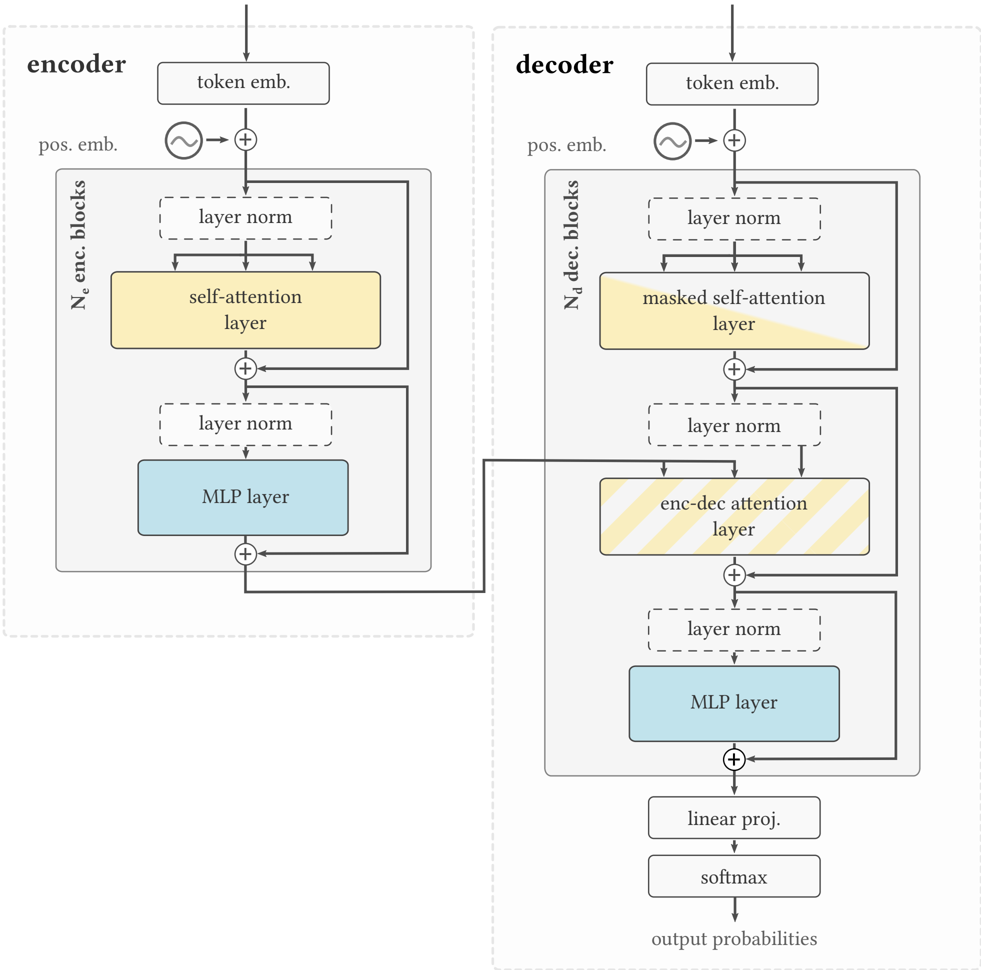



Figure 2.2: An encoder-decoder variant of the transformer architecture. The encoder has $N_e$ blocks, each consisting of a *self-attention* and *MLP layer*. The decoder has $N_d$ blocks with *masked self-attention* and *encoder-decoder attention*, again followed by an *MLP layer*. The input to each layer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

The hidden states produced by the transformer can be used for language modeling: after the last decoder block, 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.

**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 model-predicted probability distribution for the $i$-th position. The decoding stops once a special `</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 token selection step (line 4) can be realized in various ways, including:

- **Greedy decoding**: Selecting the most probable token: $y_i = \arg\max_{y \in V} p_\theta(y|y_{<i})$.
- **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.
- **Top-$k$ sampling**: Sampling the next token from the distribution of $k$ most probable tokens.
- **Top-$p$ (nucleus) sampling** (Holtzman et al., 2020): Sampling the next token from the distribution of tokens with cumulative probability $p$.

While greedy decoding and beam search are used to generate more probable sequences (approximating the exact algorithm for estimating the most probable sequence overall, which has exponential complexity), sampling algorithms are used to decode more creative outputs. Note that the list of the decoding algorithms as presented here is not exhaustive; see Zarrieß et al. (2021) and Meister et al. (2022) for an overview and further discussion.

### 2.1.5 Pretrained Language Models

To achieve good performance on an NLP task with a vanilla transformer model, we need an extensive amount of labeled training data. A more efficient workflow is as follows: the models are first *pretrained* on large-scale data—such as The Pile (Gao et al., 2021), or C4 (Raffel et al., 2020)—and then *finetuned* for downstream tasks on a smaller, task-specific dataset. Crucially, the pretraining is *self-supervised* (cf. Section 2.1.3), i.e., it can be done using general-domain data with no specific annotations. Although pretraining a model still requires significant computational resources, the checkpoints of pretrained language models (PLMs) can be used for efficient finetuning on downstream tasks.

| Type | Example Models | # Parameters | Note |
|---|---|---|---|
| Encoder | BERT (Devlin et al., 2019) | 110M-340M | notable 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., 2020) | 139M-406M | notable encoder-decoders |
| | T5 (Raffel et al., 2020) | 220M-11B | |
| | mBART (Liu et al., 2020) | 680M | multilingual version of BART |
| Decoder | GPT-2 (Radford et al., 2019) | 117M-1.5B | notable pretrained decoder |
| | Llama2 (Touvron et al., 2023) | 7B-70B | large language models (§2.1.6) |
| | Mistral (Jiang et al., 2023a) | | |
| | Zephyr (Tunstall et al., 2023) | | |

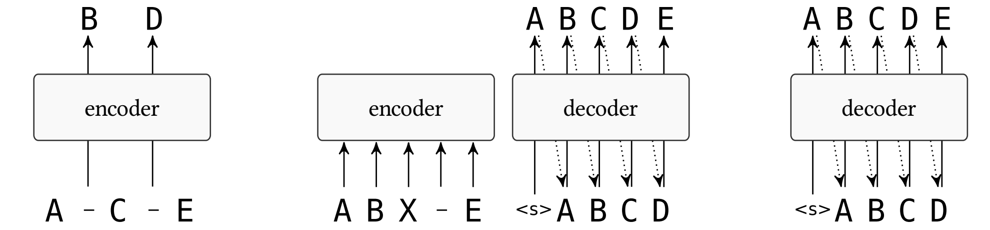
Table 2.1: Types of transformer architectures and specific models used in this work. The number of parameters may vary based on the model variant.

**Model types** Depending on the downstream task, different variants of the transformer architecture are used:

- **Encoder models** (Devlin et al., 2019; Liu et al., 2019b) 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** (Lewis et al., 2020; Raffel et al., 2020) 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** (Radford et al., 2018, 2019) 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 $\mathbf{Y}$.

Table 2.1 shows examples of the PLMs for each category, focusing on the models relevant to this work.

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




(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.

- **Masked Language Modeling**: The goal is to predict a token at a masked position given both its left and right context. This 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 encoder-only models such as BERT (Devlin et al., 2019).
- **Text Denoising**: The goal is generally to predict the original sequence from its corrupted version. This objective combines masked language modeling 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., 2020) or T5 (Raffel et al., 2020).
- **Causal language modeling**: The goal is to predict the next token given the previous sequence of tokens, as described in Equation 2.10. This objective is used for pretraining decoder-only models, including GPT-2 (Radford et al., 2019) 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 as given in Section 2.1.3 (Cotterell et al., 2023). However, all of the objectives are used in practice, often combined with 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 a pretrained model is more efficient than training a model from scratch, as the pretrained representations provide a warm start for the training process. However, finetuning typically cannot be applied repeatedly on the same model, as it leads to erasing previous knowledge, also known as *catastrophic forgetting* (McCloskey and Cohen, 1989; Kirkpatrick et al., 2016).

**Few-shot and Zero-shot Settings** If the size of the finetuning data is very limited (up to a few hundred examples), we talk about *few-shot* setting. By limiting the finetuning data to zero, we arrive at a *zero-shot* setting, where we use a model on a 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)

### 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 transformer decoders with billions of parameters and training tokens (Yang et al., 2024), although this definition is necessarily arbitrary to a degree (Rogers and Luccioni, 2024).

At the time of writing, LLMs are becoming an omnipresent phenomenon in most of the NLP areas. 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., 2023) and story generation (Xie et al., 2023), LLMs have comparable or better performance than previous task-specific approaches.

Although the most performant LLMs are currently available only through proprietary APIs (OpenAI, 2023a,b; Team et al., 2023; Anthropic, 2024), there is an increasing amount of performant open-access LLMs (Jiang et al., 2023a; Touvron et al., 2023) available through platforms such as HuggingFace Transformers (Wolf et al., 2019).

**In-context Learning** LLMs can perform certain tasks without the need for finetuning on task-specific data $E_{\text{task}} = \{(x_1, y_1), \ldots, (x_n, y_n)\}$. Instead of training, we provide $E_{\text{task}}$ as a part of the *prompt* (i.e., the text used as a decoding prefix). After $E_{\text{task}}$, we also append our test input $x_{n+1}$. Using causal language modeling with 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*.

**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., 2022; Ouyang et al., 2022). Due to their strong generalization abilities, the instruction-tuned models can be prompted to perform a task of choice in natural language, even without being directly trained for it. This allows to use the model for the task with no examples in the context, a setting known as *zero-shot prompting*.

## 2.2 Data-to-Text Generation

In this section, we provide background for the task of data-to-text generation. First, we present the task itself along with its 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), statistical (Section 2.2.4), and neural (Section 2.2.5) approaches. In the final part, we describe the D2T datasets (Section 2.2.6) and evaluation metrics (Section 2.2.7) we use in the thesis.

### 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 about 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, 2008; 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,[5] InfoSentience,[6] or vPhrase;[7] see also Dale (2023) for a recent overview).
- **Chart Captioning**: Generating captions[8] 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., 2009; 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).
- **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[9] which were addressed separately (Reiter and Dale, 1997; Reiter, 1996, 2007; Gatt and Krahmer, 2018). Even though 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 choose to split the pipeline into five representative subtasks, as illustrated in Figure 2.4:

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

[5] `https://www.arria.com`
[6] `https://infosentience.com`
[7] `https://www.vphrase.com`
[8] In contrast to image captioning (Stefanini et al., 2023), 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).
[9] The count is only approximate: for example, Mille et al. (2023) further subdivides some of the subtasks, leading to 10 subtasks in total.

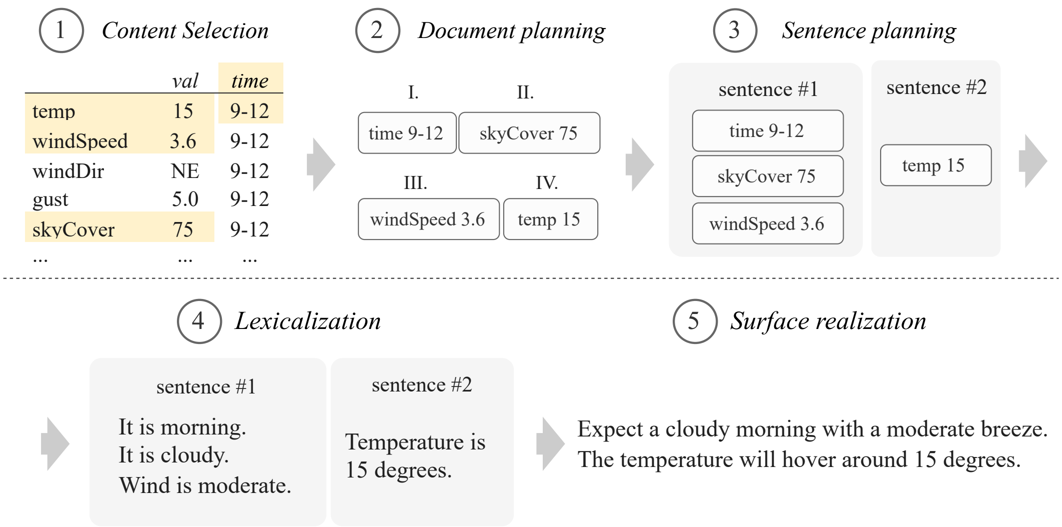



Figure 2.4: A five-step D2T generation pipeline presented on the example of generating a weather forecast. (1) The fields relevant for the forecast are selected from the data table. (2) The fields are ordered. (3) The ordered fields are aggregated into sentences. (4) Each field is transformed into a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

(2) **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 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 propagate to downstream modules. Despite this issue, the pipeline approach is the basis of many 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).

### 2.2.3 Rule-based Approaches

By *rule-based approaches* for D2T generation, we mean the approaches using manually defined rules or grammars.[10] 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., 2009). 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 that are designed to satisfy the desired communicative goals (such as *define*, *compare*, or *describe*; McKeown, 1985). The resulting rules can be formulated, for example, such as *"if a player scores two consecutive goals, describe these in the same sentence"* (Gatt and Krahmer, 2018).

**Template-based Lexicalization and Surface Realization** Simpler rule-based approaches for lexicalization and surface realization are typically based on *templates*: pre-written text snippets with placeholders that 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; see Dušek and Jurčíček (2019)). The resulting rule-based system is usually tied to a specific task and domain, but it can be a way to generate outputs of sufficient quality with reasonable development time and costs (van der Lee et al., 2018).

**Grammar-based Lexicalization and Surface Realization** A different way to handle lexicalization and surface realization in rule-based systems is using 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;

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

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 multiple valid outputs (Gatt and Krahmer, 2018).

### 2.2.4 Statistical Approaches

The idea of statistical[11] D2T generation approaches is to derive the inner workings of a D2T generation system (or its component) from statistics of a text corpus. This may apply both to individual steps of the D2T generation pipeline (estimating parameters of a specific module) or for parametrizing an end-to-end system (Liang et al., 2009; Dušek and Jurčíček, 2015). 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, 2008).

Even fully data-driven approaches still relied on grammatical rules; the only difference was that these 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, 2004) derived from the Penn Treebank (Hockenmaier and Steedman, 2007). Hybrid approaches combined a set of hand-written rules or grammars with statistical models (Konstas and Lapata, 2012; Gardent and Perez-Beltrachini, 2017).

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.

[11]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.

### 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 (Wen et al., 2015a; Dušek and Jurčíček, 2016). Thanks to advances in hardware (Hooker, 2021) and efficient 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. (2024); here, we mainly focus on the concepts and model architectures related to this thesis.

**Linearization** 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).

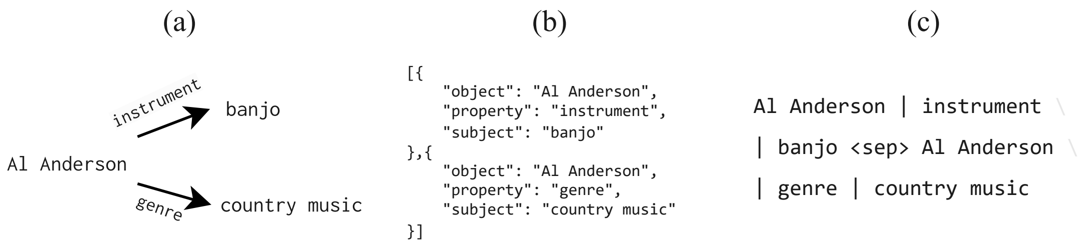



Figure 2.5: Representations of a simple knowledge graph: (a) the original knowledge graph, (b) JSON representation, (c) linearized representation.

**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, akin to template-based systems. 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) and Dušek and Jurčíček (2016) adopted RNNs for generating the response in a dialogue system, using a structured representation of the dialogue act as the input. Mei et al. (2016) use RNNs to address also the content selection step, identifying salient data records using the attention mechanism for generating weather reports.

An important addition to 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, enabling the model to fill in lexical values by itself. 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 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. (2022) propose an architecture based on three RNNs focusing separately on content, faithfulness, and fluency. 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 make 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).

PLMs made it possible to remove both *delexicalization* and *copy mechanism*. The general language modeling pretraining, along with the learned ability to copy tokens from the input, allows 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, variants of PLMs pretrained on multilingual corpora (Liu et al., 2020; Xue et al., 2021) can produce outputs in a variety of languages.

Due to the advantages above, PLM-based approaches excel in low-resource settings, which are common for many D2T generation tasks. Following Chen et al. (2020d), 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 related domains for zero-shot generation (see Section 3.3). 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 LLMs for D2T generation are still in naissance. Works which compared zero-shot or few-shot LLM prompting with finetuned PLMs on existing datasets have found that 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 zero-shot generation of data in standard data formats, with the main issue remaining the semantical accuracy of the outputs. However, to the best of our knowledge, there are yet no large-scale comparisons or attempts of finetuning LLMs for D2T generation (as of May 2024).

### 2.2.6 Datasets

In this section, we outline the format and structure of D2T generation datasets, focusing on the 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).[12]

[12]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.

| Dataset | Data Format | Domain(s) | # Total 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., 2022) | 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 |

Table 2.2: The list of D2T datasets used in this work. All listed datasets are included in the TabGenie framework (Section 5.1), except for Rotowire, where we include an updated version dubbed SportSett:Basketball instead (Thomson et al., 2020). Our main focus is on the datasets in **boldface**. Glossary of data types: *Key-value*: key-value pairs, *Table*: tabular data (*w/hl*: with highlighted cells), *SQL*: strings with SQL queries. ♦ indicates that the dataset is multi-domain; ♢ indicates that the dataset is open-domain. For brevity, we report only the total number of examples in the datasets (i.e., aggregating *train*, *dev*, and *test* sets).

.

**Data Formats** The following formats of structured data are present in the datasets that we use in this thesis (and at the same time, representative of D2T generation datasets in general):

- **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 generic value such as a text string, a number, or a boolean. The format is used, e.g., as a *meaning representation* for representing dialogue states in dialogue systems (Rastogi et al., 2020; Budzianowski et al., 2018).
- **RDF (Resource Description Framework)**[13] **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

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

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 (a text string or a number), and the predicate expresses the relation between the subject and the object.[14]

- **Tabular**: 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 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.

As we show in Section 5.1, key-value pairs and RDF triples can be converted to a tabular format with minimal information loss. We also show how to handle data in JSON format[15] in Section 6.2.

**Domains** In D2T generation, the notion of a *domain*—commonly used for drawing boundaries between the datasets or their subsets—mostly follows the dictionary definition of *an area of interest*.[16] 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. (2024) 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 other subset (van der Lee et al., 2020; Budzianowski et al., 2018; 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).

**Datasets** The following D2T generation datasets (highlighted in Table 2.2) 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 (such as Astronaut, Building, or Food), out of which 5 are *unseen*, i.e., included

[14] For this reason, a predicate may be also referred to as "relation".
[15] JavaScript Object Notation; `https://www.json.org`.
[16] `https://dictionary.cambridge.org/dictionary/english/domain`

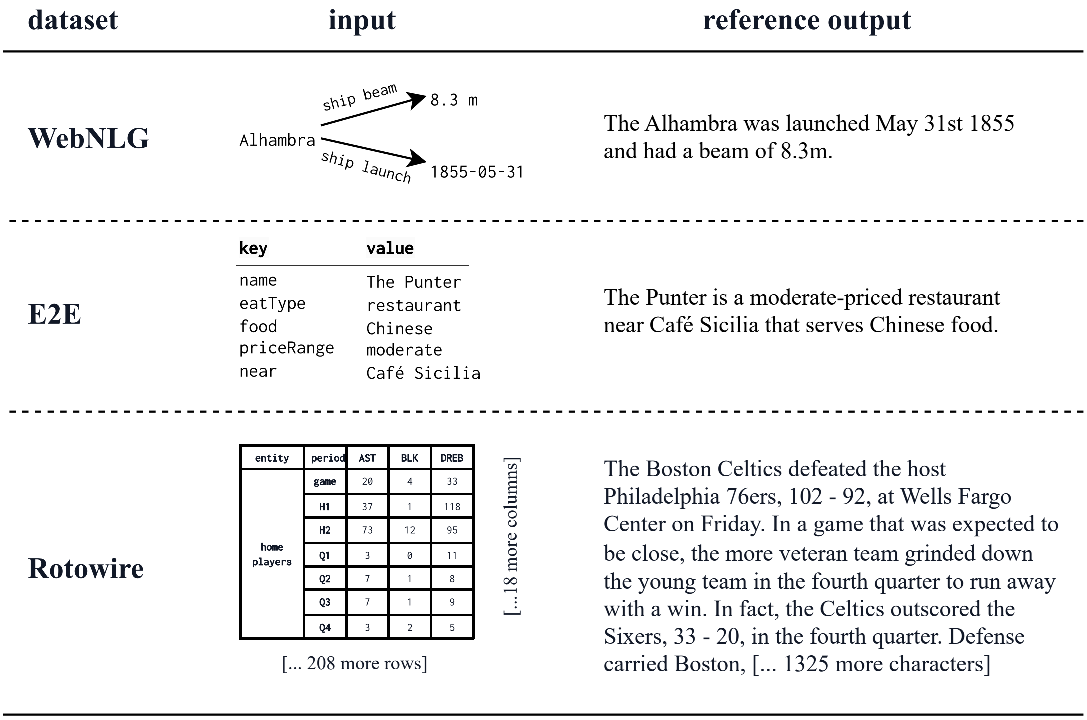



Figure 2.6: Example inputs and reference outputs from the WebNLG, E2E, and Rotowire datasets.

only in the test set. Each set of triples includes several verbalizations to promote lexical variability. In version 2, the dataset was annotated for intermediate subtasks and enriched with semi-automated German translations (Shimorina and Gardent, 2018; Ferreira et al., 2018). Version 3 of the dataset (Ferreira et al., 2020) contains one additional domain and automatic translations to Russian.

  – 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 D2T generation systems (Dušek et al., 2020). Since the original version of the dataset contained a lot of 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 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, so the systems also need to model the content selection step. Together with the 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).

See Figure 2.6 for example inputs and reference outputs from these datasets.

### 2.2.7 Evaluation Metrics

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., 2020).[17] 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, which should be correlated with human judgment (van der Lee et al., 2019). 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.

[17] As opposed to *extrinsic* measures, which evaluate the impact of the system in the external environment (Celikyilmaz et al., 2020). 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), which is why we focus on intrinsic measures in this work.

If we have human-written (also called *ground truth* or *gold-standard*[18]) 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 its fluency, diversity, and reading level, or—taking the input data into account—the faithfulness of the text with respect to the input data. (Celikyilmaz et al., 2020)

In the following paragraphs, we will introduce reference-based automatic metrics for measuring lexical similarity, semantic similarity, and semantic accuracy of the generated text, followed by referenceless automatic metrics for text fluency and lexical diversity. Finally, we will discuss evaluation methods based on human annotators and large language models.

**Lexical Similarity** Lexical similarity metrics measure the similarity between the generated and reference text using word-level (or character-level) overlap. 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., 2020). 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., 2023). 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 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 precisions, with a brevity penalty to penalize outputs shorter than the reference. BLEU was originally used for evaluating MT, but it has become commonplace in NLP. The SacreBLEU library (Post, 2018) was developed to tackle inconsistencies in implementations of the metric (Reiter, 2018).
- **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

[18]The term *gold-standard* 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.

(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.

