UNIVERSITY OF CALIFORNIA
SANTA CRUZ

DIVERSIFYING LANGUAGE GENERATED BY DEEP LEARNING MODELS IN DIALOGUE SYSTEMS

A dissertation submitted in partial satisfaction
of the requirements for the degree of

DOCTOR OF PHILOSOPHY

in

COMPUTER SCIENCE

by

Juraj Juraska

September 2022

The Dissertation of Juraj Juraska
is approved:

Professor Marilyn Walker, Chair

Professor Jim Whitehead

Scott Roy, Ph.D.

Peter Biehl
Vice Provost and Dean of Graduate Studies

Table of Contents

List of Figures

List of Tables

Abstract

Diversifying Language Generated by Deep Learning Models in Dialogue Systems

by

Juraj Juraska

Conversational AI has seen tremendous progress in recent years, achieving near-human or even surpassing human performance in certain well-defined tasks, including speech recognition and question answering. Yet it tends to struggle with tasks which are less constrained, in particular those that involve producing human language. Current approaches to natural language generation (NLG) in dialogue systems still heavily rely on techniques that lack scalability and transferability to different domains, despite the general embrace of more robust methods by the NLG community, in particular deep learning (neural) models. These methods rely on large amounts of annotated data, yet they tend to produce generic, robotic, and boring responses that lack most of the human language nuances that make conversation creative and varied.

While the naturalness of the generated language is an important factor affecting the perceived quality of a dialogue system, semantic accuracy is also extremely important. If a system is not semantically accurate, it may provide the user with incorrect information or contradict its earlier responses. In this thesis, we focus on the task of generating an utterance from a structured meaning representation (MR). To support our work, we create and release a new parallel corpus with more varied dialogue acts and more conversational utterances than previous

MR-to-text corpora. We explore different ways of promoting output diversity in neural data-to-text generation while ensuring high semantic accuracy by developing new methods to help deep learning NLG models produce diverse utterances that are faithful to their MRs. This is an important step toward making conversational AI more reliable and pleasant to interact with.

We first observe in our initial experiments that NLG models have the ability to produce more diverse and natural-sounding texts when explicitly prompted to, however, this diversity comes at the expense of semantic accuracy. This leads us to develop a set of methods for automatically assessing and enforcing semantic accuracy in the generated utterances. We introduce a general tool to find a semantic alignment between an utterance and the corresponding input, which can be used for automatically evaluating the accuracy of generated utterances and ranking a pool of candidate utterances a model produces. We also propose a novel semantically attention-guided decoding method for neural encoder-decoder models, which utilizes the models' own knowledge acquired from training in a way that enables them to track semantic accuracy during inference and rerank generated utterance candidates accordingly. We show on multiple datasets that both of these methods have an ability to dramatically reduce semantic errors in model outputs, while maintaining their overall quality and fluency.

We then systematically explore Monte-Carlo Tree Search (MCTS) as a way to simultaneously optimize both semantic accuracy and stylistic diversity during inference. To guide the MCTS, we propose a new referenceless automatic metric for utterance evaluation. Our results show that, using this novel method, we can successfully increase diversity while maintaining, or even improving, semantic accuracy.

To my fiancée Nehal and my family.

Acknowledgments

When I started my PhD program, I had no clue about the field of NLP, despite having taken several AI classes earlier. It wasn't until the end of my first year, when I took Lyn Walker's Conversational Agents class, that I discovered this fascinating research area at the intersection of computer science and linguistics.

At that time when I was struggling to find direction on my path to PhD, Lyn took me under her wing and believed in me, for which I am forever grateful to her. As my advisor, Lyn offered me unwavering support throughout the years that followed and, with her expertise and love for NLP and dialogue systems, rekindled my passion for AI. Thanks to her invaluable guidance but also her trust in my own judgment, I grew as a researcher, gradually becoming independent and confident in tackling problems that did not have an obvious solution.

My dissertation would have gone a different route, had I not been lucky to meet Scott Roy. As my mentor during my first internship at Google, he showed nothing but patience and care, and he treated me as an equal, listening to my ideas and frequently communicating appreciation along with thoughtful feedback. Working with Scott both during and after the internship has been a remarkably empowering experience, and I am very grateful for that opportunity. As a brilliant yet empathetic leader with a contagious enthusiasm for deep learning and conversational AI, Scott has been, and will continue to be, an inspiration and a role model to me.