- **METEOR** (Banerjee and Lavie, 2005) is a metric that computes the harmonic mean of precision and recall w.r.t. a reference computed on unigrams. METEOR also partially addresses non-exact matches 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 higher weights to less common n-grams, which are considered more informative (Doddington, 2002). Its length penalty is also more robust to slight variations in text length.
- **ChrF++** (Popovic, 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, ChrF++ 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 close to each other in the vector space. Semantic similarity metrics use this fact to measure the similarity of texts as a distance between their embeddings. The metrics most often rely on *contextual embeddings* computed by pretrained transformer encoders (Peters et al., 2018; Devlin et al., 2019; see Section 2.1.4). In contrast to lexical similarity metrics, semantic similarity metrics are more robust to 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., 2020a) measures the semantic similarity of 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 comparison across different works.

- **BLEURT** (Sellam et al., 2020) measures the semantic similarity of texts using a BERT model (Devlin et al., 2019) which is further finetuned for predicting human ratings on synthetically labeled data. Compared to BERTScore, BLEURT is less flexible but ensures a more consistent setup across works.
- **NUBIA** (Kané et al., 2020) measures the semantic similarity 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 perplexity from the GPT-2 model (Radford et al., 2019). 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[19] measures inaccuracies in the output text with respect to the input data. The inaccuracies in D2T generation 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, 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)

Honovich et al. (2022) presents a survey of factual consistency metrics, focusing on NLG areas such as summarization, fact verification, paraphrasing, and knowledge-grounded dialogue. Targetting specifically D2T generation, Data-QuestEval (Rebuffel et al., 2021) is a referenceless metric that uses QuestEval (Scialom et al., 2021), a tandem of question generation and question answering models. 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.

**Text Fluency** Text fluency is a catch-all term for measuring grammatical correctness, spelling, word, and stylistic choices of text (Celikyilmaz et al., 2020). In MT, lexical similarity metrics (such as BLEU) were used as a proxy for measuring text fluency, following the intuition that texts that are more similar to human written text

[19]The similar phenomenon is also called *faithfulness*, *factual accuracy*, or *factual consistency* (Celikyilmaz et al., 2020). In this work, we use the term *semantic accuracy* to refer to the faithfulness of the text *to the input data*, i.e., regardless of the factual correctness of the data itself (as opposed to *factual accuracy*, which is determined by the actual state of the world).

tend to be more fluent (Papineni et al., 2002; Celikyilmaz et al., 2020). However, outside of MT, the correlation between lexical similarity metrics and fluency was repeatedly found to be low or negative (Novikova et al., 2017; Fabbri et al., 2021; Nekvinda and Dušek, 2021). An alternative measure of text fluency is the *perplexity* of the text under a neural LM. This approach assumes that the LM assigns higher probability to more fluent sentences, which were supposedly more common in the pretraining corpus. Despite its shortcomings (Wang et al., 2022), this evaluation approach is used across various NLG works (Kann et al., 2018; Wang et al., 2020; Kané et al., 2020; Liu et al., 2021; Lee et al., 2022).

**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 the entropy of $n$-grams (Shannon, 1948). Lexical diversity is not generally required 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., 2023). 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; Howcroft 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[20] or Prolific[21] 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 and skill issues, 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.

[20] `https://www.mturk.com`
[21] `https://prolific.com`

- **Manual Error Analysis**: 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 LLMs-based metrics (Zhao et al., 2023c; 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 in following fine-grained instructions compared to other LLMs and of having high correlations with human judgment on evaluating generated texts. Since the model 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. We present experiments using a LLM-based metric for text evaluation in Section 6.2.

# 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 in-domain 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 (see Section 2.2.6 for the description of the datasets). These datasets assume that content selection was performed beforehand, i.e., we always want to verbalize the whole input (focusing on later steps of the D2T generation pipeline described in Section 2.2.2).

The most straightforward setting, that we present 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 simple setup is powerful and achieves competitive results on a shared task for generating knowledge graph descriptions, addressing RQ1. On top of that, we show that this approach can be directly applied to non-English settings, namely to text generation in Russian.

In Sections 3.2 and 3.3, we present approaches based on PLMs that can generate texts with an even more limited amount of in-domain training examples. While these approaches address the issue of efficiency (RQ2), using PLMs directly leads to another issue – how to control the model output (RQ3). 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 semantically correct) verbalization of the content to different, more controllable means, such as simple templates. Section 3.2 shows an approach based on a text-editing language model (a model that generates text by editing the input sequence) which has a limited vocabulary and is trained on iteratively fusing simple templates. The limited vocabulary and a specific training objective help the model generate semantically correct sentences. In Section 3.3, we present an alternative approach that uses an ordinary autoregressive pretrained model but 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 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 (Ferreira et al., 2020), 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. In Section 3.1.4, we show that with a moderate amount of in-domain finetuning data, a simple PLM-based approach can achieve satisfactory results in generating descriptions of knowledge graphs. We also point out its limitations in Section 3.1.5, namely its inability to infer the 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 (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 $X$.

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 (see Table 3.1). 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[1] 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.[2] 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 (∼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,[3] 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: $\text{mBART}_{\text{en}}$ on English inputs and English outputs, and $\text{mBART}_{\text{ru}}$ on English inputs and Russian outputs.

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

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

[3] 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, 2015) 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 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.

### 3.1.4 Results

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

**Automatic Metrics** The results of our approach for English are shown in Table 3.2. Our approach beats the baseline model based on the FORGe generator (Mille et al., 2019) 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 two metrics out of four (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 (worst) to 100 (best). The scores were clustered into groups 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 (out of two to four clusters) 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 (out of two to three clusters) 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 $\text{mBART}_{\text{en}}$ (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 $\text{mBART}_{\text{ru}}$, 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.[4] We found semantic 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). Semantic 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").

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

### 3.1.5 Discussion

**Why Our Approach Works** Our solution benefits from the mBART model, that 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.[5]

**Limitations** Building a high-quality training set of in-domain data, such 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., 2023b).

**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). In describing the 2023 results, 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 for high-resource languages), yielding

[5]In our case, we achieved satisfactory results using the default parameters, as described in Section 3.1.3.

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 (Sections 3.2.2 and 3.2.3). 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 (Sections 3.2.4 and 3.2.5). 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 experiments is available on Github.[6]

[6] 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., 2019), 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 $Y \operatorname{lex}(x_{k+1})$ as a source and the sequence $Y'$ as a target, where $\operatorname{lex}(x_{k+1})$ 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.

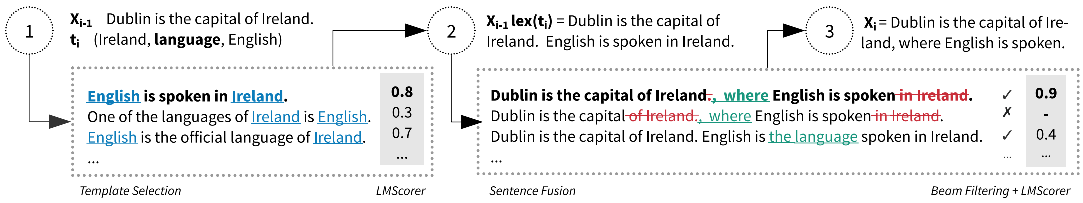



Figure 3.1: A single iteration of our algorithm for iterative D2T generation. In Step 1, the template for the triple is selected and filled. In Step 2, the sentence is fused with the template. In Step 3, the result for the next iteration is selected from the beam by filtering and language model scoring.

**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 $(y_1, \ldots, y_n)$ as a geometric mean of the token conditional probability:

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

**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 $i = (1, \ldots, T-1)$, 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 re-score 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 (E2E templates contain two objects). 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.[7] 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), which is a PLM based on BERT (Devlin 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 output 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., 2019) from the Huggingface Transformers (Wolf et al., 2019). We compute the perplexity scores using the *lm-scorer*[8] package.

[7]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.

[8]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 |
| **sent. fusion** | 0.353 | 7.923 | 0.386 | 0.555 | 0.252 | 4.460 | 0.338 | 0.436 |
| **zero-shot** | 0.288 | 6.677 | 0.385 | 0.530 | 0.220 | 3.941 | 0.340 | 0.408 |
| **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

**Baseline** For the *baseline*, we concatenate the best templates according to LMScorer without applying the sentence fusion (i.e., always using the fallback).

**Sentence Fusion** For the *sentence fusion* experiments, we use LaserTagger with the autoregressive decoder with a beam of size 10. We use all reference lexicalizations from WebNLG and E2E datasets 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 $2 \times 10^{-5}$. 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). We keep only the discourse types relevant to our use case,[9] which leaves us with >2M examples, i.e., two orders of magnitude more than in our D2T generation datasets, but lower variability of discourse connectives.

### 3.2.5 Results

**Accuracy vs. Fluency** On lexical similarity metrics (Table 3.5), 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 and Rastogi, 2020b). However, both the fusion and the zero-shot approaches show improvements

[9]The types we keep are PAIR_ANAPHORA, PAIR_NONE, SINGLE_APPOSITION, SINGLE_RELATIVE, SINGLE_S_COORD*, SINGLE_S_COORD_ANAPHORA*, SINGLE_VP_COORD*. For the discourse types with an asterisk, we only keep the examples with the connectives *"and"* or *", and"*.

| | |
|---|---|
| **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.

over the baseline. It is also important to note that our approach ensures zero entity errors by definition: we fill the entities verbatim into the templates, and if an entity is missing in the whole beam, we use a fallback instead (although semantic inconsistencies can still occur, e.g., if a verb or function words are missing).

**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. We note that the literal insertion of entities into the templates 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 conjunctions, 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 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.[10] 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 later opt for using a more generally applicable term *paragraph compression* (cf. Section 3.3).

**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 still attractive. 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., 2022; Chen et al., 2023a) 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).

[10]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 using autoregressive PLMs and adding modules for ordering and aggregation, turning the generation process into a three-step pipeline (Sections 3.3.2 and 3.3.4). We also propose a way to make each of these steps trainable on a generic synthetic corpus (Section 3.3.3). 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 (Sections 3.3.5 and 3.3.6). Our code and data is available on Github.[11]

### 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 that 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 is inspired by 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).

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

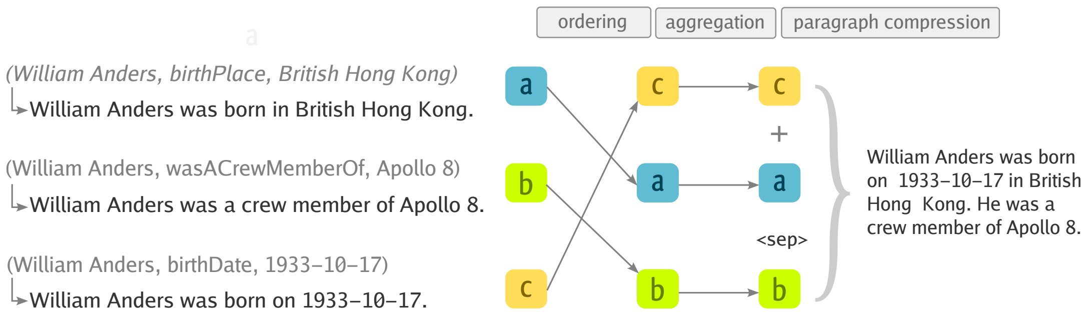



Figure 3.2: A scheme of our approach for zero-shot data-to-text generation from RDF triples. After a simple transformation of triples to facts, we apply the pipeline of modules for (1) ordering, (2) aggregation, and (3) paragraph compression. Individual modules are trained on a large general-domain text 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 remove 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., 2020d; 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 $T : X \rightarrow F$. 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 $F$ from the previous step 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 ordering adjacently 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 facts.

**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 sentences are not actually yet 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.

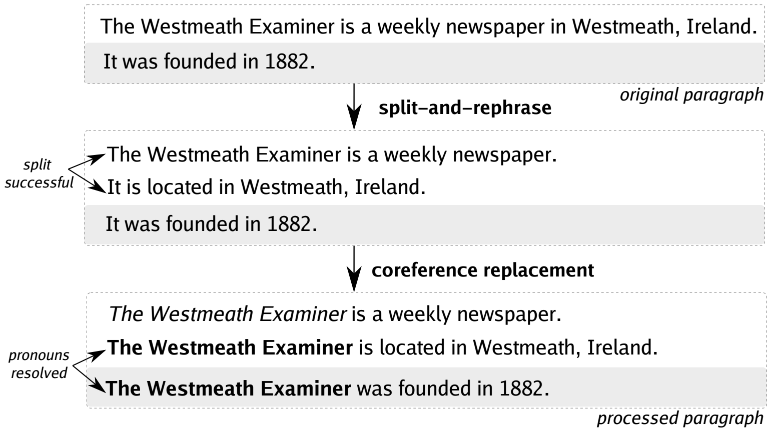



Figure 3.3: The building process of the WikiFluent corpus. We apply a split-and-rephrase model on each sentence in the paragraph and resolve coreferences in the split sentences. The result is a set of simple sentences that convey the same meaning as the original paragraph. The synthesized sentences are 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 where (1) the input is a set of simple, template-like sentences, and (2) the output is a fluent text in natural language preserving the semantics of the input. Here, we propose a way to build such a large-scale synthetic corpus from English Wikipedia. Our resulting corpus (WikiFluent) is orders of magnitude larger than in-domain D2T datasets (see Table 3.7) and provides training data for all the modules in our pipeline.

**Data Source** For building the WikiFluent corpus (Figure 3.3), we first extracted 934k first paragraphs of articles from a Wikipedia dump[12] 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.

[12] `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), the average number of tokens per source and target, the average number of sentences per source and target (after filling the templates for the D2T datasets), the 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[13] BART-base (Lewis et al., 2020) 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 et al., 2009) and apply the split-and-rephrase model to each sentence. To ensure that the splits are not deterministic, we choose uniformly randomly 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 any splitting.

**Coreference Replacement** The split sentences heavily use referring expressions, while the facts are presumably self-contained. Therefore, we apply a coreference resolution model (Lee et al., 2018) from the AllenNLP framework (Gardner et al., 2018) and 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[14] (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) 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).

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

### 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, i.e., using the templates extracted from the training data. Compared to more complex rule-based template generation engines (Laha et al., 2019; Heidari et al., 2021; Mehta et al., 2022), 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).[15] The model is based on a pretrained BART-base model (Lewis et al., 2020) 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 $\delta_i$. We ignore the outputs for the non-separator tokens while computing cross-entropy loss. We create training examples for aggregation 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.

**Paragraph Compression Model** For the paragraph compression model, we finetune BART-base (Lewis et al., 2020) on the WikiFluent corpus, concatenating the split sentences on the input and training the model to produce the original, human-written complex sentences on the output. 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.

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

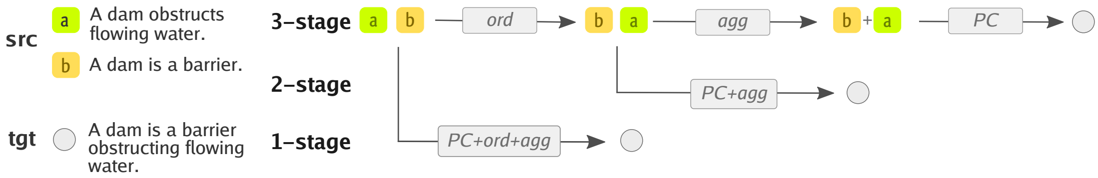



Figure 3.4: An example illustrating how the individual modules are trained and subsequently applied as the parts of the pipeline. See Section 3.3.5 for the description of the ordering model (ORD), the aggregation model (AGG), and the variants of the paragraph compression model (PC, PC+AGG, PC+ORD+AGG).

### 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,
- 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 ordering and aggregation 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.

| | | 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 3.8: 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 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.

**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.[16] We also evaluate the number of omission and hallucination errors (i.e., facts missing or added, respectively) using our automatic metric based on the RoBERTa model (Liu et al., 2019b) described in 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),
- 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 Jurčíček, 2015) – the baseline system for the E2E Challenge (Dušek et al., 2020),

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

|  |  | 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 3.9: Number of manually annotated errors on 100 examples: H = hallucinations, I = incorrect fact merging, O = omissions, R = redundancies, G = grammar errors or disfluencies.

- Harkous et al. (2020) – a state-of-the-art supervised system on the cleaned version of E2E data.

The automatic evaluation (Table 3.8) shows that our systems consistently outperform the COPY baseline (e.g., $\sim$12 BLEU points for E2E), which is already strong thanks to our manually curated set of templates.[17] Automatic scores also suggest that our systems are comparable with some older supervised systems, although they underperform the state-of-the-art supervised systems.

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 semantic errors (hallucinations, omissions, incorrect fact merging, redundancies) and grammatical errors. The results are summarized in Table 3.9.

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

[17] 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.

| | |
|---|---|
| **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 love kids. |

Table 3.10: 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. Note that in contrast to the model output, the reference contains a typo ("A amazing") and its style is generally less constrained.

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"). See Table 3.10 for an example of model output.

**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 this hypothesis. As shown in Su et al. (2021b), 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., 2022) 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 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 an LLM, as shown in Xiang et al. (2022) and Saha et al. (2023). In this case, however, special care needs to be taken to ensure a consistent style and semantic accuracy of model outputs (cf. Section 6.2).

## 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 light of recent progress, such as better controllability and output consistency.

# 4 Evaluating Semantic Accuracy

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. However, automatically evaluating this aspect using automatic metrics is not trivial, which is an issue that we raised in RQ4. To address this issue, this chapter introduces two metrics for evaluating the semantic accuracy of data-to-text (D2T) generation. The metrics are predominantly 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), i.e., classifying text entailment, which we repurpose for detecting omissions and hallucinations in the output text (see Section 2.2.7). Similarly to Chapter 3, we address the input format representation (RQ2) using simple templates for each data item. 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 semantic errors in D2T generation from complex tabular data. We propose a metric based on a PLM-based tagger, that can mark individual tokens with fine-grained error categories such as incorrect entity, incorrect number, or non-checkable fact. To provide relevant information from the structured data for the tagger, we combine the neural tagger with a rule-based fact generator and a neural-based retriever. The metric ranked first out of four automatic metrics in the Shared Task on Evaluating Accuracy in Generated Texts (Thomson and Reiter, 2021).

## 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 (Sections 4.1.2 and 4.1.3). We demonstrate that even without any extra model training and with minimal handcrafting, our approach achieves high accuracy (>90%) on the E2E dataset and produces useful results (>75% accuracy) on the more challenging WebNLG dataset. Additionally, we show with manual error analysis that some instances marked as faults of our metric were in fact assessed correctly by our metric (Section 4.1.4). The experimental code for our metric is available on GitHub.[1]

### 4.1.1 Motivation

In Section 2.2.7, we described two ways in which the semantic accuracy of D2T outputs can be violated: 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 fact is entailed by the generated text. Specifically, if we find that the 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


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* in the generated text. 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 a text entails a fact. 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 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., 2020b; 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 the 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 an omission, i.e., that the first fact is not entailed by the text (there is no mention of Blue Spice being a pub), and also a hallucination, i.e., that the text is not entailed by the facts (the information about kids is superfluous).

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, similarly as in Chapter 3 (see Section 2.2.6 for the descriptions of the datasets). Since 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 the answers for the 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 of 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 experiment with the *Default* and *Backoff* approaches to transforming triples to facts, 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. For E2E, we handcrafted eight templates for each predicate in the dataset.[5]

---

[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 in WebNLG, we choose randomly if more templates are found and use the backoff if no templates are found. Note that for E2E, we did *not* use the complex templates extracted from the training data (cf. Chapter 3).

| | WebNLG | | | | | E2E | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | **A** | **R** | **P** | **F1** | $\rho$ | **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, $\rho$ = Spearman correlation of confidence scores with human scores).

### 4.1.4 Evaluation

We evaluate our metric in terms of accuracy (**A**), precision (**P**), recall (**R**), and F1-measure[6] (**F1**) with respect to the corresponding ground truth outputs. For WebNLG, we additionally compute Spearman correlation coefficient ($\rho$) of the model's confidence scores with the average human scores. For E2E, we evaluate the accuracy for both the Fine (**Af**) and Rough (**Ar**) variants described in Section 4.1.2, making use of the fact that the automatic script reports both omissions and hallucinations. The 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 overall scores (between 77-80% for all measures) show that our metric deviates quite a lot from human judgments. 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.

[6]We treat *not_OK* samples as positive since we focus on detecting errors.

The Spearman correlation of our model's confidence scores with the average human scores is around 0.63 ($p$ <1e-10). This is similar to n-gram-based metrics on this data (Shimorina (2018) reports 0.59 for BLEU and 0.73 for METEOR), but unlike these metrics, our approach does not require human-written reference texts.

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** Automatic metrics for assessing semantic accuracy of text are mostly reference-based (Zhao et al., 2019; Dhingra et al., 2019; Sellam et al., 2020; Rony et al., 2022) or targetting tasks with non-structured inputs such as summarization, paraphrasing, or fact verification (Honovich et al., 2022; Zha et al., 2023), which makes their use-cases different from ours. The closest alternative to our metric is Data-QuestEval (Rebuffel et al., 2021): a metric based on a question generation and question answering model, which is trained on a synthetic dataset containing structured data on the input. In the future, a more flexible metric for evaluating semantic accuracy could be based on large language models (Zhao et al., 2023c; 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, we would like to refer to 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, Ondřej Dušek supervised the project.

In this section, we present an automatic metric for fine-grained detection of semantic accuracy errors in D2T generation outputs. In contrast with the example-level metric introduced in Section 4.1, the metric we introduce here can detect the hallucination errors on the level of individual *tokens*.[7] The metric combines a rule-based D2T generation system and PLMs (Section 4.2.3). We first use a rule-based D2T generation system to generate all facts that can be derived from the input as short natural-language sentences (cf. Section 4.1). 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.

[7]For the purpose of this section, the term *token* denotes the output of word-level tokenization as implemented in NLTK (Bird et al., 2009), mostly consisting of individual words or punctuations signs.

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 (Section 4.2.4). The code for our experiments is available on Github.[8]

### 4.2.1 Motivation

In Section 4.1, we presented a metric for detecting semantic errors in 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 (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 hallucination (Dušek et al., 2019). The 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). Neural metrics such RoMe (Rony et al., 2022) or Data-QuestEval (Rebuffel et al., 2021) do not target specifically content preservation, especially not on the level of individual tokens.

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

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` .

### 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.


Figure 4.3: Rule-based system that we use to generate facts from the input data. The facts are used as 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.

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, that marks each token in the output text for the presence of errors. To make the tabular input data understandable for the PLM, we use a rule-based system to exhaustively generate all the facts that can be derived from the data.[9] Since the context window of the PLM underlying the metric 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 PLM to support its decisions. We describe the individual components of our system below.

**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*).[10]

[9] Similarly to Sections 3.3 and 4.1, we represent the fact as short sentences.

[10] 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. These facts thus cannot be checked with our system.

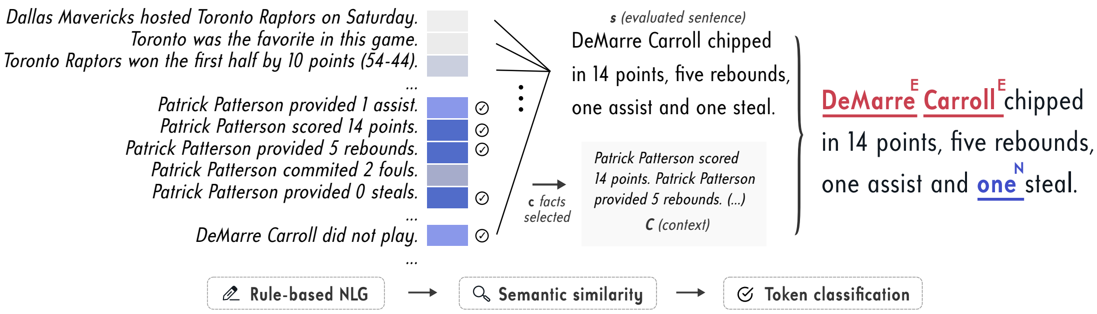



Figure 4.4: An overview of our system. First, we generate the facts from the input table with a rule-based NLG system (see Figure 4.4). For each evaluated sentence $s$, we select $c$ facts with the highest semantic similarity, getting a context $C$. The pair $(C, s)$ is given as an input to a pretrained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

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), that 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 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,[11] 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 (see Section 4.2.4) 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,

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

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 $s$.

**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*.[12] 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) swapping the days of the week.

[12] https://spacy.io

For NUMBER$^{N}$ 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`[13] and `num2words`[14] to convert to digits and back.

### 4.2.4 Experiments

For the error tagger, we train a PyTorch version of RoBERTa from the Huggingface Transfomers repository (Wolf et al., 2019) using the AdamW optimizer (Loshchilov and Hutter, 2017), learning rate $5 \times 10^{-5}$ 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.

**Development Results** The results of our model on the development data are listed in Table 4.2. For our final submission, we selected the model with the best F1-score overall, which is 0.65 (0.61 recall and 0.69 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 a 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.

[13] https://pypi.org/project/text2num/
[14] https://pypi.org/project/num2words/

| Gen. | Data | $c$ | EMR = 0.25 R | EMR = 0.25 P | EMR = 0.25 F1 | EMR = 0.5 R | EMR = 0.5 P | EMR = 0.5 F1 | EMR = 0.75 R | EMR = 0.75 P | EMR = 0.75 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 highlighted in yellow.

**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 0.69 (vs. 0.61 on the development set), and precision is 0.76 (vs. 0.69). 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 are hard to detect in our setup.

| 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.

### 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 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 0.84 recall and 0.88 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 (Kocmi and Federmann, 2023a; Zhao et al., 2023c; 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 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 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, using a combination of a rule-based system, a neural retriever, and a neural token-level classifier. While this metric requires in-domain training data, it enables annotating semantic errors on the level of individual tokens. Future research directions may include removing the need for rule-based preprocessing and improving the flexibility with respect to the output error categories.

# 5 Unified Data Processing

In this chapter, we introduce a single set of experiments in Section 5.1: an approach for unified processing of data-to-text (D2T) generation datasets. We first convert the input data in sixteen 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 helps with streamlining the process of multi-task training, helping us progress towards RQ5. 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 minor implementation tasks and paper writing.

In this section, 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 (introduced in Section 5.1.3) 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 (Section 5.1.4). 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 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 input data and generated outputs. Visualization plays an important role in examining and evaluating scientific data (Kehrer and Hauser, 2013) and can help researchers make more informed design choices. A suitable interface can also encourage researchers to step away from unreliable automatic metrics (Gehrmann et al., 2023) 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 (Sanh et al., 2022; 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 provides multiple features for unified workflows with table-to-text datasets, including:

(1) visualizing individual dataset examples in a 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

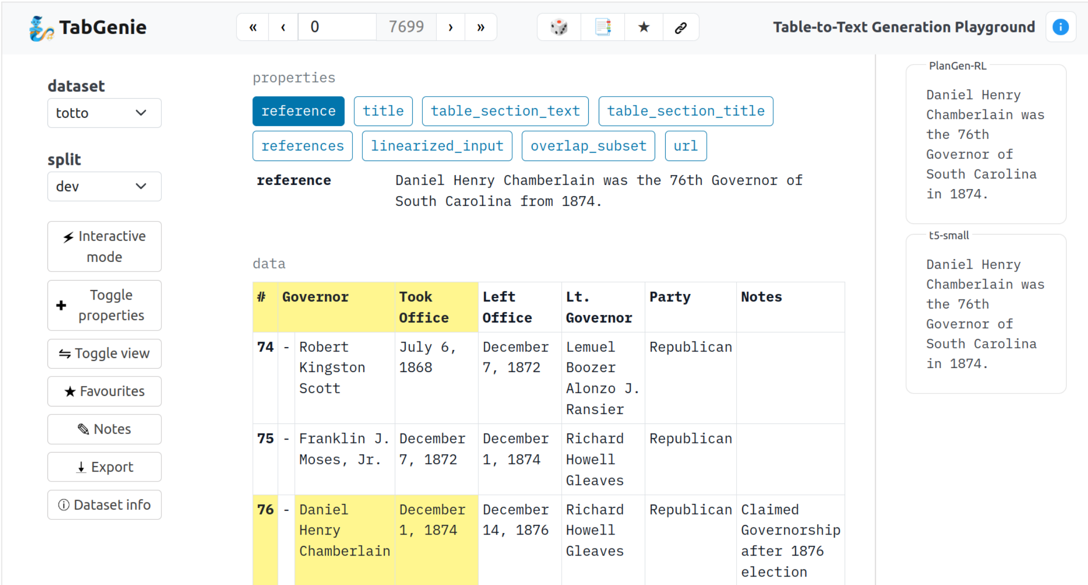



Figure 5.1: The web interface of TabGenie. The *left panel* and the *top navigation bar* contain user controls; the *center panel* shows table properties and table content; the *right panel* contains system outputs.

(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., 2022), we also allow individual cells to be *highlighted*. Highlighted cells are assumed to be preselected for generating an output description. The tables may be accompanied by an additional set of properties (see the `properties` block in Figure 5.1),[4] which we represent as key-value pairs alongside the table. These 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]The properties usually represent table metadata. 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, where the first column contains the keys and is marked as a heading, and the second column contains the values.
- For SportSett:Basketball (Thomson et al., 2020),[5] we merge its two tables *box score* and *line score* 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.[6] 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 datasets through 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.[7] 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] A derivative of Rotowire (Wiseman et al., 2017, see Sections 2.2.6 and 4.2).

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

[7] Compare, e.g., with the ToTTo dataset on Huggingface Datasets 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 the model provides access through a simple 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 JSONL[8] 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 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:

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

[8] https://jsonlines.org

- **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"
  ```

  Exports 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`,[9] a popular lightweight Python-based web framework. The server runs locally and can be configured with a YAML[10] configuration file. On startup, the server loads the data using the `datasets`[11] package. To render web pages, the server uses the `tinyhtml`[12] package and the Jinja[13] templating language.

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

**Frontend** The web frontend is built on HTML5, CSS, Bootstrap,[14] JavaScript, and jQuery.[15] We additionally use the D3.js[16] 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. TabGenie provides a customizable function `get_hf_dataset()`, which linearizes the tables and tokenizes the inputs and references with the provided tokenizer, simplifying preprocessing a dataset to the following:

```
from transformers import AutoTokenizer
import tabgenie as tg

# instantiate a tokenizer
t = 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=t)
```

**Interactive Prompting** TabGenie can be used for observing the impact of various prompts during table-to-text generation. The user customizes the provided `model_api` pipeline to communicate with the model through an API. The API can communicate either with an external model (using e.g. OpenAI API[17]), or with a model running locally (using libraries such as FastAPI[18]). 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.

[14] https://getbootstrap.com/
[15] https://jquery.com
[16] https://d3js.org
[17] https://openai.com/api/
[18] https://fastapi.tiangolo.com

**Error Analysis** TabGenie can help with annotating error categories in the outputs of a table-to-text generation model. The user generates system outputs on a test set and saves them in JSONL format. Through the command-line interface, the user will then generate an XLSX file which can be imported into 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 well-suited for the heavily nested JSON format, which we explore as the input format in Section 6.2. As the framework targets the research community, its use 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 unified 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.

## 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 a web interface. The toolkit also allows to pre-process datasets in a unified format, facilitating their processing with language models (LMs). On top of that, the toolkit provides various practical methods 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 analyze primarily the issue we outlined in RQ5: generalization abilities of neural language models (LMs). Specifically, we investigate how well LMs used for data-to-text (D2T) generation generalize to the data with domains or formats that the models were not specifically trained for. To help us with the investigation, we introduce custom datasets that we collect for the purposes of our experiments.

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 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 an automatic metric and human annotators, we quantify the semantic accuracy of the generated texts with respect to the input data. We find that although the generated 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 co-supervised the project.

In this section, 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 (Section 6.1.1). The question is whether this ability is robust enough, and how accurate the outputs are in cases where these labels are ambiguous or incomplete. To answer these questions, we focus on the specific task of describing a relation between two entities.

For our experiments, we collect Rel2Text (Section 6.1.2): 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 (Sections 6.1.3 to 6.1.6). 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, cf. Section 4.2)—no usable hints about the data semantics.

PLMs such as BART (Lewis et al., 2020) or T5 (Raffel et al., 2020) can quickly adapt to new domains and exhibit robustness to out-of-domain inputs. We investigate to what extent PLMs are limited by the expressivity of the data labels. A suitable testing ground is the task of describing RDF triples in a knowledge graph (KG), which we already tackled in Chapter 3. However, this time we focus on individual triples. Even

[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*.<br>*s* runs on *o*. | multiple equivalent lexical choices |
| *occupant* | P466 | *o* is occupied by *s*.<br>*s* plays at *o*. | semantics depends on entities |
| *parent* | P8810 | *s* is the parent of *o*.<br>*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 the subject and object in the triple, respectively. The Wikidata page for each relation is available at `https://www.wikidata.org/wiki/Property:<property_id>`.

for a single triple, there is a wide range of lexical choices for the relation label, while the entities can be copied verbatim or with only minor morphological changes. For instance, 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. (2021) 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 (2020a) 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 repeatedly shown (Sections 3.2, 3.3, and 4.1), and in line with other works (Xiang et al., 2022; Kale and Rastogi, 2020a; Gupta et al., 2020; Neeraja et al., 2021), transforming triples to text helps for PLMs-based data processing. We train a model for verbalizing single relations and use it to replace manual templates in our system described in Section 3.3.

### 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** (Vrandecic 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

For answering RQ5 on our task, we formulate the following sub-questions:

- **(a)** Are PLMs finetuned for D2T generation able to describe relations *not present in the finetuning corpus*?
- **(b)** How many *training examples* do the PLMs need to generate satisfactory outputs?
- **(c)** How do the PLMs behave when provided *limited lexical cues* about the relation?
- **(d)** 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** First, we divide our REL2TEXT dataset into a training and test split. Next, we use the REL2TEXT *test set* to evaluate a finetuned BART model (Lewis et al., 2020). We train the models on 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, see Section 2.2.6): The 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.

To answer the question (a), 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 the questions (b), (c), and (d).

**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-cased multi-word expressions in the relation labels. For finetuning the models, we linearize the input triples by prepending the individual triple constituents (subject, relation, object) with special tokens that we add to the model vocabulary. In a default scenario, we finetune BART-base (Lewis et al., 2020) 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.

**Compared Systems** In our experiments, we compare the following systems:

[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).

- **Copy Baseline**: We introduce a simple *copy* baseline by outputting the triple constituents separated by space: "`[subject] [relation] [object]`".
- **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 (Kané et al., 2020) along with its individual features: the semantic similarity score

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

(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 the detailed description of the metrics.

**Manual Error Analysis** Based on our preliminary observations, we identify four sources of model errors:

- SEM: semantic errors (text not corresponding to the input data),
- DIR: swap of the relation direction (a special case of a semantic error),
- LIT: too literal (containing awkward or misleading phrasing),
- LEX: grammar/lexical errors.

We further identified two types of input data errors:

- ENT: relations that need to be disambiguated using entities,
- LBL: relations with unclear labels.

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 sources of annotated outputs, we mark all error categories that apply.

### 6.1.5 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 consistency 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.

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

**Masked Relations** Interestingly, models that 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 relation is semantically consistent 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 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 | | | 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.

**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., 2020).

### 6.1.6 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

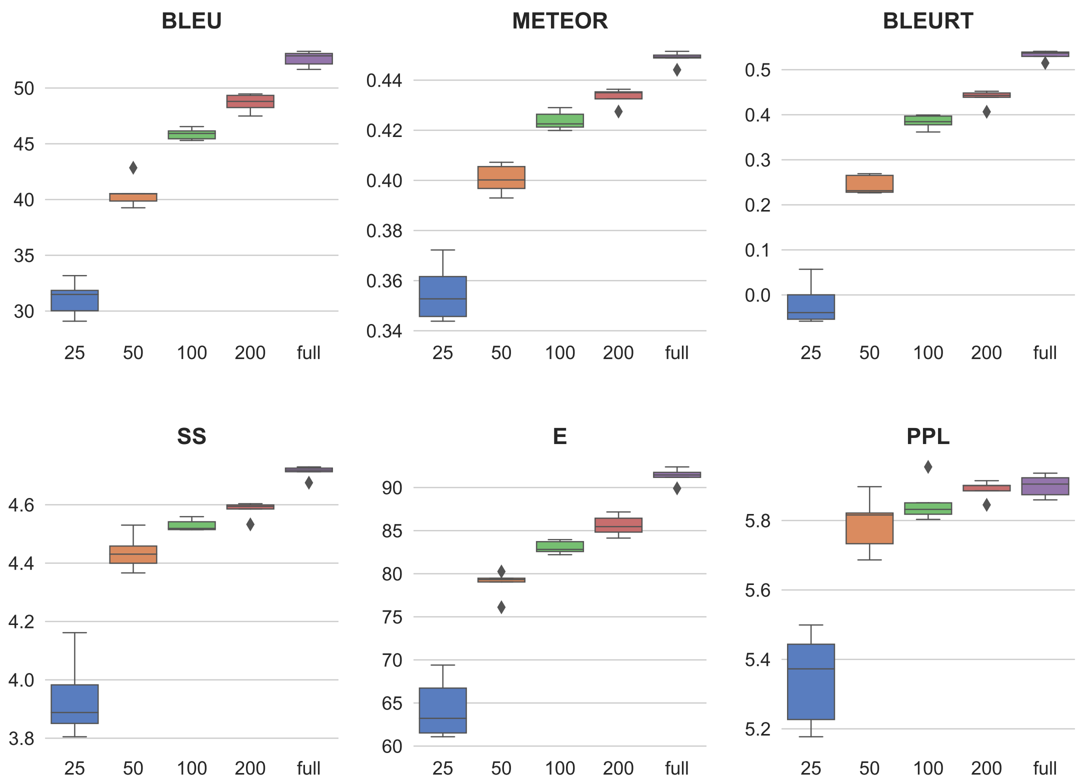



Figure 6.1: Boxplots for selected metrics from Table 6.3 w.r.t. the number of examples (displayed on the $x$-axis, *full* $= 1522$), taking into account variance from individual random seeds.

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 not that common, which is probably down to having only a few examples in our dataset prone to this kind of error.

**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: zero-shot data-to-text generation (introduced in Section 3.3) and tabular reasoning (Gupta et al., 2020; Neeraja et al., 2021).

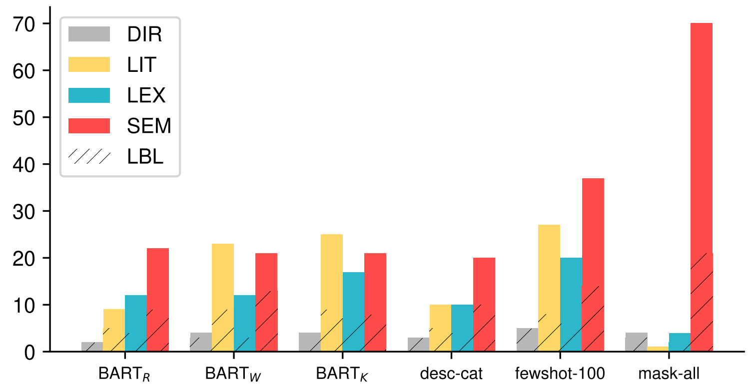



Figure 6.2: Number of annotated errors per model (see Table 6.2 for the description of error categories and Section 6.1.3 for the models). The striped part signifies that the label of the input was marked as unclear.

**Zero-shot Data-to-Text Generation** In Section 3.3, we described our approach for zero-shot D2T generation, where we handcrafted a template for each individual relation. Here, we replicate this setup on the WebNLG dataset, applying the *full-rel2text* model instead of the handcrafted templates. The results are summarized in Table 6.4. 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 entities. 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.

| **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 6.4: Lexical similarity metrics (BLEU, METEOR) and omission (O) and hallucinaton (H) rate; following the setup from Section 3.3.

**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*." (similarly to our fallback template in Section 4.1). 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*). We compute the accuracy of the 3-way NLI outcome on the dev set and three provided test sets ($\alpha_{1,2,3}$). The results are summarized in Table 6.5. Our manual evaluation suggests that sentences generated by our model are indeed more grammatical (even compared to BPR), but for the tabular NLI task, we observe a performance comparable to OPR and BPR 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.

| **premise repr.** | **dev** | $\alpha_1$ | $\alpha_2$ | $\alpha_3$ |
|---|---|---|---|---|
| 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 6.5: NLI accuracy for the dev set and test sets $\alpha_{1,2,3}$ from the InfoTabS dataset. The results are averaged over three random seeds.

### 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 (see Section 2.2.6) 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. How to handle input data with noisy labels is still an open question. We suggest detecting and fixing these cases prior to generation, for example, with a human-in-the-loop setup.

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

**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 the latter case, the models cannot learn 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., 2023c).

**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 (cf. Section 3.3). 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: Analyzing Behaviors of Open LLMs on Data-to-Text Generation* (Kasner and Dušek, 2024), joint work with Ondřej Dušek. The work is accepted to the Proceedings of the 62th Annual Meeting of the Association for Computational Linguistics (ACL 2024).

In this section, we investigate the specifics of D2T generation with large language models (LLMs). To avoid benchmarking the models on data seen during pretraining (Section 6.2.2), we do not use standard D2T datasets described in Section 2.2.6. Instead, we capitalize on the ability of LLMs to process structured data in standard formats such as JSON, CSV, and Markdown without task-specific finetuning. To collect new data, we design a tool for downloading structured data records from public APIs (Section 6.2.2). Using reference-free evaluation methods based on human annotators and GPT-4, we evaluate semantic accuracy of model outputs on the word level (Sections 6.2.3

| Task Id | Domain | Output type | Source | Format |
|---|---|---|---|---|
| `openweather` | Weather | five-day weather forecast | OpenWeather | JSON |
| `gsmarena` | Technology | product description | GSMArena | JSON |
| `ice_hockey` | Sport | ice hockey game summary | RapidAPI | JSON |
| `owid` | Health | chart caption | OurWorldInData | CSV |
| `wikidata` | World facts | entity description | Wikidata | Markdown |

Table 6.6: The domains and tasks included in the QUINTD data collection tool we use for testing D2T generation with LLMs. In our experiments, we download 100 development and 100 test examples of input data for each domain.

and 6.2.4). We find that although the outputs from the LLMs are fluent, the semantic accuracy of the outputs is a major issue: both according to human annotators and GPT-4, more than 80% of the outputs of open LLMs contain at least one semantic error. We publicly release the code, data, and model outputs.[11]

### 6.2.1 Motivation

At the time of writing, the applicability of LLMs (Ouyang et al., 2022; Touvron et al., 2023; Jiang et al., 2023a; Tunstall et al., 2023) to D2T generation remains underexplored. The only existing works evaluated proprietary models on a handful of well-established benchmarks (Axelsson and Skantze, 2023; Yuan and Färber, 2023). However, the current D2T generation benchmarks are not only getting saturated (van Miltenburg et al., 2023), but also promote optimization towards traditional reference-based evaluation metrics, which were shown to correlate poorly with human judgment (Gehrmann et al., 2023; van der Lee et al., 2021; Novikova et al., 2017).

When it comes to the models, using closed LLMs (OpenAI, 2023b,a) is increasingly considered a bad research practice due to its non-reproducibility (Rogers, 2023; Chen et al., 2023c). Contamination of LLMs training data with standard benchmarks then further restricts the space for experiments (Golchin and Surdeanu, 2023; Aiyappa et al., 2023; Balloccu et al., 2024).

Our analysis circumvents these issues by (1) collecting ad-hoc structured data without human-written references, (2) using reference-free evaluation methods for annotating errors on the level of individual tokens, and (3) focusing on open LLMs, which – apart from being more accessible – are also getting more competitive with proprietary models (Zheng et al., 2023; Beeching et al., 2023).

[11] `https://d2t-llm.github.io/`

### 6.2.2 Reference-Free D2T Generation

**Data Collection Tool** We introduce a tool named QUINTD[12] for collecting ad-hoc test sets using public APIs in five different domains. Our main reasons for departing from the traditional scheme of benchmarking on well-established datasets are:

(1) Any published test sets may be potentially included in the training data of LLMs.

(2) Public sources of structured data offer enough resources for creating ad-hoc test sets.

(3) Without human references, our data collection scheme is lightweight and replicable.

Given the available public sources of data, we settled on the five tasks shown in Table 6.6: generating five-day weather forecasts, product descriptions, ice hockey game summaries, chart captions, and entity descriptions. The tasks are based on structured data in common formats: JSON, CSV, and Markdown.

**QUINTD-1 Dataset** Using QUINTD, we collected the dataset for our experiments in this paper (QUINTD-1). The dataset contains 500 examples in the development set and 500 examples in the test set (100 examples per domain for each split). We downloaded the data between November 2023 and January 2024. Note that the dataset contains only *unlabeled* data without any reference outputs (e.g., weather data, but not a textual weather forecast), so the outputs need to be evaluated in a reference-free setup.

**Task Definition** Each example in QUINTD-1 consists of a structured data record $x$ from a domain $\mathcal{D} \in$ {`openweather`, `gsmarena`, `ice_hockey`, `owid`, `wikidata`}. Given $x$ and a prompt $\mathcal{P}$, the goal is to generate natural language output $y$ faithful to the data $x$, according to the instructions in the prompt $\mathcal{P}$ (see Figure 6.3).

### 6.2.3 Experiments

**Models** For our experiments, we selected the following LLMs available under an open license: Llama 2 (Touvron et al., 2023; TogetherAI, 2023),[13] Mistral (Jiang et al., 2023a),[14] and Zephyr (Tunstall et al., 2023).[15]

[12] Quintet of Unlabeled Inputs for Natural Tasks in Data-to-text, pronounced as "quintet"
[13] `https://huggingface.co/togethercomputer/Llama-2-7B-32K-Instruct`
[14] `https://huggingface.co/mistralai/Mistral-7B-Instruct-v0.1`
[15] `https://huggingface.co/HuggingFaceH4/zephyr-7b-beta`

**Prompt**

```
Based on the given data:
 ```
{data}
 ```
Your task is to write a brief, fluent, and coherent single-paragraph
{output_type} in natural language. The text should be balanced and
neutral. Make sure that all the facts mentioned in the text can be
derived from the input data, do *not* add any extra information.
```

**Output prefix**