Next, I want to thank my advancement and dissertation reading committee – Lyn, Scott, Jim Whitehead and Jeff Flanigan – for their advice on my early work, and for taking the time to evaluate my dissertation and providing me with valuable feedback.

When taking a break from my dissertation research in the summers, I had the opportunity to work with several wonderful and smart people at Google and Microsoft Research. Elahe Rahimtoroghi, Grady Simon, Markus Freitag, Mihir Kale, Partha Parthasarathy, and William Gale – these colleagues and mentors not only helped me navigate the non-academic landscape and understand how to do research in an industry setting, but also made sure my internship experiences were all excellent.

On my journey towards PhD, I collaborated closely with many people at UCSC. Among them I would like to especially thank Kevin for the first prototype of the slot aligner, for his help with creating the ViGGO corpus, and for making attending conferences and Alexa Prize summits fun. I also want to thank my other contemporary and former labmates Davan, Wen, Lena, Jiaqi, Shereen, Angela, Omkar, Rishi, Nikhil, Brian and Abteen, with whom I spent years developing and tirelessly improving our SlugBot and Athena chatbots. This list would not be complete without mentioning Panos, who was instrumental in developing our Slug2Slug system with which we won the E2E NLG Challenge back in 2017, which in turn led to my first and most successful conference paper thus far.

After moving to California to start my PhD, I was fortunate beyond measure with my first housing at Sonya and Jeff's place. Not only did they maintain a more-than-fair rent for the whole two years I stayed with them, despite the outrageous housing market in Santa Cruz, they even let me permanently keep their son's digital piano in my room, since I had to leave mine behind at home in Slovakia. As if that was not enough, I had access to the best tomatoes, strawberries and persimmons straight from their garden to fuel my brain all year round. Thank you, Sonya and Jeff, for your incredible kindness and making me feel at home after having

moved half across the world, as well as for all our conversations and the occasional game night!

Zuzka and Raman, dearest of all my friends, thank you for not forgetting about me after I left Europe. I can't express how much I treasure our friendship, as well as your empathy and encouragement whenever we get to talk. I know that, if I were ever to move back to Praha, we would pick up right where we left off in 2016 as if no time had passed at all!

Of course, I would not be where I am without my family. First, my big brother, Michal, was the one who inspired me to embark on the PhD journey in the U.S. in the first place and who understood my struggles the most. Second, my brother Tomik has made sure I preserved my sanity until the end of my PhD program by frequently checking in on me and letting me momentarily escape the reality during our impromptu gaming sessions, as well as by arranging for us to go for a trip or at least meet once a year despite the physical distance between us. But most of all, I am grateful to my wonderful parents, Eugénia and Dalibor, who encouraged me to travel and start getting to know the world from an early age, and eventually supported my decision to pursue my academic goals on the other side of the planet. I try not to see the distance as an obstacle though, as I think, if anything, it has even strengthened our bond. That being said, I miss you a lot, mamina and tatino! Thank you for your infinite love and patience, and also for making me take piano classes as a kid, as it was only much later that I discovered the profound effect of making music on my inner peace and sense of harmony no matter where I am.

Finally, I would like to thank my best friend and fiancée Nehal for her constant love and support all throughout my PhD program, for her solidarity in staying up with me all night (or dozing off on the couch next to my desk) when I had

deadlines, for commiserating with me when my papers got rejected and celebrating when they got accepted, and for making me laugh at times when I thought I couldn't muster even a smile. In short, I am truly grateful to her for being always by my side, and not stopping believing in me for a second, even when I myself had serious doubts about my ability or resolve to complete my PhD. Thank you for making life so beautiful!

Chapter 1

Introduction

As technology permeates every aspect of our lives, dialogue systems are becoming increasingly prevalent in facilitating our interactions with devices and services by enabling us to communicate in human language – the most natural and convenient way. While task-oriented dialogue systems can already be found in a range of settings from smartphones to customer service to event reservations, conversational agents, or "chatbots", currently have a rather limited scope of application. Although chatbots remain restricted mostly to entertainment for now, their potential lies in fields like healthcare and therapy, education, or consulting. The primary reason for their slow advancement is the enormous complexity in developing such systems that need to be able to understand any type of utterance in the given context and respond to it in a coherent and natural way irrespective of the domain.