```
Sure! Here is the {output_type}:
"
```

Figure 6.3: The prompt $\mathcal{P}$ and the model output prefix we used for the experiments in this paper. `data` is filled with the data record $x$ and `output_type` is filled accordingly for each domain $\mathcal{D}$ (see Table 6.6).

The models are instruction-tuned, operate with 32k context, and perform well on recent benchmarks. All the models have 7B parameters and thus fit on a single NVIDIA A40 (48G VRAM) in 16-bit precision. We accessed the models via an API provided by the `text-generation-webui` framework[16] running locally. For the final experiments, we also included GPT-3.5 (`gpt-3.5-turbo-1106`) accessed through the OpenAI API (OpenAI, 2023a).[17]

**Experimental Process** To avoid extensive prompt engineering, we use the same prompt template $\mathcal{P}$ for all the domains and models. For our preliminary experiments, we first wrote down the initial version of the prompt and used the data without further preprocessing. We then iteratively improved our experimental setup by observing outputs on the development set.

**Observations from Preliminary Experiments** Here, we describe all the observations and modifications we made before generating the final outputs on the test set:

[16] https://github.com/oobabooga/text-generation-webui

[17] We only included GPT-3.5 in our final experiments since proprietary models were not our focus. We did not include GPT-4 since we use the model for evaluation (see §6.2.4) and LLMs tend to be biased towards their outputs (Koo et al., 2023; Stureborg et al., 2024).

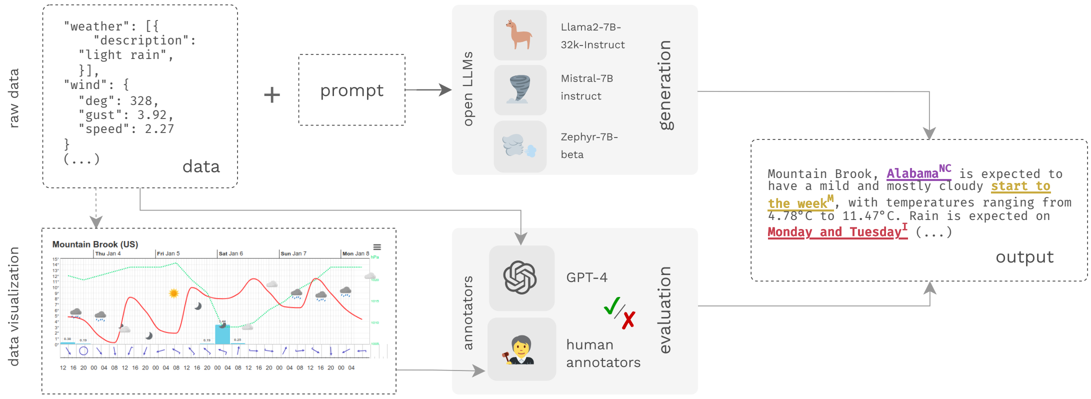



Figure 6.4: Our experimental setup. We first generate the outputs using LLMs that are given raw data and a task-specific prompt. We annotate the token-level semantic errors in the LLM outputs with (a) an automatic metric based on GPT-4 that matches the output to the raw data, and (b) human annotators, who annotate the errors in the output given the data visualization.

- **Any input field may appear in the output.** The models do not always select the most relevant fields for the given output. For example, we observed that the models commonly mention identifiers, timestamps, files, and other metadata, leading to unnatural outputs. To mitigate these issues, we manually picked irrelevant fields and filtered them out.
- **Units need to be specified explicitly.** If measurement units are not specified in the data record, the models tend to resort to their best guess. This may go unnoticed if the unit is evident from the context (e.g., the model will usually not report the temperature in Fahrenheit instead of Celsius), but it may get problematic if the value is ambiguous (e.g., wind speed in km/h versus m/s). Therefore, we explicitly add units to all data records where appropriate.
- **Understandable keys are enough.** On the flip side, we decided not to add extra descriptions to the keys if the key was understandable from its name (e.g., `homeTeam` or `dimensions`). As we discussed in Section 6.1, pretrained models interpret field names correctly as long as they are human-readable. We only include chart metadata for the CSV files in the `owid` domain.

- **Long inputs can be troublesome.** The inputs in some domains can easily get longer than 10-20k tokens. This issue is amplified by the fact that the evaluated LLMs tokenize numbers into individual digits. To accommodate for the long inputs, we picked models that accept up to 32k tokens.[18] However, with long inputs, the GPU memory consumption also gets considerably higher, so we needed to downsample the data in `owid` and `openweather` to keep their length under ~8k tokens.
- **Few-shot experiments are infeasible.** Due to the context-length limitations, we were not able to run few-shot experiments since we could not robustly fit an additional ($x_{\text{example}}$, $y_{\text{example}}$) pair in the prompt. We attempted to include only $y_{\text{example}}$ (making the setup "half-shot"), but we observed that the models then used entities from the example (unrelated to the actual input) in their outputs. Therefore, we decided to leave this line of experiments for future work.
- **Deterministic decoding and sampling are on par.** In our preliminary experiments, we observed a roughly similar output quality for both greedy decoding and sampling.[19] For the final experiments, we decided to use greedy decoding, which is non-parametric and conceptually more suitable for D2T generation.
- **Prefixing the output makes parsing easier.** Even with variations of a *"generate only the output"* instruction appended to the prompt, the models (especially Llama 2) tended to first confirm the request. For that reason, we decided to prefix the input for all the models with *"Sure! Here is the {output_type}: ""*. The opening quote at the end of the prefix allowed us to robustly parse the text simply by stripping the closing quote from the model output.
- **The outputs are fluent but inaccurate.** We observed that the vast majority of model outputs were grammatically and stylistically correct, capturing the output type specified in the prompt. However, we also noticed that the outputs contained many semantic errors (even after emphasizing the focus on semantic accuracy in the prompt, see Figure 6.3). This observation led us to evaluate the model outputs using word-level annotations focused on semantic accuracy errors (see Section 6.2.4).

[18]For this reason, we use Llama-2-7B-32k with 32k token context (TogetherAI, 2023) instead of the official Llama-2-7B-Instruct, which only supports 4k context (Touvron et al., 2023).

[19]We used the `text-generation-webui` default decoding parameters: `temperature`=0.7, `top_p`=0.9, and `top_k`=20.

- **Be careful about subtle bugs.** During our preliminary experiments, we had to fix subtle bugs in our API calls such as incorrect instruction templates[20] or involuntary input truncation. We note that with the apparent ease of API access and robustness of LLMs, such bugs could go unnoticed and artificially skew the model performance.

**Final Experiments** Taking the observations from our preliminary experiments into account, we proceeded to generate the outputs on the test set of QUINTD-1 for token-level error analysis. We first preprocessed the data as mentioned: we stripped out unnecessary fields, added units, and downsampled the data to fit the context. For all the models, we used the prompt in Figure 6.3 and greedy decoding with a maximum of 512 output tokens.

For comparison, we also generated outputs for the same inputs and identical prompts with GPT-3.5. Note that even though we fixed the temperature and seed to 0, the rest of the decoding parameters are inaccessible to us and may differ from the parameters we used for the open models.

### 6.2.4 Evaluation

For evaluation, we focus on identifying *semantic errors* in model outputs. We compare the generated texts to the input data, looking for parts of texts that are not faithful to the input data. We annotate the errors on the token level, considering all the tokens in the output text as potential sources of errors.

We use two complementary referenceless evaluation methods:

- $\mathcal{E}_{\text{hum}}$: **human evaluation** based on crowdsourcing,
- $\mathcal{E}_{\text{gpt}}$: **an automatic metric** based on GPT-4.

The methods use similar instructions and produce outputs with equivalent format. The main idea is to compensate for the shortcomings of each approach: while human evaluation is costly (about 10× more expensive than automatic evaluation in our setup), using only an automatic metric based on a closed LLM would make the evaluation potentially non-reproducible and biased (Kocmi and Federmann, 2023a; Wang et al., 2023c). Reporting the results of both methods should hopefully increase the robustness and replicability of our results.

We use similar error taxonomy and notation are we did in Section 4.2, inspired by Thomson and Reiter (2020). After preliminary examination of the outputs, we settled on four error categories: **INCORRECT**[I], **NOT_CHECKABLE**[NC], **MISLEADING**[S], and **OTHER**[O]. To set a clear boundary between the categories and reach better inter-

[20] https://huggingface.co/docs/transformers/chat_templating

| Error | Description |
|---|---|
| INCORRECT$^{\text{I}}$ | The fact in the text contradicts the data. |
| NOT_CHECKABLE$^{\text{NC}}$ | The fact in the text cannot be checked given the data. |
| MISLEADING$^{\text{S}}$ | The fact in the text is misleading in the given context. |
| OTHER$^{\text{O}}$ | The text is problematic for another reason, e.g., grammatically or stylistically incorrect, irrelevant, or repetitive. |
| **Example** | |
| *data* | **Nokia 3310** \| *color*: black, blue, grey \| *display*: 320x240px |
| *text* | Nokia 3310 is **produced in Finland**$^{\text{NC}}$ and features a **320x320**$^{\text{I}}$ display. It is **available in black color**$^{\text{S}}$. **The data seem to provide only partial information about the phone.**$^{\text{O}}$ |

Table 6.7: Categories of errors annotated in our evaluation and an example demonstrating the error types.

annotator agreement, we decided to keep our taxonomy more high-level this time and not to distinguish between fine-grained categories (e.g., *incorrect name* vs. *incorrect number*). The descriptions of our error categories, as presented in the instructions for annotation, are included in Table 6.7.

**Human-based Evaluation** For the human annotation metric, we prepared a custom web interface, where an annotator is instructed to annotate text spans with the respective error categories. We created custom visualizations for each input data format, as illustrated in Figure 6.4.[21]

We hired annotators on the Prolific crowdsourcing platform.[22] In total, we hired 100 annotators, each annotating 20 examples (4 model outputs for each of the five domains). We selected annotators with at least 10 completed tasks, a 100% approval rate, and English as their primary language. We paid the annotators £9 per hour, according to the platform's recommendations. The median time for completing the annotations was 47 minutes.

**GPT-4-based Evaluation** For automatic evaluation, we leverage the fact that LLM-based metrics can be customized for a particular task without the need for training data. In our experiments, we use a metric based on GPT-4 (`gpt-4-1106-preview`, OpenAI, 2023b), which was shown to be superior to other LLMs in following fine-grained instructions, reaching high correlations with human judgment on evaluating generated texts (Zhao et al., 2023c; Sottana et al., 2023; Kocmi and Federmann, 2023a,b).[23]

[21]We open-sourced our annotation framework as a stand-alone software package, see `https://github.com/kasnerz/factgenie`.

[22]`https://prolific.com`

[23]We confirmed that GPT-3.5 and Llama 3 have lower correlations with human judgments also in our scenario, see Appendix D of Kasner and Dušek (2024).

| | Incorrect | | Not Checkable | | Misleading | | Other | | All cat. | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | $\mathcal{E}_{\text{hum}}$ | $\mathcal{E}_{\text{gpt}}$ | $\mathcal{E}_{\text{hum}}$ | $\mathcal{E}_{\text{gpt}}$ | $\mathcal{E}_{\text{hum}}$ | $\mathcal{E}_{\text{gpt}}$ | $\mathcal{E}_{\text{hum}}$ | $\mathcal{E}_{\text{gpt}}$ | $\mathcal{E}_{\text{hum}}$ | $\mathcal{E}_{\text{gpt}}$ | **Tok.** |
| **Llama 2** | 1.57 | **2.79** | 1.25 | 0.91 | 0.25 | **0.12** | **0.10** | 0.09 | 3.18 | 3.90 | 83.8 |
| **Mistral** | 2.03 | 3.23 | 1.12 | 0.54 | 0.44 | 0.26 | 0.25 | 0.10 | 3.85 | 4.12 | 114.9 |
| **Zephyr** | **1.44** | 2.84 | **0.77** | **0.40** | **0.20** | 0.29 | 0.16 | **0.05** | **2.58** | **3.58** | 98.0 |
| **GPT-3.5** | 0.65 | 1.76 | 0.49 | 0.38 | 0.18 | 0.26 | 0.07 | 0.02 | 1.39 | 2.42 | 84.9 |

Table 6.8: The average *number of errors per output* (lower is better) based on human annotators ($\mathcal{E}_{\text{hum}}$) and GPT-4 ($\mathcal{E}_{\text{gpt}}$). We also include the average number of tokens per output in the rightmost column. The results of the best open LLM are emphasized.

We instantiate $\mathcal{E}_{\text{gpt}}$ with a prompt and a system message describing the task. We instruct the model to produce a JSON output with sequentially ordered errors using the following format:

```
{
  "errors": [{
    "reason": [REASON],
    "text": [TEXT_SPAN],
    "type": [ERROR_CATEGORY]
    },
    ...]
}.
```

Note that we require that the model first generates the free-form text *reason* for the error.[24] Generating the reason comes at almost no extra cost and our cursory observations suggest that requiring it leads to more precise outputs.

We align the model outputs with the original text by string matching on `TEXT_SPAN`, moving the current position forward after each match. We ensure that the model response is a valid JSON using OpenAI's `response_format` parameter.

### 6.2.5 Results and Discussion

A summary of the token-level annotations is given in Tables 6.8 and 6.9.

**How Accurate Are Model Outputs?** Depending on the model, between 76-86% of examples contain an error according to $\mathcal{E}_{\text{hum}}$, suggesting that open LLMs make semantic errors very often. According to $\mathcal{E}_{\text{gpt}}$, the number is as high as 89-94%. The most common error type is INCORRECT[I]. As shown in Table 6.8, all the open LLMs

[24] We did not ask the crowdworkers for free-form reasoning about the errors since that would make the annotation notably more complex.

| | Incorrect | | Not Checkable | | Misleading | | Other | | All cat. | |
|---|---|---|---|---|---|---|---|---|---|---|
| | $\mathcal{E}_{\text{hum}}$ | $\mathcal{E}_{\text{gpt}}$ | $\mathcal{E}_{\text{hum}}$ | $\mathcal{E}_{\text{gpt}}$ | $\mathcal{E}_{\text{hum}}$ | $\mathcal{E}_{\text{gpt}}$ | $\mathcal{E}_{\text{hum}}$ | $\mathcal{E}_{\text{gpt}}$ | $\mathcal{E}_{\text{hum}}$ | $\mathcal{E}_{\text{gpt}}$ |
| **Llama 2** | 53.2% | 80.0% | 57.4% | 44.8% | 17.4% | **8.8%** | **7.6%** | 7.6% | 85.6% | 94.0% |
| **Mistral** | 53.6% | 80.2% | 49.6% | 31.8% | 20.6% | 17.0% | 13.6% | 8.4% | 81.2% | 93.0% |
| **Zephyr** | **46.8%** | **78.0%** | **42.2%** | **25.0%** | **16.2%** | 20.6% | 11.6% | **4.2%** | **75.6%** | **89.4%** |
| **GPT-3.5** | 38.0% | 65.0% | 28.8% | 19.6% | 12.6% | 16.2% | 6.2% | 2.2% | 60.6% | 75.8% |

Table 6.9: The percentage of *outputs containing at least one error* (lower is better) based on human annotators ($\mathcal{E}_{\text{hum}}$) and GPT-4 ($\mathcal{E}_{\text{gpt}}$). The results of the best open LLM are emphasized.

make more than **two statements contradicting the data per output on average**. The NOT_CHECKABLE[NC] errors are also relatively common: more than one per output on average according to $\mathcal{E}_{\text{hum}}$, and at least one being present in more than 25% of examples according to both metrics.

The differences between the open LLMs are not major. Out of the open LLMs, Zephyr has the best results across categories and metrics, followed by Llama 2. However, the outputs of Mistral are longer on average, leaving more space for errors. GPT-3.5 (which we consider separately) does generally better according to both $\mathcal{E}_{\text{gpt}}$and $\mathcal{E}_{\text{hum}}$, although it still makes an error in 60-75% of examples (more than 1 error per example on average). In general, the results show that LLMs make too many semantic errors to be usable in practice for D2T generation in a zero-shot setting.

**Do Evaluation Methods Agree?** To quantify the agreement of $\mathcal{E}_{\text{hum}}$ and $\mathcal{E}_{\text{gpt}}$, we computed the Pearson correlation coefficient between the error counts on the level of tokens, examples, and domains as follows (note that each error category was considered separately):

- For $r_{\text{domain}}$, we used the average error counts per domain.[25]
- For $r_{\text{example}}$, we used the count of errors per example.
- For $r_{\text{token}}$, we used binary indicators marking whether each word is labeled as an error.

We see that the correlation on the level of words is weak ($r_{\text{token}} = 0.26$) but gets better on the example level ($r_{\text{example}} = 0.52$) and even better on the domain level ($r_{\text{domain}} = 0.93$).

We also measure inter-annotator agreement between human annotators. For that, we obtained annotations from two annotators for 100 model outputs. The results are similar: the annotators agree weakly on the token level ($r_{\text{token}} = 0.36$), stronger on the example level ($r_{\text{example}} = 0.53$), and even stronger on the domain level ($r_{\text{domain}} = 0.85$).

[25]See Appendix F of Kasner and Dušek (2024) for the results for individual domains.

We conclude that while the details regarding error spans and categories may vary, the annotators as well as GPT-4 generally agree on the accuracy of model outputs for a given set of examples. In the future, the agreement could be improved by measuring errors on the phrase level (Vamvas and Sennrich, 2022).

**Recommendations for Future Work** Based on the above results, we formulate a few recommendations for future works exploring D2T generation with LLMs:

- **Focus on semantic accuracy.** The output of LLMs is satisfactory regarding the style, format, and purpose of the text. However, the amount of semantic errors remains very high. Improving the semantic accuracy of the models (Li et al., 2022), along with new model-based evaluation metrics (Liu et al., 2023; Xu et al., 2023a), could thus help to bring improve LLM-based D2T generation systems where it is most needed.
- **Use long-context models.** The memory issues with long context, making few-shot experiments infeasible, can potentially be solved by using more efficient long-context models equipped with Flash Attention (Dao et al., 2022) and fast inference libraries such as `llama.cpp`,[26] especially in the light of the recent rapid increase of the available context window size of LLMs (Bai et al., 2023; Munkhdalai et al., 2024).
- **Test the models in the wild.** Except for using an ad-hoc dataset of real-world data as we did in our work, the validity of D2T evaluation beyond the experimental setting can also be ensured by continuous evaluation with human users (Zheng et al., 2023) and evaluating the real-world impact of the systems (Reiter, 2023).
- **Multilinguality is an opportunity.** With the recent efforts in extending D2T generation to low-resource languages (Cripwell et al., 2023), multilingual D2T generation with open LLMs seems a promising direction. Although we did not go beyond English, initial steps were already taken by works such as Lorandi and Belz (2023) and Lorandi and Belz (2024).

[26] `https://github.com/ggerganov/llama.cpp`

## 6.3 Conclusion

To improve D2T generation systems based on neural LMs, it is important to know the *input data* we are dealing with, along with the strong and weak points of the *models* we are using. We had both of these points in mind when conducting the experiments described in this chapter. We analyzed the capabilities of pretrained models of various sizes, focusing on openly available models, and provided recommendations for future directions in D2T generation with these models. Our insights are based on custom datasets, carefully assembled for the purpose of the analysis.

In Section 6.1, we investigated how well PLMs can generalize to unseen data labels. We collected a dataset for triple-to-text generation with a large variety of unique relation labels. Using the dataset, we analyzed the performance of finetuned PLMs in various scenarios, concluding that for good generalization, it is important to equip the data with unambiguous, human-readable labels.

In Section 6.2, we presented an exploratory study into D2T generation with open LLMs. We used ad-hoc collected data in standard formats from five domains, prompting LLMs to generate texts based on the data. Using a combination of a GPT-4-based metric and human evaluation, we found out that the outputs contain unacceptably large amounts of semantic errors. In light of these findings, we recommended improving semantic accuracy as the main future direction for D2T generation with LLMs.

We conclude that pretraining equips LMs with the abilities to generate text from data with new data labels or formats. This ability is most apparent with LLMs, that can perform a large variety of D2T generation tasks without the need for task-specific training data. However, their zero-shot abilities are a double-edged sword, as the text fluency text may conceal semantic errors in the output. Finetuned models are the safer choice for small, targetted operations, such as the ones we performed in Sections 3.3, 4.1, and 6.1, where consistency across examples is required.

# 7 Conclusions

We set out to explore how to use language models (LMs) to improve the robustness and fluency of data-to-text (D2T) generation systems. The issues we had to deal with—semantic inaccuracies of generated texts, lack of automatic evaluation metrics, heterogeneous data formats, and unknown scope of LMs abilities—are as pressing as ever, despite the increase in LM capabilities over the past few years. That is not to say that there has not been any progress: the opposite is evidenced by numerous works introduced in Chapter 2 and others we did not get the chance to mention.

We hope to have also contributed to the progress with the answers to our research questions outlined in Section 1.1, including:

- **RQ1**: Our finding that simple LM-based approaches can be used for generating fluent outputs from structured data (Sections 3.1 and 6.2),
- **RQ2**: Our finding that preprocessing the data with templates and rule-based systems can help downstream LM components (Sections 3.2, 3.3, 4.1, and 4.2),
- **RQ3**: Our finding that constraining LM to improving text quality helps to improve the semantic accuracy of the system outputs (Sections 3.2 and 3.3),
- **RQ4**: Our finding that LMs can be also used for automatic metrics for evaluating semantic accuracy (Sections 4.1 and 4.2),
- **RQ5**: Our finding that the best way to leverage LM pretraining is to use standardized and understandable input formats (Sections 5.1, 6.1, and 6.2).

At this point, we are well-equipped to also discuss a few meta-questions. Are we solving the right problems in D2T generation? Do we want to continue integrating LMs in D2T generation systems? And is there value in developing specialized approaches, or are all the problems going to be solved by using ever larger models?

A good starting point to answering these questions is realizing that language models (and LLMs in particular) are here to stay. The ongoing proliferation of LLMs in natural language processing (Min et al., 2024; Zhao et al., 2023a; Naveed et al., 2023) makes it hard to imagine a subfield that would be left intact by their impact. There is a solid reason for that: with LLMs, certain things unimaginable during previous decades—such as fine-grained steering of a system using natural language instructions—are now becoming possible. We can expect that D2T *without LLMs* would, to put it bluntly, soon start to feel awkward. People are already becoming used to consuming fluent texts and seamlessly interacting with language generation systems, aspects that are hard to replace with non-LLM systems. As we stated in the introduction, D2T generation is primarily about simplifying interactions for end users, so these aspects cannot be neglected if D2T research is to stay relevant.

At the same time, users are (hopefully, along with us researchers) becoming aware of the limitations of LM-based systems. Even the most powerful LMs nowadays cannot reliably perform symbolic tasks such as basic arithmetic operations (Qian et al., 2023), understand reflexivity of relations between entities (Berglund et al., 2024), or recognize unanswerable or unknowable questions (Yin et al., 2023). All of these issues are tied to D2T generation: for example, understanding the scope of relations (and recognizing the ambiguous ones) is crucial for the correct verbalization of knowledge graphs, as we discussed in Section 6.1. It is reasonable to expect that these issues will not be fully solved with further scaling of LLMs or minor architectural improvements. We therefore need to tread carefully when integrating LMs into D2T generation systems: a system relying too heavily on LMs may not be ever able to guarantee outputs accurate enough for day-to-day usage, let alone for sensitive applications.

It would be, however, counter-productive to dismiss LMs by likening them to a "black box", picking on their unpredictable behavior. It is better to acknowledge that even the *black boxes* are still *boxes*: components that can be embedded in a larger system. As we repeatedly showed throughout the thesis, such a component can be helpful when used wisely. We can, for example, over-generate LM outputs and select only the relevant ones (Section 3.2) or train the LM in a way that its outputs are more predictable (Section 3.3). We can also build our system around the tasks on which LMs achieve state-of-the-art performance, such as natural language inference (Section 4.1) or text classification (Section 4.2).

Looking at recent developments, we only scratched the surface of what is possible. Even now, we can go beyond the lexicalization and surface realization steps—which were the primary focus of this thesis—by connecting LMs to tools such as a Python interpreter or an SQL engine, enabling LMs to perform content selection as well (Cao et al., 2023; Jiang et al., 2023b; Gemmell and Dalton, 2023). We can imagine that by combining code execution with approaches such as chain-of-thought prompting and its advanced variants (Wei et al., 2022b; Chu et al., 2023), the systems will be able to automatically perform logical operations over the data to derive interesting insights (Zhao et al., 2023b; Chen et al., 2020a,c). Soon, we may think of literal data transcription or shallow data summarization the way we think about, for example, word-for-word translation: as an approach that is too basic to even consider using. LM-powered systems could thus get us closer to presenting useful insights from large-scale structured data, the ultimate purpose of D2T generation.

As the systems get better at handling multiple natural language processing (NLP) tasks in a unified way, the role of individual tasks—such as D2T generation—could become somewhat less important. A single LM-based component could jointly tackle all the tasks that are currently thought of as stand-alone: natural language understanding, text-to-SQL, data mining, question answering, or data-to-text generation (Schopf et al., 2023; Chen et al., 2021). Rather than in the tasks themselves, the researchers would then specialize in auxiliary tools used on top of the LMs such as approaches for steering the generation process, output quality assurance, or personalization (Zhang et al., 2024; Chen et al., 2023b).

Evaluating future systems may get more difficult. The first step we need to focus on is making the current evaluation measures reflect actual system improvements (Gehrmann et al., 2023; van Miltenburg et al., 2023). As we discussed in Section 6.2, this can mean moving away from traditional benchmarks, that can get saturated (Kiela et al., 2021; Raji et al., 2021), or even included in the LLM training data (Balloccu et al., 2024). In the long run, we should also focus on the ecological validity of the systems we are developing (Reiter, 2017). To achieve that, we should focus more on extrinsic evaluation, i.e., evaluating the system as a whole instead of its individual components (cf. Reiter et al., 2003; Eugenio et al., 2002, see also Section 2.2.7). These measures are harder to iterate on but give us a better picture of the real-world impact of the systems we are building.

A final recommendation, that perhaps should have come a bit sooner: *look in the data and try to perform the task yourselves first.* As much as we introduced the data as the language of computers, its content and structure can always be traced down to human-made inputs. Our experiments—such as the ones in Sections 5.1 and 6.1—made it clear to us that the inputs themselves are often incomplete and hard to understand. In these cases, trying to present the data in understandable form is the same as translating gibberish language. Until the computers can reliably fix our errors, we need to do the legwork and fix them ourselves. In other words: keep on learning the language of the data we are producing.

# 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, WMT at ACL 2008*, p. 115–118, Columbus, Ohio, USA, 2008. Available at: https://aclanthology.org/W08-0312/.

Agarwal, O. – Ge, H. – Shakeri, S. – Al-Rfou, R. Knowledge Graph Based Synthetic Corpus Generation for Knowledge-Enhanced Language Model Pre-training. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2021*, p. 3554–3565, Online, 2021. doi: 10.18653/V1/2021.NAACL-MAIN.278. Available at: https://doi.org/10.18653/v1/2021.naacl-main.278.

Aiyappa, R. – An, J. – Kwak, H. – Ahn, Y.-y. Can We Trust the Evaluation on ChatGPT? In *Proceedings of the 3rd Workshop on Trustworthy Natural Language Processing (TrustNLP 2023)*, p. 47–54, Toronto, Canada, July 2023. doi: 10.18653/v1/2023.trustnlp-1.5. Available at: https://aclanthology.org/2023.trustnlp-1.5.

Amaral, G. – Rodrigues, O. – Simperl, E. WDV: A Broad Data Verbalisation Dataset Built from Wikidata. In *The Semantic Web - ISWC 2022 - 21st International Semantic Web Conference, Proceedings*, 13489 / *Lecture Notes in Computer Science*, p. 556–574, Virtual Event, 2022. doi: 10.1007/978-3-031-19433-7_32. Available at: https://doi.org/10.1007/978-3-031-19433-7_32.

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/.

Anthropic, A. The Claude 3 Model Family: Opus, Sonnet, Haiku. *Claude-3 Model Card*. 2024. Available at: https://www-cdn.anthropic.com/de8ba9b01c9ab7cbabf5c33b80b7bbc618857627/Model_Card_Claude_3.pdf.

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, Tilburg University, The Netherlands, 2018. doi: 10.18653/V1/W18-6515. Available at: https://doi.org/10.18653/v1/w18-6515.

Aroca-Ouellette, S. – Rudzicz, F. On Losses for Modern Language Models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020*, p. 4970–4981, Online, 2020. doi: 10.18653/V1/2020.EMNLP-MAIN.403. Available at: https://doi.org/10.18653/v1/2020.emnlp-main.403.

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

Auer, S. – Bizer, C. – Kobilarov, G. – Lehmann, J. – Cyganiak, R. – Ives, Z. G. DBpedia: A Nucleus for a Web of Open Data. In *The Semantic Web, 6th International Semantic Web Conference, 2nd Asian Semantic Web Conference, ISWC 2007 + ASWC 2007*, 4825 / *Lecture Notes in Computer Science*, p. 722–735, Busan, Korea, 2007. doi: 10.1007/978-3-540-76298-0_52. Available at: https://doi.org/10.1007/978-3-540-76298-0_52.

Axelsson, A. – Skantze, G. Using Large Language Models for Zero-Shot Natural Language Generation from Knowledge Graphs. *CoRR*. 2023, abs/2307.07312. doi: 10.48550/ARXIV.2307.07312. Available at: https://doi.org/10.48550/arXiv.2307.07312.

Ba, L. J. – Kiros, J. R. – Hinton, G. E. Layer Normalization. *CoRR*. 2016, abs/1607.06450. Available at: http://arxiv.org/abs/1607.06450.

Bahdanau, D. – Cho, K. – Bengio, Y. Neural Machine Translation by Jointly Learning to Align and Translate. In *3rd International Conference on Learning Representations, ICLR 2015Proceedings*, San Diego, CA, USA, 2015. Available at: http://arxiv.org/abs/1409.0473.

Bai, Y. – Lv, X. – Zhang, J. – Lyu, H. – Tang, J. – Huang, Z. – Du, Z. – Liu, X. – Zeng, A. – Hou, L. – Dong, Y. – Tang, J. – Li, J. LongBench: A Bilingual, Multitask Benchmark for Long Context Understanding. *CoRR*. 2023, abs/2308.14508. doi: 10.48550/ARXIV.2308.14508. Available at: https://doi.org/10.48550/arXiv.2308.14508.

Balakrishnan, A. – Rao, J. – Upasani, K. – White, M. – Subba, R. Constrained Decoding for Neural NLG From Compositional Representations in Task-Oriented Dialogue. In *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. *CoRR*. 2022, abs/2206.13435. doi: 10.48550/ARXIV.2206.13435. Available at: https://doi.org/10.48550/arXiv.2206.13435.

BALLOCCU, S. – SCHMIDTOVÁ, P. – LANGO, M. – DUŠEK, O. Leak, Cheat, Repeat: Data Contamination and Evaluation Malpractices in Closed-Source LLMs. In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2024 - Volume 1: Long Papers*, p. 67–93, St. Julian's, Malta, 2024. Available at: https://aclanthology.org/2024.eacl-long.5.

BANERJEE, S. – LAVIE, A. METEOR: An Automatic Metric for MT Evaluation with Improved Correlation with Human Judgments. In *Proceedings of the Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization at ACL 2005*, p. 65–72, Ann Arbor, Michigan, USA, 2005. Available at: https://aclanthology.org/W05-0909/.

BANGALORE, S. – RAMBOW, O. Corpus-Based Lexical Choice in Natural Language Generation. In *38th Annual Meeting of the Association for Computational Linguistics*, p. 464–471, Hong Kong, China, 2000. doi: 10.3115/1075218.1075277. Available at: https://aclanthology.org/P00-1059/.

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 Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18)*, p. 5020–5027, New Orleans, Louisiana, USA, 2018. doi: 10.1609/AAAI.V32I1.11944. Available at: https://doi.org/10.1609/aaai.v32i1.11944.

BARZILAY, R. – LEE, L. Catching the Drift: Probabilistic Content Models, with Applications to Generation and Summarization. In *Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics, HLT-NAACL 2004*, p. 113–120, Boston, Massachusetts, USA, 2004. Available at: https://aclanthology.org/N04-1015/.

BARZILAY, R. – MCKEOWN, K. R. Sentence Fusion for Multidocument News Summarization. *Comput. Linguistics*. 2005, 31, 3, p. 297–328. doi: 10.1162/089120105774321091. Available at: https://doi.org/10.1162/089120105774321091.

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, California, 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.

Beeching, E. – Fourrier, C. – Habib, N. – Han, S. – Lambert, N. – Rajani, N. – Sanseviero, O. – Tunstall, L. – Wolf, T. Open LLM Leaderboard. https://huggingface.co/spaces/HuggingFaceH4/open_llm_leaderboard, 2023.

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.* 2008, 14, 4, p. 431–455. doi: 10.1017/S1351324907004664. Available at: https://doi.org/10.1017/S1351324907004664.

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, INLG 2020*, p. 183–194, Dublin, Ireland, 2020. doi: 10.18653/V1/2020.INLG-1.24. Available at: https://doi.org/10.18653/v1/2020.inlg-1.24.

Bengio, Y. – Ducharme, R. – Vincent, P. – Janvin, C. A Neural Probabilistic Language Model. *J. Mach. Learn. Res.* 2003, 3, p. 1137–1155. Available at: http://jmlr.org/papers/v3/bengio03a.html.

Berglund, L. – Tong, M. – Kaufmann, M. – Balesni, M. – Stickland, A. C. – Korbak, T. – Evans, O. The Reversal Curse: LLMs trained on "A is B" fail to learn "B is A". In *The Twelfth International Conference on Learning Representations*, 2024. Available at: https://openreview.net/forum?id=GPKTIktA0k.

Bird, S. – Klein, E. – Loper, E. *Natural Language Processing With Python: Analyzing Text With the Natural Language Toolkit.* O'Reilly Media, Inc., 2009.

Bishop, C. M. Pattern Recognition and Machine Learning. *Springer.* 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. B. et al. Language Models are Few-Shot Learners. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020*, 2020. Available at: https://proceedings.neurips.cc/paper/2020/hash/1457c0d6bfcb4967418bfb8ac142f64a-Abstract.html.

Bubeck, S. – Chandrasekaran, V. – Eldan, R. – Gehrke, J. – Horvitz, E. – Kamar, E. – Lee, P. – Lee, Y. T. – Li, Y. – Lundberg, S. M. – Nori, H. – Palangi, H. – Ribeiro, M. T. – Zhang, Y. Sparks of Artificial General Intelligence: Early Experiments with GPT-4. *CoRR.* 2023, abs/2303.12712. doi: 10.48550/ARXIV.2303.12712. Available at: https://doi.org/10.48550/arXiv.2303.12712.

Budzianowski, P. – Wen, T. – Tseng, B. – Casanueva, I. – Ultes, S. – Ramadan, O. – Gasic, M. MultiWOZ - A Large-Scale Multi-Domain Wizard-of-Oz Dataset for Task-Oriented Dialogue Modelling. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, p. 5016–5026, Brussels, Belgium, 2018. Available at: https://aclanthology.org/D18-1547/.

Calizzano, R. – Ostendorff, M. – Rehm, G. Ordering Sentences and Paragraphs with Pre-Trained Encoder-Decoder Transformers and Pointer Ensembles. In *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.

Cao, Y. – Chen, S. – Liu, R. – Wang, Z. – Fried, D. API-Assisted Code Generation for Question Answering on Varied Table Structures. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023*, p. 14536–14548, Singapore, 2023. doi: 10.18653/V1/2023.EMNLP-MAIN.897. Available at: https://doi.org/10.18653/v1/2023.emnlp-main.897.

Celikyilmaz, A. – Clark, E. – Gao, J. Evaluation of Text Generation: A Survey. *CoRR*. 2020, abs/2006.14799. Available at: https://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 *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, p. 1–14, Vancouver, Canada, August 2017. 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, INLG 2021*, p. 325–330, Aberdeen, Scotland, UK, 2021a. doi: 10.18653/V1/2021.INLG-1.36. Available at: https://doi.org/10.18653/v1/2021.inlg-1.36.