There has recently been a substantial amount of research and rapid progress in natural language processing (NLP), achieving near-human or even surpassing human performance in certain well-defined tasks, including automatic speech recognition and question answering. However, the capabilities of digital personal assistants, such as Google Assistant or Alexa, remain fairly limited and lacking in various as-

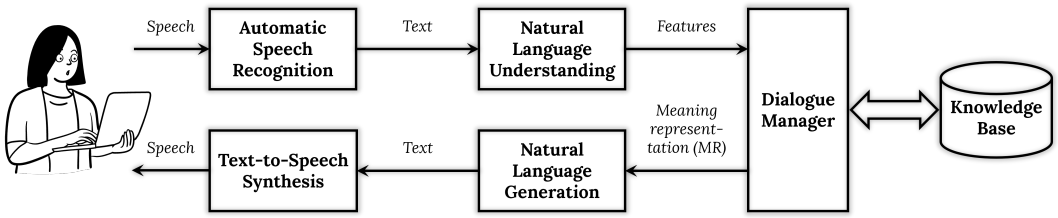


Figure 1.1: Standard architecture of a spoken dialogue system.

pects, one of the most challenging of which is the ability to produce responses with human-like coherence and naturalness on any topic and for many different kinds of content. In a traditional (i.e., not end-to-end) dialogue system, this is the responsibility of the natural language generation (NLG) component (see Figure 1.1).

NLG in conversational AI, in general, remains a difficult task, primarily due to it being a less constrained task which involves producing rich human language. While end-to-end generation has greatly improved since the advent of large pretrained language models, even models like DialoGPT (Zhang et al., 2020b), GPT-3 (Brown et al., 2020) or LaMDA (Thoppilan et al., 2022) are not quite ready for a wide-spread adoption in customer-facing systems because of their lack of predictability and interpretability. Since end-to-end generation approaches currently have a limited ability to maintain a coherent and engaging conversation on a subject for an extended number of turns, open-domain dialogue systems like the "socialbots" developed in the Alexa Prize competition (Khatri et al., 2018; Bowden et al., 2018; Harrison et al., 2019; Juraska et al., 2021; Hu et al., 2021) still rely to some extent on handcrafted rules and response retrieval for sufficient controllability.

In task-oriented dialogue systems, controllability is probably even more essential, as they require high semantic fidelity of the generated responses in order to correctly track what information exactly has been exchanged with the user. There-

fore, their NLG components are typically conditioned on structured input data, performing *data-to-text* generation. In a typical task-oriented dialogue system, at each turn in a conversation, the dialogue manager produces an object with the desired content for the system's response, and possibly other metadata specifying additional aspects of the response. This object is passed on to the NLG component, whose responsibility is to generate a fluent natural-language utterance with all the desired content faithfully and adequately realized. The structured data object can take on different forms, ranging from simple meaning representations (Mairesse et al., 2010; Mairesse and Young, 2014; Wen et al., 2016; Novikova et al., 2017b; Juraska et al., 2019), to graph-based abstract meaning representations (Banarescu et al., 2013), to RDF triples (Gardent et al., 2017; Ferreira et al., 2020; Nan et al., 2021).

Besides dialogue systems, data-to-text NLG has many successful applications in various domains (Sharma et al., 2022), including finance (Plachouras et al., 2016; Murakami et al., 2017), healthcare (Pauws et al., 2019), journalism (Leppänen et al., 2017), sports (Chen and Mooney, 2008; Wiseman et al., 2017) and weather (Liang et al., 2009; Balakrishnan et al., 2019), in the majority of which the task is to provide a summary of the information given in the structured input. In contrast to data-to-text generation for dialogue, they often do not require all the source content to be mentioned in the output text. Depending on the domain, the data format in these tasks ranges from time series (Reiter et al., 2005) to tables (Lebret et al., 2016; Chen et al., 2020; Parikh et al., 2020).