Chang, E. – Shen, X. – Zhu, D. – Demberg, V. – Su, H. Neural Data-to-Text Generation with LM-based Text Augmentation. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume, EACL 2021*, p. 758–768, Online, 2021b. doi: 10.18653/V1/2021.EACL-MAIN.64. Available at: https://doi.org/10.18653/v1/2021.eacl-main.64.

Chapman, C. L. – Hillebrand, L. P. – Stenzel, M. R. – Deußer, T. – Biesner, D. – Bauckhage, C. – Sifa, R. Towards Generating Financial Reports from Tabular Data using Transformers. In *Machine Learning and Knowledge Extraction - 6th IFIP TC 5, TC 12, WG 8.4, WG 8.9, WG 12.9 International Cross-Domain Conference, CD-MAKE 2022, Proceedings*, 13480 / *Lecture Notes in Computer Science*, p. 221–232, Vienna, Austria, 2022. doi: 10.1007/978-3-031-14463-9_14. Available at: https://doi.org/10.1007/978-3-031-14463-9_14.

Chen, J. – Xu, R. – Zeng, W. – Sun, C. – Li, L. – Xiao, Y. Converge to the Truth: Factual Error Correction via Iterative Constrained Editing. In *Thirty-Seventh AAAI Conference on Artificial Intelligence, AAAI 2023, Thirty-Fifth Conference on Innovative Applications of Artificial Intelligence, IAAI 2023, Thirteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2023*, p. 12616–12625, Washington, DC, USA, 2023a. doi: 10.1609/AAAI.V37I11.26485. Available at: https://doi.org/10.1609/aaai.v37i11.26485.

Chen, J. – Liu, Z. – Huang, X. – Wu, C. – Liu, Q. – Jiang, G. – Pu, Y. – Lei, Y. – Chen, X. – Wang, X. – Lian, D. – Chen, E. When Large Language Models Meet Personalization: Perspectives of Challenges and Opportunities. *CoRR*. 2023b, abs/2307.16376. doi: 10.48550/ARXIV.2307.16376. Available at: https://doi.org/10.48550/arXiv.2307.16376.

Chen, L. – Zaharia, M. – Zou, J. How Is ChatGPT's Behavior Changing Over Time? *CoRR*. 2023c, abs/2307.09009. doi: 10.48550/ARXIV.2307.09009. Available at: https://doi.org/10.48550/arXiv.2307.09009.

Chen, S. – Zhang, Y. – Yang, Q. Multi-Task Learning in Natural Language Processing: An Overview. *CoRR*. 2021, abs/2109.09138. Available at: https://arxiv.org/abs/2109.09138.

Chen, W. – Chen, J. – Su, Y. – Chen, Z. – Wang, W. Y. Logical Natural Language Generation from Open-Domain Tables. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020*, p. 7929–7942, Online, 2020a. doi: 10.18653/V1/2020.ACL-MAIN.708. Available at: https://doi.org/10.18653/v1/2020.acl-main.708.

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 2020*, p. 8635–8648, Online, 2020b. doi: 10.18653/V1/2020.EMNLP-MAIN.697. Available at: https://doi.org/10.18653/v1/2020.emnlp-main.697.

Chen, Y. – Eger, S. MENLI: Robust Evaluation Metrics from Natural Language Inference. *CoRR*. 2022, abs/2208.07316. doi: 10.48550/ARXIV.2208.07316. Available at: https://doi.org/10.48550/arXiv.2208.07316.

Chen, Z. – Chen, W. – Zha, H. – Zhou, X. – Zhang, Y. – Sundaresan, S. – Wang, W. Y. Logic2Text: High-Fidelity Natural Language Generation from Logical Forms. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, EMNLP 2020 / *Findings of ACL*, p. 2096–2111, Online Event, 2020c. doi: 10.18653/V1/2020.FINDINGS-EMNLP.190. Available at: https://doi.org/10.18653/v1/2020.findings-emnlp.190.

Chen, Z. – Eavani, H. – Chen, W. – Liu, Y. – Wang, W. Y. Few-Shot NLG with Pre-Trained Language Model. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020*, p. 183–190, Online, 2020d. doi: 10.18653/V1/2020.ACL-MAIN.18. Available at: https://doi.org/10.18653/v1/2020.acl-main.18.

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, EMNLP 2016*, p. 551–561, Austin, Texas, USA, 2016. doi: 10.18653/V1/D16-1053. Available at: https://doi.org/10.18653/v1/d16-1053.

Cheng, Z. – Dong, H. – Wang, Z. – Jia, R. – Guo, J. – Gao, Y. – Han, S. – Lou, J. – Zhang, D. HiTab: A Hierarchical Table Dataset for Question Answering and Natural Language Generation. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2022*, p. 1094–1110, Dublin, Ireland, 2022. doi: 10.18653/V1/2022.ACL-LONG.78. Available at: https://doi.org/10.18653/v1/2022.acl-long.78.

Chiang, D. C. – Lee, H. Can Large Language Models Be an Alternative to Human Evaluations? In *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. – Gülçehre, Ç. – 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 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, p. 1724–1734, 2014. doi: 10.3115/V1/D14-1179. Available at: https://doi.org/10.3115/v1/d14-1179.

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

Chu, Z. – Chen, J. – Chen, Q. – Yu, W. – He, T. – Wang, H. – Peng, W. – Liu, M. – Qin, B. – Liu, T. A Survey of Chain of Thought Reasoning: Advances, Frontiers and Future. *CoRR*. 2023, abs/2309.15402. doi: 10.48550/ARXIV.2309.15402. Available at: https://doi.org/10.48550/arXiv.2309.15402.

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. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, (Volume 1: Long Papers)*, p. 7282–7296, Virtual Event, 2021. doi: 10.18653/V1/2021.ACL-LONG.565. Available at: https://doi.org/10.18653/v1/2021.acl-long.565.

Colas, A. M. – Sadeghian, A. – Wang, Y. – Wang, D. Z. EventNarrative: A Large-scale Event-centric Dataset for Knowledge Graph-to-Text Generation. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks 1, NeurIPS Datasets and Benchmarks 2021*, 2021. Available at: https://datasets-benchmarks-proceedings.neurips.cc/paper/2021/hash/a3f390d88e4c41f2747bfa2f1b5f87db-Abstract-round1.html.

Cotterell, R. – Svete, A. – Meister, C. – Liu, T. – Du, L. Formal Aspects of Language Modeling. *CoRR*. 2023, abs/2311.04329. doi: 10.48550/ARXIV.2311.04329. Available at: https://doi.org/10.48550/arXiv.2311.04329.

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 *Proceedings of the Workshop on Multimodal, Multilingual Natural Language Generation and Multilingual WebNLG Challenge (MM-NLG 2023)*, p. 55–66, Prague, Czech Republic, 2023. Available at: https://aclanthology.org/2023.mmnlg-1.6.

Dale, R. Natural Language Generation: The Commercial State of the Art in 2020. *Nat. Lang. Eng.* 2020, 26, 4, p. 481–487. doi: 10.1017/S135132492000025X. Available at: https://doi.org/10.1017/S135132492000025X.

Dale, R. Navigating the Text Generation Revolution: Traditional Data-to-Text NLG Companies and the Rise of ChatGPT. *Nat. Lang. Eng.* 2023, 29, 4, p. 1188–1197. doi: 10.1017/S1351324923000347. Available at: https://doi.org/10.1017/s1351324923000347.

Dao, T. – Fu, D. Y. – Ermon, S. – Rudra, A. – Ré, C. FlashAttention: Fast and Memory-Efficient Exact Attention With IO-Awareness. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022*, New Orleans, LA, USA, 2022. Available at: http://papers.nips.cc/paper_files/paper/2022/hash/67d57c32e20fd0a7a302cb81d36e40d5-Abstract-Conference.html.

Demir, S. – Carberry, S. – McCoy, K. F. Generating Textual Summaries of Bar Charts. In *INLG 2008 - Proceedings of the Fifth International Natural Language Generation Conference, June 12-14, 2008, Salt Fork*, Ohio, USA, 2008. Available at: https://aclanthology.org/W08-1103/.

Demir, S. – Carberry, S. – McCoy, K. F. Summarizing Information Graphics Textually. *Comput. Linguistics*. 2012, 38, 3, p. 527–574. doi: 10.1162/COLI_A_00091. Available at: https://doi.org/10.1162/COLI_a_00091.

Devlin, J. – Chang, M. – Lee, K. – Toutanova, K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, Volume 1 (Long and Short Papers)*, p. 4171–4186, USA, 2019. doi: 10.18653/V1/N19-1423. Available at: https://doi.org/10.18653/v1/n19-1423.

Dhingra, B. – Faruqui, M. – Parikh, A. P. – Chang, M. – Das, D. – Cohen, W. W. Handling Divergent Reference Texts when Evaluating Table-to-Text Generation. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Volume 1: Long Papers*, p. 4884–4895, Florence, Italy, 2019. doi: 10.18653/V1/P19-1483. Available at: https://doi.org/10.18653/v1/p19-1483.

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. Available at: https://aclanthology.org/www.mt-archive.info/HLT-2002-Doddington.pdf.

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. Available at: https://doi.org/10.48550/arXiv.2301.00234.

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. *Neurocomputing*. 2022, 503, p. 92–108. doi: 10.1016/J.NEUCOM.2022.06.111. Available at: https://doi.org/10.1016/j.neucom.2022.06.111.

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. *Comput. Linguistics*. 2022, 48, 3, p. 733–763. doi: 10.1162/COLI_A_00445. Available at: https://doi.org/10.1162/coli_a_00445.

Dušek, O. – Jurčíček, F. Sequence-to-Sequence Generation for Spoken Dialogue via Deep Syntax Trees and Strings. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Volume 2: Short Papers*, Berlin, Germany, 2016. doi: 10.18653/V1/P16-2008. Available at: https://doi.org/10.18653/v1/p16-2008.

Dušek, O. – Jurčíček, 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 of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Volume 1: Long Papers*, p. 451–461, Beijing, China, 2015. doi: 10.3115/V1/P15-1044. Available at: https://doi.org/10.3115/v1/p15-1044.

Dušek, O. – Jurčíček, F. Neural Generation for Czech: Data and Baselines. In *Proceedings of the 12th International Conference on Natural Language Generation, INLG 2019*, p. 563–574, Tokyo, Japan, 2019. doi: 10.18653/V1/W19-8670. Available at: https://aclanthology.org/W19-8670/.

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, INLG 2020*, p. 131–137, Dublin, Ireland, 2020. doi: 10.18653/V1/2020.INLG-1.19. Available at: https://doi.org/10.18653/v1/2020.inlg-1.19.

Dušek, O. – Howcroft, D. M. – Rieser, V. Semantic Noise Matters for Neural Natural Language Generation. In *Proceedings of the 12th International Conference on Natural Language Generation, INLG 2019*, p. 421–426, Tokyo, Japan, 2019. doi: 10.18653/V1/W19-8652. Available at: https://aclanthology.org/W19-8652/.

Dušek, O. – Novikova, J. – Rieser, V. Evaluating the State-of-the-Art of End-to-End Natural Language Generation: The E2E NLG challenge. *Comput. Speech Lang.* 2020, 59, p. 123–156. doi: 10.1016/J.CSL.2019.06.009. Available at: https://doi.org/10.1016/j.csl.2019.06.009.

Elhadad, M. – Robin, J. SURGE: A Comprehensive Plug-in Syntactic Realization Component for Text Generation. *Computational Linguistics.* 1997, 99, 4.

Eugenio, B. D. – Glass, M. – Trolio, M. J. The DIAG experiments: Natural Language Generation for Intelligent Tutoring Systems. In *Proceedings of the International Natural Language Generation Conference, Harriman*, p. 120–127, New York, USA, 2002. Available at: https://aclanthology.org/W02-2116/.

Fabbri, A. R. – Kryscinski, W. – McCann, B. – Xiong, C. – Socher, R. – Radev, D. R. SummEval: Re-evaluating Summarization Evaluation. *Trans. Assoc. Comput. Linguistics.* 2021, 9, p. 391–409. doi: 10.1162/TACL_A_00373. Available at: https://doi.org/10.1162/tacl_a_00373.

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. – 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. doi: 10.18653/V1/W18-6521. Available at: https://doi.org/10.18653/v1/w18-6521.

Ferreira, T. C. – Lee, C. – Miltenburg, E. – Krahmer, E. Neural Data-to-Text Generation: A Comparison Between Pipeline And End-to-End Architectures. 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 2019*, p. 552–562, Hong Kong, China, 2019. doi: 10.18653/V1/D19-1052. Available at: https://doi.org/10.18653/v1/D19-1052.

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. Available at: https://aclanthology.org/2020.webnlg-1.7/.

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. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, p. 3922–3929, Brussels, Belgium, 2018. doi: 10.18653/V1/D18-1426. Available at: https://doi.org/10.18653/v1/d18-1426.

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. – Presser, S. – Leahy, C. The Pile: An 800GB Dataset of Diverse Text for Language Modeling. *CoRR*. 2021, abs/2101.00027. Available at: https://arxiv.org/abs/2101.00027.

Gardent, C. – Perez-Beltrachini, L. A Statistical, Grammar-Based Approach to Microplanning. *Comput. Linguistics*. 2017, 43, 1, p. 1–30. doi: 10.1162/COLI_A_00273. Available at: https://doi.org/10.1162/COLI_a_00273.

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, ACL 2017 4, Volume 1: Long Papers*, p. 179–188, Vancouver, Canada, 2017a. doi: 10.18653/V1/P17-1017. Available at: https://doi.org/10.18653/v1/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, INLG 2017, Santiago de Compostela*, p. 124–133, Spain, 2017b. doi: 10.18653/V1/W17-3518. Available at: https://doi.org/10.18653/v1/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, INLG 2021*, p. 266–270, Aberdeen, Scotland, UK, 2021. doi: 10.18653/V1/2021.INLG-1.26. Available at: https://doi.org/10.18653/v1/2021.inlg-1.26.

Gatt, A. – Krahmer, E. Survey of the State of the Art in Natural Language Generation: Core Tasks, Applications and Evaluation. *J. Artif. Intell. Res.* 2018, 61, p. 65–170. doi: 10.1613/JAIR.5477. Available at: https://doi.org/10.1613/jair.5477.

Gatt, A. – Reiter, E. SimpleNLG: A Realisation Engine for Practical Applications. In *ENLG 2009 - Proceedings of the 12th European Workshop on Natural Language Generation, March 30-31, 2009*, p. 90–93, Athens, Greece, 2009. Available at: https://aclanthology.org/W09-0613/.

Gehrmann, S. – Dai, F. Z. – Elder, H. – Rush, A. M. End-to-End Content and Plan Selection for Data-to-Text Generation. In *Proceedings of the 11th International Conference on Natural Language Generation*, p. 46–56, Tilburg University, The Netherlands, 2018. doi: 10.18653/V1/W18-6505. Available at: https://doi.org/10.18653/v1/w18-6505.

Gehrmann, S. et al. The GEM Benchmark: Natural Language Generation, its Evaluation and Metrics. *CoRR*. 2021, abs/2102.01672. Available at: https://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. *J. Artif. Intell. Res.* 2023, 77, p. 103–166. doi: 10.1613/JAIR.1.13715. Available at: https://doi.org/10.1613/jair.1.13715.

Gemmell, C. – Dalton, J. Generate, Transform, Answer: Question Specific Tool Synthesis for Tabular Data. *CoRR*. 2023, abs/2303.10138. doi: 10.48550/ARXIV.2303.10138. Available at: https://doi.org/10.48550/arXiv.2303.10138.

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. 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. *CoRR*. 2016, abs/1610.08375. Available at: http://arxiv.org/abs/1610.08375.

Gkatzia, D. – Mahamood, S. A Snapshot of NLG Evaluation Practices 2005 - 2014. In *ENLG 2015 - Proceedings of the 15th European Workshop on Natural Language Generation, 10-11 September 2015, University of Brighton*, p. 57–60, Brighton, UK, 2015. doi: 10.18653/V1/W15-4708. Available at: https://doi.org/10.18653/v1/w15-4708.

Golchin, S. – Surdeanu, M. Time Travel in LLMs: Tracing Data Contamination in Large Language Models. *CoRR*. 2023, abs/2308.08493. doi: 10.48550/ARXIV.2308.08493. Available at: https://doi.org/10.48550/arXiv.2308.08493.

Goldberg, E. – Driedger, N. – Kittredge, R. I. Using Natural-Language Processing to Produce Weather Forecasts. *IEEE Expert*. 1994, 9, 2, p. 45–53. doi: 10.1109/64.294135. Available at: https://doi.org/10.1109/64.294135.

Goodfellow, I. J. – Bengio, Y. – Courville, A. C. *Deep Learning*. Adaptive Computation and Machine Learning. MIT Press, 2016. Available at: http://www.deeplearningbook.org/. ISBN 978-0-262-03561-3.

Grajcar, P. Data-to-Text Generation With Text-Editing Models. *Master's thesis.* 2023. Available at: http://hdl.handle.net/20.500.11956/184140.

Grusky, M. Rogue Scores. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2023*, p. 1914–1934, Toronto, Canada, 2023. doi: 10.18653/V1/2023.ACL-LONG.107. Available at: https://doi.org/10.18653/v1/2023.acl-long.107.

Gu, J. – Lu, Z. – Li, H. – Li, V. O. K. Incorporating Copying Mechanism in Sequence-to-Sequence Learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Volume 1: Long Papers*, Berlin, Germany, 2016. doi: 10.18653/V1/P16-1154. Available at: https://doi.org/10.18653/v1/p16-1154.

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 *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020*, p. 2309–2324, Online, 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. G. – 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, EMNLP 2023*, p. 13310–13325, Singapore, 2023. doi: 10.18653/V1/2023.EMNLP-MAIN.822. Available at: https://doi.org/10.18653/v1/2023.emnlp-main.822.

Haller, A. – Polleres, A. – Dobriy, D. – Ferranti, N. – Méndez, S. J. R. An Analysis of Links in Wikidata. In *The Semantic Web - 19th International Conference, ESWC 2022, Proceedings*, 13261 / *Lecture Notes in Computer Science*, p. 21–38, Hersonissos, Crete, Greece, 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. In *Proceedings of the 14th International Conference on Natural Language Generation, INLG 2021*, p. 1–11, Aberdeen, Scotland, UK, 2021. doi: 10.18653/V1/2021.INLG-1.1. Available at: https://doi.org/10.18653/v1/2021.inlg-1.1.

Harkous, H. – Groves, I. – Saffari, A. Have Your Text and Use It Too! End-to-End Neural Data-to-Text Generation with Semantic Fidelity. In *Proceedings of the 28th International Conference on Computational Linguistics, COLING 2020*, p. 2410–2424, Barcelona, Spain (Online, 2020. doi: 10.18653/V1/2020.COLING-MAIN.218. Available at: https://doi.org/10.18653/v1/2020.coling-main.218.

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, NAACL-HLT 2021*, p. 2545–2568, Online, 2021. doi: 10.18653/V1/2021.NAACL-MAIN.201. Available at: https://doi.org/10.18653/v1/2021.naacl-main.201.

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 *Proceedings of the 22nd Annual Meeting of the Special Interest Group on Discourse and Dialogue, SIGdial 2021*, p. 66–76, Singapore and Online, 2021. doi: 10.18653/V1/2021.SIGDIAL-1.8. Available at: https://doi.org/10.18653/v1/2021.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. *Int. J. Uncertain. Fuzziness Knowl. Based Syst.* 1998, 6, 2, p. 107–116. doi: 10.1142/S0218488598000094. Available at: https://doi.org/10.1142/S0218488598000094.

Hochreiter, S. – Schmidhuber, J. Long Short-Term Memory. *Neural Comput.* 1997, 9, 8, p. 1735–1780. doi: 10.1162/NECO.1997.9.8.1735. Available at: https://doi.org/10.1162/neco.1997.9.8.1735.

Hockenmaier, J. – Steedman, M. CCGbank: A Corpus of CCG Derivations and Dependency Structures Extracted from the Penn Treebank. *Comput. Linguistics*. 2007, 33, 3, p. 355–396. doi: 10.1162/COLI.2007.33.3.355. Available at: https://doi.org/10.1162/coli.2007.33.3.355.

Hoffmann, J. et al. Training Compute-Optimal Large Language Models. *CoRR*. 2022, abs/2203.15556. doi: 10.48550/ARXIV.2203.15556. Available at: https://doi.org/10.48550/arXiv.2203.15556.

Holtzman, A. – Buys, J. – Du, L. – Forbes, M. – Choi, Y. The Curious Case of Neural Text Degeneration. In *8th International Conference on Learning Representations, ICLR 2020*, Addis Ababa, Ethiopia, 2020. Available at: https://openreview.net/forum?id=rygGQyrFvH.

Holtzman, A. – West, P. – Zettlemoyer, L. Generative Models as a Complex Systems Science: How Can We Make Sense of Large Language Model Behavior? *CoRR*. 2023, abs/2308.00189. doi: 10.48550/ARXIV.2308.00189. Available at: https://doi.org/10.48550/arXiv.2308.00189.

Honovich, O. – Aharoni, R. – Herzig, J. – Taitelbaum, H. – Kukliansky, D. – Cohen, V. – Scialom, T. – Szpektor, I. – Hassidim, A. – Matias, Y. TRUE: Re-evaluating Factual Consistency Evaluation. In *Proceedings of the Second DialDoc Workshop on Document-grounded Dialogue and Conversational Question Answering, DialDoc at ACL 2022*, p. 161–175, Dublin, Ireland, 2022. doi: 10.18653/V1/2022.DIALDOC-1.19. Available at: https://doi.org/10.18653/v1/2022.dialdoc-1.19.

Hooker, S. The Hardware Lottery. *Commun. ACM*. 2021, 64, 12, p. 58–65. doi: 10.1145/3467017. Available at: https://doi.org/10.1145/3467017.

Hornik, K. – Stinchcombe, M. B. – White, H. Multilayer Feedforward Networks Are Universal Approximators. *Neural Networks*. 1989, 2, 5, p. 359–366. doi: 10.1016/0893-6080(89)90020-8. Available at: https://doi.org/10.1016/0893-6080(89)90020-8.

Howcroft, D. M. – Belz, A. – Clinciu, M. – Gkatzia, D. – Hasan, S. A. – Mahamood, S. – Mille, S. – Miltenburg, E. – Santhanam, S. – Rieser, V. Twenty Years of Confusion in Human Evaluation: NLG Needs Evaluation Sheets and Standardised Definitions. In *Proceedings of the 13th International Conference on Natural Language Generation, INLG 2020*, p. 169–182, Dublin, Ireland, 2020. doi: 10.18653/V1/2020.INLG-1.23. Available at: https://doi.org/10.18653/v1/2020.inlg-1.23.

Hoyle, A. M. – Marasovic, A. – Smith, N. A. Promoting Graph Awareness in Linearized Graph-to-Text Generation. In *Findings of the Association for Computational Linguistics: ACL/IJCNLP 2021*, ACL/IJCNLP 2021 / *Findings of ACL*, p. 944–956, Online Event, 2021. doi: 10.18653/V1/2021.FINDINGS-ACL.82. Available at: https://doi.org/10.18653/v1/2021.findings-acl.82.

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

Jiang, J. – Zhou, K. – Dong, Z. – Ye, K. – Zhao, X. – Wen, J.-R. StructGPT: A General Framework for Large Language Model to Reason over Structured Data. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, p. 9237–9251, Singapore, December 2023b. doi: 10.18653/v1/2023.emnlp-main.574. Available at: https://aclanthology.org/2023.emnlp-main.574.

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, COLING 2020*, p. 2398–2409, Barcelona, Spain (Online, 2020. doi: 10.18653/V1/2020.COLING-MAIN.217. Available at: https://doi.org/10.18653/v1/2020.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. 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, NAACL-HLT 2018, Volume 1 (Long Papers)*, p. 152–162, New Orleans, Louisiana, USA, 2018. doi: 10.18653/V1/N18-1014. Available at: https://doi.org/10.18653/v1/n18-1014.

Kale, M. – Rastogi, A. Template Guided Text Generation for Task-Oriented Dialogue. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020*, p. 6505–6520, Online, 2020a. doi: 10.18653/V1/2020.EMNLP-MAIN.527. Available at: https://doi.org/10.18653/v1/2020.emnlp-main.527.

Kale, M. – Rastogi, A. Text-to-Text Pre-Training for Data-to-Text Tasks. In *Proceedings of the 13th International Conference on Natural Language Generation, INLG 2020*, p. 97–102, Dublin, Ireland, 2020b. doi: 10.18653/V1/2020.INLG-1.14. Available at: https://doi.org/10.18653/v1/2020.inlg-1.14.

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

Kann, K. – Rothe, S. – Filippova, K. Sentence-Level Fluency Evaluation: References Help, But Can Be Spared! In *Proceedings of the 22nd Conference on Computational Natural Language Learning, CoNLL 2018*, p. 313–323, Brussels, Belgium, 2018. doi: 10.18653/V1/K18-1031. Available at: https://doi.org/10.18653/v1/k18-1031.

Kann, K. – Ebrahimi, A. – Koh, J. 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, ConvAI at ACL 2022*, p. 148–165, Dublin, Ireland, 2022. doi: 10.18653/V1/2022.NLP4CONVAI-1.13. Available at: https://doi.org/10.18653/v1/2022.nlp4convai-1.13.

Kantharaj, S. – Leong, R. T. K. – Lin, X. – Masry, A. – Thakkar, M. – Hoque, E. – Joty, S. R. Chart-to-Text: A Large-Scale Benchmark for Chart Summarization. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2022*, p. 4005–4023, Dublin, Ireland, 2022. doi: 10.18653/V1/2022.ACL-LONG.277. Available at: https://doi.org/10.18653/v1/2022.acl-long.277.

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. *CoRR*. 2020, abs/2001.08361. Available at: https://arxiv.org/abs/2001.08361.

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. Neural Pipeline for Zero-Shot Data-to-Text Generation. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2022*, p. 3914–3932, Dublin, Ireland, 2022. doi: 10.18653/V1/2022.ACL-LONG.271. Available at: https://doi.org/10.18653/v1/2022.acl-long.271.

Kasner, Z. – Dušek, O. Data-to-Text Generation with Iterative Text Editing. In *Proceedings of the 13th International Conference on Natural Language Generation, INLG 2020*, p. 60–67, Dublin, Ireland, 2020a. doi: 10.18653/V1/2020.INLG-1.9. Available at: https://doi.org/10.18653/v1/2020.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 *Proceedings of the 3rd International Workshop on Natural Language Generation from the Semantic Web (WebNLG+)*, p. 171–176, Dublin, Ireland (Virtual), 12 2020b. Available at: https://aclanthology.org/2020.webnlg-1.20.

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, INLG 2021*, p. 259–265, Aberdeen, Scotland, UK, 2021. doi: 10.18653/V1/2021.INLG-1.25. Available at: https://doi.org/10.18653/v1/2021.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: System Demonstrations, ACL 2023*, p. 444–455, Toronto, Canada, 2023a. doi: 10.18653/V1/2023.ACL-DEMO.42. Available at: https://doi.org/10.18653/v1/2023.acl-demo.42.

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, EACL 2023, Dubrovnik*, p. 2390–2407, Croatia, 2023b. doi: 10.18653/V1/2023.EACL-MAIN.176. Available at: https://doi.org/10.18653/v1/2023.eacl-main.176.

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. In *Findings of the Association for Computational Linguistics: ACL/IJCNLP 2021*, ACL/IJCNLP 2021 / *Findings of ACL*, p. 2526–2538, Online Event, 2021. doi: 10.18653/V1/2021.FINDINGS-ACL.223. Available at: https://doi.org/10.18653/v1/2021.findings-acl.223.

Kedzie, C. – McKeown, K. R. 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, INLG 2019*, p. 584–593, Tokyo, Japan, 2019. doi: 10.18653/V1/W19-8672. Available at: https://aclanthology.org/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.

Kiela, D. et al. Dynabench: Rethinking Benchmarking in NLP. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2021*, p. 4110–4124, Online, 2021. doi: 10.18653/V1/2021.NAACL-MAIN.324. Available at: https://doi.org/10.18653/v1/2021.naacl-main.324.

Kingma, D. P. – Ba, J. Adam: A Method for Stochastic Optimization. In *3rd International Conference on Learning Representations, ICLR 2015Proceedings*, San Diego, CA, USA, 2015. Available at: http://arxiv.org/abs/1412.6980.

Kirkpatrick, J. – Pascanu, R. – Rabinowitz, N. C. – Veness, J. – Desjardins, G. – Rusu, A. A. – Milan, K. – Quan, J. – Ramalho, T. – Grabska-Barwinska, A. – Hassabis, D. – Clopath, C. – Kumaran, D. – Hadsell, R. Overcoming Catastrophic Forgetting in Neural Networks. *CoRR*. 2016, abs/1612.00796. Available at: http://arxiv.org/abs/1612.00796.

Kocmi, T. – Federmann, C. GEMBA-MQM: Detecting Translation Quality Error Spans With GPT-4. In *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 *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. In *Proceedings of the Sixth Conference on Machine Translation, WMT at EMNLP 2021*, p. 478–494, Online Event, 2021. Available at: https://aclanthology.org/2021.wmt-1.57.

Koncel-Kedziorski, R. – Bekal, D. – Luan, Y. – Lapata, M. – Hajishirzi, H. Text Generation from Knowledge Graphs with Graph Transformers. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, Volume 1 (Long and Short Papers)*, p. 2284–2293, USA, 2019. doi: 10.18653/V1/N19-1238. Available at: https://doi.org/10.18653/v1/n19-1238.

Konstas, I. – Lapata, M. Concept-to-text Generation via Discriminative Reranking. In *The 50th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference, July 8-14, 2012Volume 1: Long Papers*, p. 369–378, Jeju Island, Korea, 2012. Available at: https://aclanthology.org/P12-1039/.

Koo, R. – Lee, M. – Raheja, V. – Park, J. I. – Kim, Z. M. – Kang, D. Benchmarking Cognitive Biases in Large Language Models as Evaluators. *CoRR*. 2023, abs/2309.17012. doi: 10.48550/ARXIV.2309.17012. Available at: https://doi.org/10.48550/arXiv.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. 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. Subword Regularization: Improving Neural Network Translation Models with Multiple Subword Candidates. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Volume 1: Long Papers*, p. 66–75, Melbourne, Australia, 2018. doi: 10.18653/V1/P18-1007. Available at: https://aclanthology.org/P18-1007/.

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, EMNLP 2018: System Demonstrations*, p. 66–71, Brussels, Belgium, 2018. doi: 10.18653/V1/D18-2012. Available at: https://doi.org/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 *6th Applied Natural Language Processing Conference, ANLP 2000*, p. 170–177, Seattle, Washington, USA, 2000. Available at: https://aclanthology.org/A00-2023/.

Lapata, M. Probabilistic Text Structuring: Experiments with Sentence Ordering. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics, 7-12 July 2003, Sapporo Convention Center, Sapporo*, p. 545–552, 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 *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016*, p. 1203–1213, Austin, Texas, USA, 2016. doi: 10.18653/V1/D16-1128. Available at: https://doi.org/10.18653/v1/d16-1128.

LeCun, Y. – Bengio, Y. – Hinton, G. E. Deep Learning. *Nat.* 2015, 521, 7553, p. 436–444. doi: 10.1038/NATURE14539. Available at: https://doi.org/10.1038/nature14539.

Lee, C. – Cheng, H. – Ostendorf, M. Dialogue State Tracking with a Language Model using Schema-Driven Prompting. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event /*, p. 4937–4949, Punta Cana, Dominican Republic, 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 *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, Louisiana, 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. – Fung, P. – Shoeybi, M. – Catanzaro, B. Factuality Enhanced Language Models for Open-Ended Text Generation. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022*, New Orleans, LA, USA, 2022. Available at: http://papers.nips.cc/paper_files/paper/2022/hash/df438caa36714f69277daa92d608dd63-Abstract-Conference.html.

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, INLG 2017, Santiago de Compostela*, p. 188–197, Spain, 2017. doi: 10.18653/V1/W17-3528. Available at: https://doi.org/10.18653/v1/w17-3528.

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. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020*, p. 7871–7880, Online, 2020. doi: 10.18653/V1/2020.ACL-MAIN.703. Available at: https://doi.org/10.18653/v1/2020.acl-main.703.

Lhoest, Q. et al. Datasets: A Community Library for Natural Language Processing. In *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, B. A Diversity-Promoting Objective Function for Neural Conversation Models. In *NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, p. 110–119, San Diego California, USA, 2016. doi: 10.18653/V1/N16-1014. Available at: https://doi.org/10.18653/v1/n16-1014.

Li, S. – Han, C. – Yu, P. – Edwards, C. – Li, M. – Wang, X. – Fung, Y. R. – Yu, C. – Tetreault, J. R. – Hovy, E. H. – Ji, H. Defining a New NLP Playground. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, p. 11932–11951, Singapore, 2023. doi: 10.18653/V1/2023.FINDINGS-EMNLP.799. Available at: https://doi.org/10.18653/v1/2023.findings-emnlp.799.

Li, W. – Wu, W. – Chen, M. – Liu, J. – Xiao, X. – Wu, H. Faithfulness in Natural Language Generation: A Systematic Survey of Analysis, Evaluation and Optimization Methods. *CoRR*. 2022, abs/2203.05227. doi: 10.48550/ARXIV.2203.05227. Available at: https://doi.org/10.48550/arXiv.2203.05227.

Liang, P. – Jordan, M. I. – Klein, D. Learning Semantic Correspondences with Less Supervision. In *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. 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 Trans. Knowl. Data Eng.* 2024, 36, 4, p. 1431–1449. doi: 10.1109/TKDE.2023.3304385. Available at: https://doi.org/10.1109/TKDE.2023.3304385.

Liu, X. – He, P. – Chen, W. – Gao, J. Multi-Task Deep Neural Networks for Natural Language Understanding. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Volume 1: Long Papers*, p. 4487–4496, Florence, Italy, 2019a. doi: 10.18653/V1/P19-1441. Available at: https://doi.org/10.18653/v1/p19-1441.

Liu, Y. – Medlar, A. – Glowacka, D. Can Language Models Identify Wikipedia Articles with Readability and Style Issues? In *ICTIR '21: The 2021 ACM SIGIR International Conference on the Theory of Information Retrieval*, p. 113–117, Virtual Event, Canada, 2021. doi: 10.1145/3471158.3472234. Available at: https://doi.org/10.1145/3471158.3472234.

Liu, Y. – Iter, D. – Xu, Y. – Wang, S. – Xu, R. – Zhu, C. G-Eval: NLG Evaluation using GPT-4 with Better Human Alignment. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023*, p. 2511–2522, Singapore, 2023. doi: 10.18653/V1/2023.EMNLP-MAIN.153. Available at: https://doi.org/10.18653/v1/2023.emnlp-main.153.

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. *CoRR*. 2019b, abs/1907.11692. 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. *Trans. Assoc. Comput. Linguistics*. 2020, 8, p. 726–742. doi: 10.1162/TACL_A_00343. Available at: https://doi.org/10.1162/tacl_a_00343.

Lorandi, M. – Belz, A. Data-to-Text Generation for Severely Under-Resourced Languages With GPT-3.5: A Bit of Help Needed From Google Translate (WebNLG 2023). In *Proceedings of the Workshop on Multimodal, Multilingual Natural Language Generation and Multilingual WebNLG Challenge (MM-NLG 2023)*, p. 80–86, 2023. Available at: https://aclanthology.org/2023.mmnlg-1.9/.

Lorandi, M. – Belz, A. High-quality Data-to-Text Generation for Severely Under-Resourced Languages with Out-of-the-box Large Language Models. In *Findings of the Association for Computational Linguistics: EACL 2024*, p. 1451–1461, St. Julian's, Malta, 2024. Available at: https://aclanthology.org/2024.findings-eacl.98.

Loshchilov, I. – Hutter, F. Fixing Weight Decay Regularization in Adam. *CoRR*. 2017, abs/1711.05101. Available at: http://arxiv.org/abs/1711.05101.

Luong, T. – Pham, H. – Manning, C. D. Effective Approaches to Attention-based Neural Machine Translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015*, p. 1412–1421, Lisbon, Portugal, 2015. doi: 10.18653/V1/D15-1166. Available at: https://doi.org/10.18653/v1/d15-1166.

Mairesse, F. – Gasic, M. – Jurčíček, F. – Keizer, S. – Thomson, B. – Yu, K. – Young, S. J. Phrase-Based Statistical Language Generation using Graphical Models and Active Learning. In *ACL 2010, Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, July 11-16, 2010*, p. 1552–1561, Uppsala, Sweden, 2010. Available at: https://aclanthology.org/P10-1157/.

Mallinson, J. – Severyn, A. – Malmi, E. – Garrido, G. FELIX: Flexible Text Editing Through Tagging and Insertion. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, EMNLP 2020 / *Findings of ACL*, p. 1244–1255, Online Event, 2020. doi: 10.18653/V1/2020.FINDINGS-EMNLP.111. Available at: https://doi.org/10.18653/v1/2020.findings-emnlp.111.

Malmi, E. – Krause, S. – Rothe, S. – Mirylenka, D. – Severyn, A. Encode, Tag, Realize: High-Precision Text Editing. 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 2019*, p. 5053–5064, Hong Kong, China, 2019. doi: 10.18653/V1/D19-1510. Available at: https://doi.org/10.18653/v1/D19-1510.

Malmi, E. – Dong, Y. – Mallinson, J. – Chuklin, A. – Adámek, J. – Mirylenka, D. – Stahlberg, F. – Krause, S. – Kumar, S. – Severyn, A. Text Generation with Text-Editing Models. *CoRR*. 2022, abs/2206.07043. doi: 10.48550/ARXIV.2206.07043. Available at: https://doi.org/10.48550/arXiv.2206.07043.

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. In *Proceedings of the 11th International Conference on Natural Language Generation*, p. 1–9, Tilburg University, The Netherlands, 2018. doi: 10.18653/V1/W18-6501. Available at: https://doi.org/10.18653/v1/w18-6501.

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. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020*, p. 4984–4997, Online, 2020. doi: 10.18653/V1/2020.ACL-MAIN.448. Available at: https://doi.org/10.18653/v1/2020.acl-main.448.

Matthiessen, C. Lexico (Grammatical) Choice in Text Generation. 1991, p. 249–292.

Maynez, J. – Narayan, S. – Bohnet, B. – McDonald, R. T. On Faithfulness and Factuality in Abstractive Summarization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020*, p. 1906–1919, Online, 2020. doi: 10.18653/V1/2020.ACL-MAIN.173. Available at: https://doi.org/10.18653/v1/2020.acl-main.173.

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 *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Volume 1: Long Papers*, p. 3428–3448, Florence, Italy, 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. – Strubell, E. Improving Compositional Generalization with Self-Training for Data-to-Text Generation. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2022*, p. 4205–4219, Dublin, Ireland, 2022. doi: 10.18653/V1/2022.ACL-LONG.289. Available at: https://doi.org/10.18653/v1/2022.acl-long.289.

Mei, H. – Bansal, M. – Walter, M. R. What to Talk About and How? Selective Generation using LSTMs with Coarse-to-Fine Alignment. In *NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, p. 720–730, San Diego California, USA, 2016. doi: 10.18653/V1/N16-1086. Available at: https://doi.org/10.18653/v1/n16-1086.

MEISTER, C. – WIHER, G. – COTTERELL, R. On Decoding Strategies for Neural Text Generators. *Trans. Assoc. Comput. Linguistics*. 2022, 10, p. 997–1012. Available at: https://transacl.org/ojs/index.php/tacl/article/view/3807.

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. – BOWMAN, S. R. 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), ACL 2023*, p. 16334–16368, Toronto, Canada, 2023. doi: 10.18653/V1/2023.ACL-LONG.903. Available at: https://doi.org/10.18653/v1/2023.acl-long.903.

MIKOLOV, T. – SUTSKEVER, I. – CHEN, K. – CORRADO, G. S. – DEAN, J. Distributed Representations of Words and Phrases and their Compositionality. In *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe*, p. 3111–3119, Nevada, United States, 2013. Available at: https://proceedings.neurips.cc/paper/2013/hash/9aa42b31882ec039965f3c4923ce901b-Abstract.html.

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, INLG 2019*, p. 473–483, Tokyo, Japan, 2019. doi: 10.18653/V1/W19-8659. Available at: https://aclanthology.org/W19-8659/.

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 *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks 1, NeurIPS Datasets and Benchmarks 2021*, 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, INLG 2023*, p. 455–466, Prague, Czechia, 2023. doi: 10.18653/V1/2023.INLG-MAIN.36. Available at: https://doi.org/10.18653/v1/2023.inlg-main.36.

MIN, B. – ROSS, H. – SULEM, E. – VEYSEH, A. P. B. – NGUYEN, T. H. – SAINZ, O. – AGIRRE, E. – HEINTZ, I. – ROTH, D. Recent Advances in Natural Language Processing via Large Pre-trained Language Models: A Survey. *ACM Comput. Surv.* 2024, 56, 2, p. 30:1–30:40. doi: 10.1145/3605943. Available at: https://doi.org/10.1145/3605943.

MITTAL, V. O. – MOORE, J. D. – CARENINI, G. – ROTH, S. F. Describing Complex Charts in Natural Language: A Caption Generation System. *Comput. Linguistics*. 1998, 24, 3, p. 431–467.

Moosavi, N. S. – Rücklé, A. – Roth, D. – Gurevych, I. Learning to Reason for Text Generation from Scientific Tables. *CoRR*. 2021, abs/2104.08296. Available at: https://arxiv.org/abs/2104.08296.

Moryossef, A. – Goldberg, Y. – Dagan, I. Improving Quality and Efficiency in Plan-based Neural Data-to-text Generation. In *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 *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, Volume 1 (Long and Short Papers)*, p. 2267–2277, USA, 2019b. doi: 10.18653/V1/N19-1236. Available at: https://doi.org/10.18653/v1/n19-1236.

Munkhdalai, T. – Faruqui, M. – Gopal, S. Leave No Context Behind: Efficient Infinite Context Transformers with Infini-attention. *CoRR*. 2024, abs/2404.07143. doi: 10.48550/ARXIV.2404.07143. Available at: https://doi.org/10.48550/arXiv.2404.07143.

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, ACL 2017 4, Volume 1: Long Papers*, p. 1374–1384, Vancouver, Canada, 2017. doi: 10.18653/V1/P17-1126. Available at: https://doi.org/10.18653/v1/P17-1126.

Nair, V. – Hinton, G. E. Rectified Linear Units Improve Restricted Boltzmann Machines. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10), June 21-24, 2010*, p. 807–814, Haifa, Israel, 2010. Available at: https://icml.cc/Conferences/2010/papers/432.pdf.

Nan, L. et al. 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, NAACL-HLT 2021*, p. 432–447, Online, 2021. doi: 10.18653/V1/2021.NAACL-MAIN.37. Available at: https://doi.org/10.18653/v1/2021.naacl-main.37.

Narayan, S. – Gardent, C. – Cohen, S. B. – Shimorina, A. Split and Rephrase. In *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.

Naveed, H. – Khan, A. U. – Qiu, S. – Saqib, M. – Anwar, S. – Usman, M. – Barnes, N. – Mian, A. A Comprehensive Overview of Large Language Models. *CoRR*. 2023, abs/2307.06435. doi: 10.48550/ARXIV.2307.06435. Available at: https://doi.org/10.48550/arXiv.2307.06435.

Neeraja, J. – Gupta, V. – Srikumar, V. Incorporating External Knowledge to Enhance Tabular Reasoning. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2021*, p. 2799–2809, Online, 2021. doi: 10.18653/V1/2021.NAACL-MAIN.224. Available at: https://doi.org/10.18653/v1/2021.naacl-main.224.

Nekvinda, T. – Dušek, O. Shades of BLEU, Flavours of Success: The Case of MultiWOZ. *CoRR*. 2021, abs/2106.05555. Available at: https://arxiv.org/abs/2106.05555.

Nie, F. – Yao, J. – Wang, J. – Pan, R. – Lin, C. A Simple Recipe towards Reducing Hallucination in Neural Surface Realisation. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Volume 1: Long Papers*, p. 2673–2679, Florence, Italy, 2019. doi: 10.18653/V1/P19-1256. Available at: https://doi.org/10.18653/v1/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.

Novikova, J. – Dušek, O. – Curry, A. C. – Rieser, V. Why We Need New Evaluation Metrics for NLG. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017*, p. 2241–2252, Copenhagen, Denmark, 2017. doi: 10.18653/V1/D17-1238. Available at: https://doi.org/10.18653/v1/d17-1238.

Obeid, J. – Hoque, E. Chart-to-Text: Generating Natural Language Descriptions for Charts by Adapting the Transformer Model. In *Proceedings of the 13th International Conference on Natural Language Generation, INLG 2020*, p. 138–147, Dublin, Ireland, 2020. doi: 10.18653/V1/2020.INLG-1.20. Available at: https://doi.org/10.18653/v1/2020.inlg-1.20.

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: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Demonstrations*, p. 48–53, 2019. doi: 10.18653/V1/N19-4009. Available at: https://doi.org/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. BLEU: A Method for Automatic Evaluation of Machine Translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics, July 6-12, 2002*, p. 311–318, Philadelphia, PA, USA, 2002. doi: 10.3115/1073083.1073135. Available at: https://aclanthology.org/P02-1040/.

Parikh, A. P. – Wang, X. – Gehrmann, S. – Faruqui, M. – Dhingra, B. – Yang, D. – Das, D. ToTTo: A Controlled Table-To-Text Generation Dataset. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020*, p. 1173–1186, Online, 2020. doi: 10.18653/V1/2020.EMNLP-MAIN.89. Available at: https://doi.org/10.18653/v1/2020.emnlp-main.89.

Pascanu, R. – Mikolov, T. – Bengio, Y. On the Difficulty of Training Recurrent Neural Networks. In *Proceedings of the 30th International Conference on Machine Learning, ICML 2013*, 28 / *JMLR Workshop and Conference Proceedings*, p. 1310–1318, Atlanta, GA, USA, 2013. Available at: http://proceedings.mlr.press/v28/pascanu13.html.

Paszke, A. et al. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *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. *CoRR*. 2022, abs/2205.10938. doi: 10.48550/ARXIV.2205.10938. Available at: https://doi.org/10.48550/arXiv.2205.10938.

Peters, M. E. – Neumann, M. – Iyyer, M. – Gardner, M. – Clark, C. – Lee, K. – Zettlemoyer, L. Deep Contextualized Word Representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018, Volume 1 (Long Papers)*, p. 2227–2237, New Orleans, Louisiana, USA, 2018. doi: 10.18653/V1/N18-1202. Available at: https://doi.org/10.18653/v1/n18-1202.

Petroni, F. – Rocktäschel, T. – Riedel, S. – Lewis, P. S. H. – Bakhtin, A. – Wu, Y. – Miller, A. H. 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 2019*, p. 2463–2473, Hong Kong, China, 2019. doi: 10.18653/V1/D19-1250. Available at: https://doi.org/10.18653/v1/D19-1250.

Ponnamperuma, K. – Siddharthan, A. – Zeng, C. – Mellish, C. – Wal, R. Tag2Blog: Narrative Generation from Satellite Tag Data. In *51st Annual Meeting of the Association for Computational Linguistics, ACL 2013, Proceedings of the Conference System Demonstrations, 4-9 August 2013*, p. 169–174, Sofia, Bulgaria, 2013. Available at: https://aclanthology.org/P13-4029/.

Popovic, M. chrF: Character N-Gram F-Score for Automatic MT Evaluation. In *Proceedings of the Tenth Workshop on Statistical Machine Translation, WMT at EMNLP 2015, 17-18 September 2015*, p. 392–395, Lisbon, Portugal, 2015. doi: 10.18653/V1/W15-3049. Available at: https://doi.org/10.18653/v1/w15-3049.

Popovic, M. chrF++: Words Helping Character N-Grams. In *Proceedings of the Second Conference on Machine Translation, WMT 2017*, p. 612–618, Copenhagen, Denmark, 2017. doi: 10.18653/V1/W17-4770. Available at: https://doi.org/10.18653/v1/w17-4770.

Portet, F. – Reiter, E. – Gatt, A. – Hunter, J. – Sripada, S. – Freer, Y. – Sykes, C. Automatic Generation of Textual Summaries From Neonatal Intensive Care Data. *Artif. Intell.* 2009, 173, 7-8, p. 789–816. doi: 10.1016/J.ARTINT.2008.12.002. Available at: https://doi.org/10.1016/j.artint.2008.12.002.

Post, M. A Call for Clarity in Reporting BLEU Scores. In *Proceedings of the Third Conference on Machine Translation: Research Papers, WMT 2018*, p. 186–191, Belgium, Brussels, 2018. doi: 10.18653/V1/W18-6319. Available at: https://doi.org/10.18653/v1/w18-6319.

Puduppully, R. – Lapata, M. Data-to-text Generation with Macro Planning. *Trans. Assoc. Comput. Linguistics.* 2021, 9, p. 510–527. doi: 10.1162/TACL_A_00381. Available at: https://doi.org/10.1162/tacl_a_00381.

Puduppully, R. – Dong, L. – Lapata, M. Data-to-text Generation with Entity Modeling. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Volume 1: Long Papers*, p. 2023–2035, Florence, Italy, 2019. doi: 10.18653/V1/P19-1195. Available at: https://doi.org/10.18653/v1/p19-1195.

Qian, J. – Wang, H. – Li, Z. – Li, S. – Yan, X. Limitations of Language Models in Arithmetic and Symbolic Induction. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2023*, p. 9285–9298, Toronto, Canada, 2023. doi: 10.18653/V1/2023.ACL-LONG.516. Available at: https://doi.org/10.18653/v1/2023.acl-long.516.

Radford, A. – Narasimhan, K. – Salimans, T. – Sutskever, I. – others. Improving Language Understanding by Generative Pre-Training. *OpenAI Blog.* 2018. Available at: https://cdn.openai.com/research-covers/language-unsupervised/language_understanding_paper.pdf.

Radford, A. – Wu, J. – Child, R. – Luan, D. – Amodei, D. – Sutskever, I. Language Models Are Unsupervised Multitask Learners. *OpenAI Blog*. 2019, p. 24. Available at: https://cdn.openai.com/better-language-models/language_models_are_unsupervised_multitask_learners.pdf.

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. *J. Mach. Learn. Res.* 2020, 21, p. 140:1–140:67. Available at: http://jmlr.org/papers/v21/20-074.html.

Raji, I. D. – Denton, E. – Bender, E. M. – Hanna, A. – Paullada, A. AI and the Everything in the Whole Wide World Benchmark. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks 1, NeurIPS Datasets and Benchmarks 2021*, 2021. Available at: https://datasets-benchmarks-proceedings.neurips.cc/paper/2021/hash/084b6fbb10729ed4da8c3d3f5a3ae7c9-Abstract-round2.html.

Rastogi, A. – Zang, X. – Sunkara, S. – Gupta, R. – Khaitan, P. Towards Scalable Multi-Domain Conversational Agents: The Schema-Guided Dialogue Dataset. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020*, p. 8689–8696, New York, NY, USA, 2020. doi: 10.1609/AAAI.V34I05.6394. Available at: https://doi.org/10.1609/aaai.v34i05.6394.

Ratnaparkhi, A. Trainable Methods for Surface Natural Language Generation. In *6th Applied Natural Language Processing Conference, ANLP 2000*, p. 194–201, Seattle, Washington, USA, 2000. Available at: https://aclanthology.org/A00-2026/.

Rebuffel, C. – Scialom, T. – Soulier, L. – Piwowarski, B. – Lamprier, S. – Staiano, J. – Scoutheeten, G. – Gallinari, P. Data-QuestEval: A Referenceless Metric for Data-to-Text Semantic Evaluation. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event /*, p. 8029–8036, Punta Cana, Dominican Republic, 2021. doi: 10.18653/V1/2021.EMNLP-MAIN.633. Available at: https://doi.org/10.18653/v1/2021.emnlp-main.633.

Rebuffel, C. – Roberti, M. – Soulier, L. – Scoutheeten, G. – Cancelliere, R. – Gallinari, P. Controlling Hallucinations at Word Level in Data-to-Text Generation. *Data Min. Knowl. Discov.* 2022, 36, 1, p. 318–354. doi: 10.1007/S10618-021-00801-4. Available at: https://doi.org/10.1007/s10618-021-00801-4.

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. doi: 10.18653/v1/D19-1410. Available at: https://www.aclweb.org/anthology/D19-1410.

Reiter, E. Building Natural-Language Generation Systems. *CoRR*. 1996, cmp-lg/9605002. Available at: http://arxiv.org/abs/cmp-lg/9605002.

Reiter, E. NLG vs Templates: Levels of Sophistication in Generating Text. https://ehudreiter.com/2016/12/18/nlg-vs-templates, 2016. Accessed on June 15, 2024.

Reiter, E. A Structured Review of the Validity of BLEU. *Comput. Linguistics*. 2018, 44, 3. doi: 10.1162/COLI_A_00322. Available at: https://doi.org/10.1162/coli_a_00322.

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. How to do an NLG Evaluation: Task-Based (Extrinsic) Performance in Real-World Context. https://ehudreiter.com/2017/04/27/task-based-real-world-nlg-eval/, 2017. Accessed on June 14, 2024.

Reiter, E. We Should Evaluate Real-World Impact! https://ehudreiter.com/2023/11/13/evaluate-real-world-impact/, 2023. Accessed on January 11, 2024.

Reiter, E. An Architecture for Data-to-Text Systems. In *Proceedings of the Eleventh European Workshop on Natural Language Generation, ENLG 2007*, Schloss Dagstuhl, Germany, 2007. Available at: https://aclanthology.org/W07-2315/.

Reiter, E. – Dale, R. Building Applied Natural Language Generation Systems. *Nat. Lang. Eng.* 1997, 3, 1, p. 57–87. doi: 10.1017/S1351324997001502. Available at: https://doi.org/10.1017/S1351324997001502.

Reiter, E. – Thomson, C. Shared Task on Evaluating Accuracy. In *Proceedings of the 13th International Conference on Natural Language Generation, INLG 2020*, p. 227–231, Dublin, Ireland, 2020. doi: 10.18653/V1/2020.INLG-1.28. Available at: https://doi.org/10.18653/v1/2020.inlg-1.28.