In our work, we focus on data-to-text NLG for dialogue systems, with simple *meaning representation* (MR) as the input format. This type of MR provides information about the content to be conveyed in the response (utterance) and can optionally indicate the *dialogue act* (DA) type (e.g., yes/no question, or opinion)

MR	*inform* (NAME [**The Waterman**], FOOD [**English**], PRICERANGE [**cheap**], CUSTOMERRATING [**low**], AREA [**city centre**], FAMILYFRIENDLY [**yes**])
Utt. #1	**The Waterman** is a **family-friendly** restaurant in the **city centre**. It serves **English** food at a **cheap** price. It has a **low** customer rating.
Utt. #2	There is a **cheap**, **family-friendly** restaurant in the **city centre**, called **The Waterman**. It serves **English** food, but received a **low** rating by customers.

Table 1.1: Example of a meaning representation (MR) and two corresponding utterances of different styles.

and other specifications for the utterance. It is typically structured as a list, in which each element is a slot-value pair, where the *slot* specifies the type of information, and the *value* is the corresponding content. Table 1.1 shows an example MR for a restaurant called "The Waterman" along with two (out of many) possible utterances expressing all the given information in two different ways.

Although in task-oriented dialogue systems, it is typically feasible to design a set of templates or syntactic rules for responses, this approach lacks scalability, as well as transferability to different domains. As a result, more robust methods have become favored in recent years, employing statistical or deep learning models.[1] While large pretrained generative language models (LMs), such as GPT-2 (Radford et al., 2019) or T5 (Raffel et al., 2020), are remarkably good at generating fluent text, when fine-tuned on a data-to-text NLG task, even they often fail to produce text that reliably and correctly mentions all the information provided in the input (Li et al., 2022a). In fact, pretrained LMs also tend to hallucinate information that is not supported by the inputs, but might have been present in their training data in a related context (Maynez et al., 2020; Raunak et al., 2021; Wang et al., 2021; Ji et al., 2022). To achieve high semantic accuracy, neural models for data-to-text NLG

[1]Since deep learning models typically involve some kind of neural networks, they are often referred to as "neural" models, and we use the two terms interchangeably.

have invariably been reliant on extrinsic components or methods, which typically require training a separate classifier or changing the NLG model's architecture. Furthermore, data-to-text NLG models often rely on relatively large amounts of annotated training data, yet they tend to produce generic sentences that lack most of the human language nuances that make it creative and varied. The naturalness of the generated language is an important factor affecting the perceived quality of a dialogue system. We therefore explore different ways of ensuring output diversity in neural data-to-text generation without a negative impact on semantic accuracy.

We believe that progress in dialogue systems overall has been stymied by the challenges of creating robust natural language generators for dialogue. In particular we posit that progress in this field is dependent on novel methods for:

- NLG models that can simultaneously control for semantic and DA accuracy;
- NLG models that produce diverse utterances that are stylistically varied and natural;
- Automatic evaluation metrics that can precisely measure semantic accuracy, DA accuracy, and stylistic diversity.

This thesis contributes to addressing these challenges. In order to contextualize and motivate our work, Section 1.1 gives an overview of different approaches to NLG, and Section 1.2 discusses in more detail the requirements for NLG models for dialogue. Section 1.3 then describes the state of the art for stylistic control in NLG, while Section 1.4 summarizes where the field is in terms of automatic evaluation metrics.

1.1 Natural Language Generation Approaches

A natural language generator must produce a syntactically and semantically correct utterance from a given MR that typically specifies both a DA type and a set of semantic attributes that should be realized in the generated utterance. The utterance should express all the information contained in the MR, in a natural and conversational way. In traditional language generator architectures, the assembling of an utterance from an MR is performed in two stages: *sentence planning*, which enforces semantic correctness and determines the structure of the utterance, and *surface realization*, which enforces syntactic correctness and produces the final utterance form.

Earlier work on statistical NLG approaches were typically hybrids of a handcrafted component and a statistical training method (Langkilde and Knight, 1998; Stent et al., 2004; Rieser and Lemon, 2010; Stent et al., 2004; Walker et al., 2007; Mairesse and Walker, 2007). The handcrafted aspects, however, lead to decreased portability and potentially limit the variability of the outputs. New corpus-based approaches emerged that used semantically aligned data to train language models that generate utterances directly from the MRs (Mairesse et al., 2010; Mairesse and Young, 2014). The alignment provides valuable information during training, but the semantic annotation is costly.