Reiter, E. – Robertson, R. – Osman, L. Lessons From a Failure: Generating Tailored Smoking Cessation Letters. *Artif. Intell.* 2003, 144, 1-2, p. 41–58. doi: 10.1016/S0004-3702(02)00370-3. Available at: https://doi.org/10.1016/S0004-3702(02)00370-3.

Rogers, A. Closed AI Models Make Bad Baselines. https://hackingsemantics.xyz/2023/closed-baselines/, 2023. Accessed on January 11, 2024.

Rogers, A. – Luccioni, S. Position: Key Claims in LLM Research Have a Long Tail of Footnotes. In *Forty-first International Conference on Machine Learning*, 2024. Available at: https://openreview.net/forum?id=M2cwkGleRL.

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), ACL 2022*, p. 5645–5657, Dublin, Ireland, 2022. doi: 10.18653/V1/2022.ACL-LONG.387. Available at: https://doi.org/10.18653/v1/2022.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. – Bansal, M. – Pasunuru, R. – Celikyilmaz, A. MURMUR: Modular Multi-Step Reasoning for Semi-Structured Data-to-Text Generation. In *Findings of the Association for Computational Linguistics: ACL 2023*, p. 11069–11090, Toronto, Canada, 2023. doi: 10.18653/V1/2023.FINDINGS-ACL.704. Available at: https://doi.org/10.18653/v1/2023.findings-acl.704.

Salehinejad, H. – Baarbe, J. – Sankar, S. – Barfett, J. – Colak, E. – Valaee, S. Recent Advances in Recurrent Neural Networks. *CoRR*. 2018, abs/1801.01078. Available at: http://arxiv.org/abs/1801.01078.

Sanh, V. et al. Multitask Prompted Training Enables Zero-Shot Task Generalization. In *The Tenth International Conference on Learning Representations, ICLR 2022*, Virtual Event, 2022. Available at: https://openreview.net/forum?id=9Vrb9D0WI4.

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.

Schopf, T. – Arabi, K. – Matthes, F. Exploring the Landscape of Natural Language Processing Research. In *Proceedings of the 14th International Conference on Recent Advances in Natural Language Processing, RANLP 2023*, p. 1034–1045, Varna, Bulgaria, 2023. Available at: https://aclanthology.org/2023.ranlp-1.111.

Scialom, T. – Dray, P. – Lamprier, S. – Piwowarski, B. – Staiano, J. – Wang, A. – Gallinari, P. QuestEval: Summarization Asks for Fact-based Evaluation. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event /*, p. 6594–6604, Punta Cana, Dominican Republic, 2021. doi: 10.18653/V1/2021.EMNLP-MAIN.529. Available at: https://doi.org/10.18653/v1/2021.emnlp-main.529.

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. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017 4, Volume 1: Long Papers*, p. 1073–1083, Vancouver, Canada, 2017. doi: 10.18653/V1/P17-1099. Available at: `https://doi.org/10.18653/v1/P17-1099`.

SELLAM, T. – DAS, D. – PARIKH, A. P. BLEURT: Learning Robust Metrics for Text Generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020*, p. 7881–7892, Online, 2020. doi: 10.18653/V1/2020.ACL-MAIN.704. Available at: `https://doi.org/10.18653/v1/2020.acl-main.704`.

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, ACL 2016, August 7-12, 2016, Volume 1: Long Papers*, Berlin, Germany, 2016. doi: 10.18653/V1/P16-1162. Available at: `https://doi.org/10.18653/v1/p16-1162`.

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 *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 *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, China, 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. *CoRR*. 2022, abs/2207.12571. doi: 10.48550/ARXIV.2207.12571. Available at: `https://doi.org/10.48550/arXiv.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 *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. Human vs Automatic Metrics: on the Importance of Correlation Design. *CoRR*. 2018, abs/1805.11474. Available at: `http://arxiv.org/abs/1805.11474`.

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. doi: 10.18653/V1/W18-6543. Available at: https://doi.org/10.18653/v1/w18-6543.

Shimorina, A. – Gardent, C. – Narayan, S. – Perez-Beltrachini, L. WebNLG Challenge: Human Evaluation Results. 2019, p. 16. Available at: https://synalp.gitlabpages.inria.fr/webnlg-challenge/files/human-eval-outline-v2.pdf.

Siddharthan, A. – Green, M. – Deemter, K. – Mellish, C. – 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, 30 May 2012 - 1 June 2012, Starved Rock State Park, Utica, IL,*, p. 120–124, USA, 2012. Available at: https://aclanthology.org/W12-1520/.

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 *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023*, p. 8776–8788, Singapore, 2023. doi: 10.18653/V1/2023.EMNLP-MAIN.543. Available at: https://doi.org/10.18653/v1/2023.emnlp-main.543.

Steedman, M. *The Syntactic Process*. Language, Speech, and Communication. MIT Press, 2004. ISBN 978-0-262-69268-7.

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 Trans. Pattern Anal. Mach. Intell.* 2023, 45, 1, p. 539–559. doi: 10.1109/TPAMI.2022.3148210. Available at: https://doi.org/10.1109/TPAMI.2022.3148210.

Stureborg, R. – Alikaniotis, D. – Suhara, Y. Large Language Models are Inconsistent and Biased Evaluators. *CoRR*. 2024, abs/2405.01724. doi: 10.48550/ARXIV.2405.01724. Available at: https://doi.org/10.48550/arXiv.2405.01724.

Su, Y. – Meng, Z. – Baker, S. – Collier, N. Few-Shot Table-to-Text Generation with Prototype Memory. In *Findings of the Association for Computational Linguistics: EMNLP 2021, Virtual Event /*, p. 910–917, Punta Cana, Dominican Republic, 2021a. doi: 10.18653/V1/2021.FINDINGS-EMNLP.77. Available at: https://doi.org/10.18653/v1/2021.findings-emnlp.77.

Su, Y. – Vandyke, D. – Wang, S. – Fang, Y. – Collier, N. Plan-then-Generate: Controlled Data-to-Text Generation via Planning. In *Findings of the Association for Computational Linguistics: EMNLP 2021, Virtual Event /*, p. 895–909, 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.

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, ACL/IJCNLP 2021, (Volume 1: Long Papers)*, p. 1451–1465, Virtual Event, 2021. doi: 10.18653/V1/2021.ACL-LONG.115. Available at: https://doi.org/10.18653/v1/2021.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 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014*, p. 3104–3112, Montreal, Quebec, Canada, 2014. Available at: https://proceedings.neurips.cc/paper/2014/hash/a14ac55a4f27472c5d894ec1c3c743d2-Abstract.html.

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 *The Semantic Web - 17th International Conference, ESWC 2020, Proceedings*, 12123 / *Lecture Notes in Computer Science*, p. 583–596, Heraklion, Crete, Greece, 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.

Team, G. – Anil, R. – Borgeaud, S. – Wu, Y. – Alayrac, J.-B. – Yu, J. – Soricut, R. – Schalkwyk, J. – Dai, A. M. – Hauth, A. – others. Gemini: A Family of Highly Capable Multimodal Models. *arXiv preprint arXiv:2312.11805*. 2023. Available at: https://doi.org/10.48550/arXiv.2312.11805.

Thomson, C. – Reiter, E. Generation Challenges: Results of the Accuracy Evaluation Shared Task. In *Proceedings of the 14th International Conference on Natural Language Generation, INLG 2021*, p. 240–248, Aberdeen, Scotland, UK, 2021. doi: 10.18653/V1/2021.INLG-1.23. Available at: https://doi.org/10.18653/v1/2021.inlg-1.23.

Thomson, C. – Reiter, E. A Gold Standard Methodology for Evaluating Accuracy in Data-To-Text Systems. In *Proceedings of the 13th International Conference on Natural Language Generation, INLG 2020*, p. 158–168, Dublin, Ireland, 2020. doi: 10.18653/V1/2020.INLG-1.22. Available at: https://doi.org/10.18653/v1/2020.inlg-1.22.

Thomson, C. – Reiter, E. – Sripada, S. SportSett:Basketball - A Robust and Maintainable Dataset for Natural Language Generation. In *Proceedings of the Workshop on Intelligent Information Processing and Natural Language Generation*, p. 32–40, Santiago de Compostela, Spain, September 2020. Association for Computational Lingustics. Available at: https://aclanthology.org/2020.intellang-1.4.

Thomson, C. – Reiter, E. – Sundararajan, B. Evaluating Factual Accuracy in Complex Data-to-Text. *Comput. Speech Lang.* 2023, 80, p. 101482. doi: 10.1016/J.CSL.2023.101482. Available at: https://doi.org/10.1016/j.csl.2023.101482.

Tian, R. – Narayan, S. – Sellam, T. – Parikh, A. P. Sticking to the Facts: Confident Decoding for Faithful Data-to-Text Generation. *CoRR*. 2019, abs/1910.08684. Available at: http://arxiv.org/abs/1910.08684.

TogetherAI. Preparing for the Era of 32K Context: Early Learnings and Explorations. https://www.together.ai/blog/llama-2-7b-32k, 2023. Accessed on January 2, 2024.

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. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020*, p. 9057–9064, New York, NY, USA, 2020. doi: 10.1609/AAAI.V34I05.6439. Available at: https://doi.org/10.1609/aaai.v34i05.6439.

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.

Vamvas, J. – Sennrich, R. As Little as Possible, as Much as Necessary: Detecting Over- and Undertranslations with Contrastive Conditioning. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), ACL 2022*, p. 490–500, Dublin, Ireland, 2022. doi: 10.18653/V1/2022.ACL-SHORT.53. Available at: https://doi.org/10.18653/v1/2022.acl-short.53.

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, INLG 2017, Santiago de Compostela*, p. 95–104, Spain, 2017. doi: 10.18653/V1/W17-3513. Available at: https://doi.org/10.18653/v1/w17-3513.

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. doi: 10.18653/V1/W18-6504. Available at: https://doi.org/10.18653/v1/w18-6504.

LEE, C. – GATT, A. – 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, INLG 2019*, p. 355–368, Tokyo, Japan, 2019. doi: 10.18653/V1/W19-8643. Available at: https://aclanthology.org/W19-8643/.

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, INLG 2020*, p. 68–79, Dublin, Ireland, 2020. doi: 10.18653/V1/2020.INLG-1.10. Available at: https://doi.org/10.18653/v1/2020.inlg-1.10.

LEE, C. – GATT, A. – MILTENBURG, E. – KRAHMER, E. Human Evaluation of Automatically Generated Text: Current Trends and Best Practice Guidelines. *Comput. Speech Lang.* 2021, 67, p. 101151. doi: 10.1016/J.CSL.2020.101151. Available at: https://doi.org/10.1016/j.csl.2020.101151.

MILTENBURG, E. – ELLIOTT, D. – VOSSEN, P. Measuring the Diversity of Automatic Image Descriptions. In *Proceedings of the 27th International Conference on Computational Linguistics, COLING 2018, Santa Fe, New Mexico, USA, August 20-26*, p. 1730–1741, 2018. Available at: https://aclanthology.org/C18-1147/.

MILTENBURG, E. – CLINCIU, M. – DUŠEK, 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 *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, L. – POLOSUKHIN, I. Attention Is All You Need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA,*, p. 5998–6008, USA, 2017. Available at: https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html.

VEJVAR, M. – FUJIMOTO, Y. ASPIRO: Any-shot Structured Parsing-error-Induced ReprOmpting for Consistent Data-to-Text Generation. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, p. 3550–3563, Singapore, 2023. doi: 10.18653/V1/2023.FINDINGS-EMNLP.229. Available at: https://doi.org/10.18653/v1/2023.findings-emnlp.229.

VESELOVSKY, V. – RIBEIRO, M. H. – WEST, R. Artificial Artificial Artificial Intelligence: Crowd Workers Widely Use Large Language Models for Text Production Tasks. *CoRR*. 2023, abs/2306.07899. doi: 10.48550/ARXIV.2306.07899. Available at: https://doi.org/10.48550/arXiv.2306.07899.

VRANDECIC, 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. – SHI, H. – LI, Z. – XU, J. – QU, J. – ZHOU, J. Is ChatGPT a Good NLG Evaluator? A Preliminary Study. *CoRR*. 2023a, abs/2303.04048. doi: 10.48550/ARXIV.2303.04048. Available at: https://doi.org/10.48550/arXiv.2303.04048.

WANG, J. On Computational Sentence Generation From Logical Form. In *Proceedings of the 8th International Conference on Computational Linguistics, COLING '80*, p. 405–411, Tokyo, Japan, 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 *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023*, p. 16646–16661, Singapore, 2023b. doi: 10.18653/V1/2023.EMNLP-MAIN.1036. Available at: https://doi.org/10.18653/v1/2023.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. – WANG, X. – QIN, Y. – PACKER, B. – LI, K. – CHEN, J. – BEUTEL, A. – CHI, E. H. CAT-Gen: Improving Robustness in NLP Models via Controlled Adversarial Text Generation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020*, p. 5141–5146, Online, 2020. doi: 10.18653/V1/2020.EMNLP-MAIN.417. Available at: https://doi.org/10.18653/v1/2020.emnlp-main.417.

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, Hawaii, 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*. 2021, 9, p. 176–194. doi: 10.1162/TACL_A_00360. Available at: https://doi.org/10.1162/tacl_a_00360.

Wang, Y. – Deng, J. – Sun, A. – Meng, X. Perplexity from PLM Is Unreliable for Evaluating Text Quality. *CoRR*. 2022, abs/2210.05892. doi: 10.48550/ARXIV.2210.05892. Available at: https://doi.org/10.48550/arXiv.2210.05892.

Wang, Z. et al. Interactive Natural Language Processing. *CoRR*. 2023d, abs/2305.13246. doi: 10.48550/ARXIV.2305.13246. Available at: https://doi.org/10.48550/arXiv.2305.13246.

Wei, J. et al. Emergent Abilities of Large Language Models. *Trans. Mach. Learn. Res.* 2022a, 2022. Available at: https://openreview.net/forum?id=yzkSU5zdwD.

Wei, J. – Wang, X. – Schuurmans, D. – Bosma, M. – Ichter, B. – Xia, F. – Chi, E. H. – Le, Q. V. – Zhou, D. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022*, New Orleans, LA, USA, 2022b. Available at: http://papers.nips.cc/paper_files/paper/2022/hash/9d5609613524ecf4f15af0f7b31abca4-Abstract-Conference.html.

Wen, T. – Young, S. J. Recurrent Neural Network Language Generation for Spoken Dialogue Systems. *Comput. Speech Lang.* 2020, 63, p. 101017. doi: 10.1016/J.CSL.2019.06.008. Available at: https://doi.org/10.1016/j.csl.2019.06.008.

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. – Gasic, M. – Mrksic, N. – Su, P. – Vandyke, D. – Young, S. J. Semantically Conditioned LSTM-based Natural Language Generation for Spoken Dialogue Systems. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015*, p. 1711–1721, Lisbon, Portugal, 2015b. doi: 10.18653/V1/D15-1199. Available at: https://doi.org/10.18653/v1/d15-1199.

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 *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. – CONNEAU, A. – CHAUDHARY, V. – GUZMÁN, F. – JOULIN, A. – GRAVE, E. CCNet: Extracting High Quality Monolingual Datasets from Web Crawl Data. In *Proceedings of The 12th Language Resources and Evaluation Conference, LREC 2020*, p. 4003–4012, Marseille, France, 2020. Available at: https://aclanthology.org/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*, Copenhagen, Denmark, 2007. Available at: https://aclanthology.org/2007.mtsummit-ucnlg.4.

WILCOXON, F. Individual Comparisons by Ranking Methods. In *Breakthroughs in statistics.* : Springer, 1992. p. 196–202.

WILLIAMS, A. – NANGIA, N. – BOWMAN, S. R. 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, NAACL-HLT 2018, Volume 1 (Long Papers)*, p. 1112–1122, New Orleans, Louisiana, USA, 2018. doi: 10.18653/V1/N18-1101. Available at: https://doi.org/10.18653/v1/n18-1101.

WISEMAN, S. – SHIEBER, S. M. – RUSH, A. M. Challenges in Data-to-Document Generation. In *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 *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. *CoRR*. 2019, abs/1910.03771. Available at: http://arxiv.org/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.

WU, Y. et al. Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation. *CoRR*. 2016, abs/1609.08144. Available at: http://arxiv.org/abs/1609.08144.

Xiang, J. – Liu, Z. – Zhou, Y. – Xing, E. P. – Hu, Z. ASDOT: Any-Shot Data-to-Text Generation with Pretrained Language Models. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, p. 1886–1899, Abu Dhabi, United Arab Emirates, 2022. doi: 10.18653/V1/2022.FINDINGS-EMNLP.136. Available at: https://doi.org/10.18653/v1/2022.findings-emnlp.136.

Xie, T. et al. UnifiedSKG: Unifying and Multi-Tasking Structured Knowledge Grounding with Text-to-Text Language Models. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, EMNLP 2022*, p. 602–631, Abu Dhabi, United Arab Emirates, 2022. doi: 10.18653/V1/2022.EMNLP-MAIN.39. Available at: https://doi.org/10.18653/v1/2022.emnlp-main.39.

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, INLG 2023*, p. 323–351, Prague, Czechia, 2023. doi: 10.18653/V1/2023.INLG-MAIN.23. Available at: https://doi.org/10.18653/v1/2023.inlg-main.23.

Xu, W. – Wang, D. – Pan, L. – Song, Z. – Freitag, M. – Wang, W. Y. – Li, L. INSTRUCTSCORE: Explainable Text Generation Evaluation with Finegrained Feedback, 2023a. Available at: http://arxiv.org/abs/2305.14282.

Xu, X. – Dušek, O. – Rieser, V. – Konstas, I. AggGen: Ordering and Aggregating while Generating. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, (Volume 1: Long Papers)*, p. 1419–1434, Virtual Event, 2021. doi: 10.18653/V1/2021.ACL-LONG.113. Available at: https://doi.org/10.18653/v1/2021.acl-long.113.

Xu, X. – Titov, I. – Lapata, M. Compositional Generalization for Data-to-Text Generation. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, p. 9299–9317, Singapore, 2023b. doi: 10.18653/V1/2023.FINDINGS-EMNLP.623. Available at: https://doi.org/10.18653/v1/2023.findings-emnlp.623.

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. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2021*, p. 483–498, Online, 2021. doi: 10.18653/V1/2021.NAACL-MAIN.41. Available at: https://doi.org/10.18653/v1/2021.naacl-main.41.

Yang, J. – Jin, H. – Tang, R. – Han, X. – Feng, Q. – Jiang, H. – Zhong, S. – Yin, B. – Hu, X. B. Harnessing the Power of LLMs in Practice: A Survey on ChatGPT and Beyond. *ACM Trans. Knowl. Discov. Data.* 2024, 18, 6, p. 160:1–160:32. doi: 10.1145/3649506. Available at: https://doi.org/10.1145/3649506.

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.

Yin, Z. – Sun, Q. – Guo, Q. – Wu, J. – Qiu, X. – Huang, X. Do Large Language Models Know What They Don't Know? In *Findings of the Association for Computational Linguistics: ACL 2023*, p. 8653–8665, Toronto, Canada, 2023. doi: 10.18653/V1/2023.FINDINGS-ACL.551. Available at: https://doi.org/10.18653/v1/2023.findings-acl.551.

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. In *Proceedings of the 14th International Conference on Recent Advances in Natural Language Processing, RANLP 2023*, p. 1256–1264, Varna, Bulgaria, 2023. Available at: https://aclanthology.org/2023.ranlp-1.133.

Zarrieß, S. – Voigt, H. – Schüz, S. Decoding Methods in Neural Language Generation: A Survey. *Inf.* 2021, 12, 9, p. 355. doi: 10.3390/INFO12090355. Available at: https://doi.org/10.3390/info12090355.

Zha, Y. – Yang, Y. – Li, R. – Hu, Z. AlignScore: Evaluating Factual Consistency with A Unified Alignment Function. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2023*, p. 11328–11348, Toronto, Canada, 2023. doi: 10.18653/V1/2023.ACL-LONG.634. Available at: https://doi.org/10.18653/v1/2023.acl-long.634.

Zhang, H. – Song, H. – Li, S. – Zhou, M. – Song, D. A Survey of Controllable Text Generation using Transformer-based Pre-trained Language Models. *ACM Comput. Surv.* 2024, 56, 3, p. 64:1–64:37. doi: 10.1145/3617680. Available at: https://doi.org/10.1145/3617680.

Zhang, T. – Kishore, V. – Wu, F. – Weinberger, K. Q. – Artzi, Y. BERTScore: Evaluating Text Generation with BERT. In *8th International Conference on Learning Representations, ICLR 2020*, Addis Ababa, Ethiopia, 2020a. Available at: https://openreview.net/forum?id=SkeHuCVFDr.

Zhang, T. – Ladhak, F. – Durmus, E. – Liang, P. – McKeown, K. R. – Hashimoto, T. B. Benchmarking Large Language Models for News Summarization. *CoRR.* 2023, abs/2301.13848. doi: 10.48550/ARXIV.2301.13848. Available at: https://doi.org/10.48550/arXiv.2301.13848.

Zhang, Z. – Wu, Y. – Zhao, H. – Li, Z. – Zhang, S. – Zhou, X. – Zhou, X. Semantics-Aware BERT for Language Understanding. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020*, p. 9628–9635, New York, NY, USA, 2020b. doi: 10.1609/AAAI.V34I05.6510. Available at: https://doi.org/10.1609/aaai.v34i05.6510.

Zhao, W. X. et al. A Survey of Large Language Models. *CoRR*. 2023a, abs/2303.18223. doi: 10.48550/ARXIV.2303.18223. Available at: https://doi.org/10.48550/arXiv.2303.18223.

Zhao, W. – Peyrard, M. – Liu, F. – Gao, Y. – Meyer, C. M. – Eger, S. MoverScore: Text Generation Evaluating with Contextualized Embeddings and Earth Mover Distance. 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 2019*, p. 563–578, Hong Kong, China, 2019. doi: 10.18653/V1/D19-1053. Available at: https://doi.org/10.18653/v1/D19-1053.

Zhao, Y. – Qi, Z. – Nan, L. – Mi, B. – Liu, Y. – Zou, W. – Han, S. – Chen, R. – Tang, X. – Xu, Y. – Radev, D. – Cohan, A. QTSumm: Query-Focused Summarization over Tabular Data. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, p. 1157–1172, Singapore, December 2023b. doi: 10.18653/v1/2023.emnlp-main.74. Available at: https://aclanthology.org/2023.emnlp-main.74.

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, 2023c. Available at: http://arxiv.org/abs/2305.14987.

Zheng, L. – Chiang, W. – Sheng, Y. – Zhuang, S. – Wu, Z. – Zhuang, Y. – Lin, Z. – Li, Z. – Li, D. – Xing, E. P. – Zhang, H. – Gonzalez, J. E. – Stoica, I. Judging LLM-as-a-Judge With MT-Bench and Chatbot Arena. *CoRR*. 2023, abs/2306.05685. doi: 10.48550/ARXIV.2306.05685. Available at: https://doi.org/10.48550/arXiv.2306.05685.

Zhong, V. – Xiong, C. – Socher, R. Seq2SQL: Generating Structured Queries from Natural Language using Reinforcement Learning. *CoRR*. 2017, abs/1709.00103. Available at: http://arxiv.org/abs/1709.00103.

Zhou, H. – Bradley, A. – Littwin, E. – Razin, N. – Saremi, O. – Susskind, J. – Bengio, S. – Nakkiran, P. Understanding Length Generalization by Thinking Like Transformers. In *The 3rd Workshop on Mathematical Reasoning and AI at NeurIPS'23*, 2023. Available at: https://openreview.net/forum?id=tEUJiua8ir.

# List of Abbreviations

# List of Tables

# List of Figures

# List of Publications


Kasner, Z. – Dušek, O. Train Hard, Finetune Easy: Multilingual Denoising for RDF-to-Text Generation. In *Proceedings of the 3rd International Workshop on Natural Language Generation from the Semantic Web (WebNLG+)*, p. 171–176, Dublin, Ireland (Virtual), 12 2020b. Available at: https://aclanthology.org/2020.webnlg-1.20

- The data-to-text generation system based on the finetuned mBART model (Section 3.1).
- Our submission for the WebNLG+ shared task.
- Citations (without self-citations): 9

Kasner, Z. – Dušek, O. Data-to-Text Generation with Iterative Text Editing. In *Proceedings of the 13th International Conference on Natural Language Generation, INLG 2020*, p. 60–67, Dublin, Ireland, 2020a. doi: 10.18653/V1/2020.INLG-1.9. Available at: https://doi.org/10.18653/v1/2020.inlg-1.9

- The data-to-text generation system based on iterative text editing (Section 3.2).
- Citations (without self-citations): 17

Kasner, Z. – Dušek, O. Neural Pipeline for Zero-Shot Data-to-Text Generation. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2022*, p. 3914–3932, Dublin, Ireland, 2022. doi: 10.18653/V1/2022.ACL-LONG.271. Available at: https://doi.org/10.18653/v1/2022.acl-long.271

- The data-to-text generation system based on a pipeline of neural modules (Section 3.3).
- Citations (without self-citations): 21

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, INLG 2020*, p. 131–137, Dublin, Ireland, 2020. doi: 10.18653/V1/2020.INLG-1.19. Available at: https://doi.org/10.18653/v1/2020.inlg-1.19

- The metric for detecting omissions and hallucinations in generated texts (Section 4.2).

- Best short paper at INLG 2020.
- Citations (without self-citations): 47

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, INLG 2021*, p. 259–265, Aberdeen, Scotland, UK, 2021. doi: 10.18653/V1/2021.INLG-1.25. Available at: https://doi.org/10.18653/v1/2021.inlg-1.25

- The metric for token-level error detection in generated texts (Section 4.2)
- Our submission to the shared task Evaluating Accuracy in Generated Texts.
- Citations (without self-citations): 6

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: System Demonstrations, ACL 2023*, p. 444–455, Toronto, Canada, 2023a. doi: 10.18653/V1/2023.ACL-DEMO.42. Available at: https://doi.org/10.18653/v1/2023.acl-demo.42

- The toolkit for processing and visualization of data-to-text generation datasets (Section 5.1).
- Citations (without self-citations): 2

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, EACL 2023, Dubrovnik*, p. 2390–2407, Croatia, 2023b. doi: 10.18653/V1/2023.EACL-MAIN.176. Available at: https://doi.org/10.18653/v1/2023.eacl-main.176

- The analysis of verbalizing relations in knowledge graphs with pretrained language models (Section 6.1).
- Citations (without self-citations): 3

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

- The analysis of data-to-text generation with open large language models (Section 6.2).
- Citations (without self-citations): 2

Only publications relevant to this thesis are included. The number of citations was computed using Semantic Scholar API. Total number of citations of publications related to the topic of the thesis (without self-citations) by June 14, 2024: **107**.