More recent methods do not require aligned data and use an *end-to-end* approach to training, performing sentence planning and surface realization simultaneously (Konstas and Lapata, 2013). The first successful systems using this end-to-end training paradigm utilized recurrent neural networks (RNNs) paired with an encoder-decoder system design (Mei et al., 2016; Dušek and Jurčíček, 2016), but also

other concepts, such as imitation learning (Lampouras and Vlachos, 2016). These NLG models, however, typically require greater amount of data for training due to the lack of semantic alignment, and they still have problems producing syntactically and semantically correct output, as well as being limited in naturalness (Nayak et al., 2017).

Many recent advances in NLG have come from the field of machine translation (Chen et al., 2018). Although machine translation is a text-to-text NLG task, data-to-text NLG is closely related and has similarly benefited from recent rapid development of deep learning methods. Data-to-text NLG systems thus gradually also moved toward neural *sequence-to-sequence* models (Sutskever et al., 2014) with an *encoder-decoder* architecture (Cho et al., 2014) and equipped with an *attention* mechanism (Bahdanau et al., 2015). Even more recently, purely attention-based models, based on the transformer (Vaswani et al., 2017) architecture, started consistently outperforming RNN-based sequence-to-sequence models. Both that and their significantly faster training times are why they quickly became widespread.

NLG methods continued evolving rapidly since then, and the paradigm has shifted to very large transformer-based language models pretrained on massive amounts of text data in an unsupervised way, and subsequently fine-tuned on much smaller domain-specific datasets. This gave birth to innumerable models with different pretraining techniques and corpora, among which GPT-2 (Radford et al., 2019), an autoregressive decoder-only model, became the first popular and widely-used model in NLG. It was, however, more suitable for open-ended than data-to-text generation. GPT-2 spawned many variants, including CTRL (Keskar et al., 2019), Transformer-XL (Dai et al., 2019) and Reformer (Kitaev et al., 2020). These were soon followed by a number of full-fledged encoder-decoder models, including T5 (Raf-

fel et al., 2020), BART (Lewis et al., 2020) and ProphetNet (Qi et al., 2020), which are generally better suited for sequence-to-sequence tasks like data-to-text NLG.

Most recently, pretrained NLG models grew to behemoths with hundreds of billions of parameters, such as GPT-3 (Brown et al., 2020), Jurassic-1 (Lieber et al., 2021) and BLOOM (Laurençon et al., 2022). These models often take months to train on powerful GPUs, but once they are trained, they are sufficiently robust to tackle tasks in zero- or few-shot setting. For data-to-text NLG, this means that a model could generate a reasonable utterance by just seeing a definition of the task or a few example MR-utterance pairs provided along with the input MR. Nevertheless, these gigantic models typically only have restricted access to inference through an API, and are therefore not practical for use in developing new methods for NLG models.

In this thesis, we experiment with a variety of neural approaches ranging from RNN-based encoder-decoder models with attention that we train from scratch, to fine-tuning pretrained transformer-based language models, such as T5 and BART.

1.2 Semantic Control in NLG

Prior to the advent of pretrained language models, NLG systems would rely on slot delexicalization (Mairesse et al., 2010; Henderson et al., 2014), which allows the model to better generalize to unseen inputs, as exemplified by TGen (Dušek and Jurčíček, 2016). However, Nayak et al. (2017) point out that there are frequent scenarios where delexicalization behaves inadequately (see Section 4.2 for more details). More recently, a similar effect was achieved by using a copy mechanism (Vinyals et al., 2015; Gu et al., 2016; See et al., 2017) integrated directly into the model's

decoder, and Agarwal and Dymetman (2017) show that a character-level approach to NLG may avoid the need for delexicalization, at the potential cost of making more semantic omission errors.

The end-to-end approach to NLG typically requires a mechanism for aligning slots with the output utterances: this allows the model to generate utterances with fewer missing slots mentions or hallucinations. Cuayáhuitl et al. (2014) perform automatic slot labeling using a Bayesian network trained on a labeled dataset, and show that a method using spectral clustering can be extended to unlabeled data with high accuracy. In one of the first successful neural approaches to data-to-text generation, Wen et al. (2015a) augment the generator's inputs with a control vector indicating which slots still need to be realized at each step. Wen et al. (2015b) take the idea further by embedding a new sigmoid gate into the LSTM cells of their RNN network, which directly conditions the generator on the input MR. The coverage mechanism (Tu et al., 2016; Mi et al., 2016; See et al., 2017) similarly tracks the already realized information within the model itself. Dušek and Jurčíček (2016), on the other hand, supplement their encoder-decoder model with a trainable classifier which they use to rerank the beam search candidates based on incorrect slot mentions. These approaches mostly depend on a high-quality parallel training corpus, where all reference utterances should correctly mention all the slots in the corresponding MR. Unfortunately, that is not the case with all datasets, for instance, one of the largest data-to-text NLG datasets in the restaurant domain, the E2E dataset (Novikova et al., 2017b), which is popular due to its size for training end-to-end neural models, is known to be noisy (Dušek et al., 2019).

One of the contributions of this thesis is a general-purpose slot aligner that addresses some of these problems (Juraska et al., 2018). When we were first tackling

this data-to-text generation problem, we quickly realized that being able to align slots in an MR with their mentions in the corresponding utterance would be beneficial in multiple phases of training and evaluating the system. We thus gradually developed a heuristic slot aligner applicable in several different aligning tasks: besides its use for denoising and augmenting training data, it can complement a neural NLG model to enhance its semantic accuracy through utterance reranking, particularly in data-starved scenarios. Furthermore, it can be utilized for evaluation by indicating how many of the input MR's slots have not been realized in an utterance correctly. The slot aligner proves particularly useful when training our models on the popular, but very noisy, E2E dataset. In addition to the slot aligner, we propose a novel decoding method, SEA-GUIDE (Juraska and Walker, 2021), that utilizes the model's own attention mechanism to automatically track slot mentions during the inference and subsequently rerank the candidate utterances, giving preference to those that have all slots realized. Unlike the slot aligner, SEA-GUIDE works out of the box for any new domain.

1.3 Stylistic Variation in NLG

The restaurant domain has always been the domain of choice for NLG tasks in dialogue systems (Stent et al., 2004; Gašić et al., 2008; Mairesse et al., 2010; Howcroft et al., 2013), as it offers a good combination of structured information availability, expression complexity, and ease of incorporation into conversation. Hence, even the more recent neural models for NLG continue to be tested primarily on data in this domain (Wen et al., 2015b; Dušek and Jurčíček, 2016; Nayak et al., 2017). These tend to focus solely on syntactic and semantic correctness of the gener-

ated utterances, nevertheless, there have also been recent efforts to collect training data for NLG with emphasis on stylistic variation (Nayak et al., 2017; Novikova et al., 2017b; Oraby et al., 2017).

While there is previous work on stylistic variation in NLG (Paiva and Evans, 2004; Mairesse and Walker, 2007), this work did not use crowd-sourced utterances for training. More recent work in neural NLG that explores stylistic control has not needed to control semantic correctness, or examined the interaction between semantic correctness and stylistic variation (Sennrich et al., 2016; Ficler and Goldberg, 2017). Also related is the work of Niu and Carpuat (2017) that analyzes how dense word embeddings capture style variations, Kabbara and Cheung (2016) who explore the ability of neural NLG systems to transfer style without the need for parallel corpora, which are difficult to collect (Rao and Tetreault, 2018), while Li et al. (2018) use a simple delete-and-retrieve method also without alignment to outperform adversarial methods in style transfer. Finally, Oraby et al. (2018) propose two different methods that give neural generators control over the language style, corresponding to the Big Five personalities, while maintaining semantic fidelity of the generated utterances.

Overall, there has been a lack of research exploring the use of and utility of stylistic selection for controlling stylistic variation in NLG from structured MRs. This may be either because there have not been sufficiently large corpora in a particular domain until recently, or because it is surprising, as we show, that relatively small corpora (2,000 samples) whose style is controlled can be used to train a neural generator to achieve relatively high semantic correctness while producing stylistic variation.

Among the contributions of this thesis is a systematic exploration of differ-

ent ways of gaining more control over the style of a neural model's outputs. We were not satisfied with the generic-sounding utterances we had observed neural models to produce, and wanted to see if they are capable of generating more complex language, as also seen in the training set, only not as frequently. Take, for instance, the two alternative utterances for the same MR in Table 1.1. The first example utterance sounds almost robotic, while the second one might be considered stylistically interesting, since the name of the restaurant follows the mention of certain aspects of the description emphasized at the beginning of the utterance, and contains a concession in its second sentence. One goal of our work is to have neural models consistently generate utterances like the latter, if the training set contains such examples.

We experiment with training data manipulation and model input augmentation, both of which enable stylistic control to a certain degree (Juraska and Walker, 2018). We then propose an inference method for sequence-to-sequence neural models, that we posit should result in increased diversity of NLG outputs while maintaining high semantic accuracy. The inference method based on Monte-Carlo Tree Search (MCTS) automatically promotes diversity better than beam search and, at the same time, optimizes for an arbitrary metric. We developed a comprehensive referenceless metric to guide the tree search. Besides capturing the fluency of the language, the metric also reflects the semantic accuracy, i.e., whether the generated sentence correctly conveys all the information it is supposed to. By making generated language sound less repetitive and more natural, we seek to make the experience more pleasant for users regularly interacting with conversational AI.

1.4 Automatic Metrics for Data-to-Text NLG Evaluation

Automatic evaluation in data-to-text NLG is a challenging problem, and to this day remains without a standard method or metric, let alone one that could replace human evaluation. For years the data-to-text NLG community has been relying on metrics from other NLG disciplines, which are not entirely compatible with this task (Sai et al., 2022). The most popular one, BLEU (Papineni et al., 2002), works very well in machine translation, where the structure of the target outputs is strictly determined by the source text, and the generated texts are thus not supposed to deviate much from the references. ROUGE-L (Lin and Och, 2004), on the other hand, is less strict about exact phrasing but still expects information to be in the same order, which is effective in its original discipline of text summarization. Another example is the CIDEr (Vedantam et al., 2015) metric, proposed for image captioning evaluation, where the desired captions are typically very simple – often just incomplete sentences – and lack more advanced discourse relations. These are all reference-based metrics that compare a generated utterance with one or more reference utterances written by humans, and penalize it for differences.

There are several obvious conflicts between what these metrics were designed for and the basic principles of data-to-text language generation. Most importantly, in our task there is a multitude of very different, but equally correct and accurate, ways to express all but the simplest MRs. The utterances can express the input information in a virtually arbitrary order, as long as they are grammatical, and in many cases using synonymous expressions, varying phrasing, and possibly different discourse relations. Neural models are capable of capturing general con-

cepts across different samples in the training data, and subsequently applying them to new, previously unseen, inputs. Therefore, datasets rarely provide more than one reference utterance per MR. However, during evaluation, it means that if the sentence structure chosen by the model is not common, or at all present, among the reference utterances for the given input, the generated utterance will score lowly according to these metrics regardless of its correctness.

The majority of automatic metrics commonly used in NLG, including the above, depend to a high degree on lexical overlap with human-authored reference texts. Most of the metrics give better and more objective results with an increasing number of references available, however, collecting many references for each example in a dataset is expensive. Not surprisingly, automatic metrics in general were shown to only weakly correlate with human judgments in data-to-text NLG (Novikova et al., 2017a), and have long been criticized for not accurately reflecting the true quality of the texts generated by NLG models overall (Callison-Burch et al., 2006; Reiter and Belz, 2009; Smith et al., 2016). Other work suggests that, even when these metrics are shown to strongly correlate with human references, it is often not for the right reasons (Schluter, 2017; Caglayan et al., 2020). In general, their analyses suggest that automatic metrics, while reliable in indicating a model's poor performance, are weak in distinguishing mediocre utterances from good ones, like humans can. This may make these metrics adequate for model development, but, in most cases, not for distinguishing the best NLG system in a group of good systems. Automatic metrics have also been shown to penalize utterances for variation (Stent et al., 2005; Wang and Chan, 2019), which further clashes with our objective of making model outputs more diverse.

As a result, the data-to-text NLG community has been dependent on hu-

man evaluation, which is typically performed alongside of automatic metric evaluation. The main reason for that is that there is no consensus on human evaluation criteria in data-to-text NLG either, and new metrics with individual definitions are thus constantly being devised, tailored to the task at hand (Howcroft et al., 2020). More than 40 criteria across nearly 90 NLG papers from 2018 (van der Lee et al., 2019) are an evidence of the lack of consensus. Evaluating NLG systems is, in general, a challenging and nuanced task requiring linguistic and domain knowledge (Sai et al., 2022). Nevertheless, human evaluation is typically performed by crowdworkers who often lack such knowledge necessary to objectively judge the quality of model outputs in specific NLG tasks, especially when the texts are largely fluent and grammatically correct, which is mostly the case in the era of pretrained language models. It has been shown in other NLG fields that crowdsourced human evaluation is unreliable and more sophisticated automatic metrics even correlate better with professional annotators' ratings than the ratings of crowdworkers do (Freitag et al., 2021). All in all, automatic metric evaluation still has high value in the research community, as the standardized metric calculations allow for better independent benchmarking and system comparison.

The recent emergence of neural reference-based metrics offers a more robust alternative for automatic evaluation in NLG. They compare a pair of texts on a more semantic level, as opposed to focusing on surface-level similarities only. BERTScore, proposed in Zhang et al. (2020a), calculates the cosine similarity between the contextualized embeddings of individual candidate tokens and the reference tokens, and returns the F1 score of their best matches. A pretrained BERT-based model (Devlin et al., 2019) is used to calculate the contextual embeddings, i.e., learned vector representations of tokens given their context in the text (Liu et al., 2020a).

BLEURT (Sellam et al., 2020), on the other hand, does not try to explicitly find a semantic alignment between the candidate and the reference. Instead, it is a metric directly trained to score pairs of texts, in a way that makes it domain-agnostic. The metric was developed by further training the already pretrained BERT model with millions of synthetic sentence pairs obtained by perturbing Wikipedia sentences and automatically scored using a set of standard automatic metrics. It was subsequently fine-tuned on a smaller amount of data with human ratings. Although both of these metrics were developed with the machine translation task in mind, they are equally effective for data-to-text NLG tasks. BERTScore and BLEURT have a superior ability to capture semantic similarities between a pair of texts and have been shown to correlate with human judgments significantly better than metrics that only rely on lexical overlap.

There have also been a few efforts to come up with alternate forms of automatic evaluation avoiding the drawbacks of reference-based metrics. The first such metric for data-to-text NLG, proposed in Dušek et al. (2017), uses a neural model to predict the quality score for an utterance based on the corresponding MR only. The main drawback of their approach is that human ratings of multiple NLG systems on multiple datasets are required to train the model. This approach limits the cross-domain generalization capabilities of the metric, and it may be difficult for others to reproduce the same results. In another attempt at a referenceless metric, for the sentence compression task, Kann et al. (2018) show that a normalized unsupervised language model score (Lau et al., 2017) correlates with human evaluation of sentence fluency significantly more than ROUGE-L. By further combining the language model with the ROUGE-L metric, they achieved an additional – and even more substantial – increase in correlation. This indicates that the two components

account for complementary aspects (fluency and adequacy) of the generated texts, and together form a highly accurate metric, though dependent on references.

In this thesis, inspired by the approach in Kann et al. (2018), we develop a novel referenceless metric for data-to-text NLG. Since, in our task, adequacy is reflected by the slot realization accuracy, we can use our proposed slot aligner to replace ROUGE-L with the slot error rate (SER) metric, which we calculate based on the generated utterance and input MR only, i.e., without an explicit reference. In order to make the metric even more generalizable, we experiment with swapping the slot aligner for a neural metric, such as BERTScore or BLEURT, which we use to score utterance candidates against pseudo-references automatically generated from the corresponding MRs. Although we use the referenceless metric to guide MCTS in our proposed inference method, it may find application in other tasks, including automatic evaluation of data-to-text NLG systems.

1.5 Contributions

This thesis makes several novel contributions to the field. We systematically explore the extent to which corpus manipulation and augmentation can control style in data-to-text NLG. Our experiments with training a model on specific stylistic partitions of a large dataset, such as E2E, show that the model does learn more advanced discourse phenomena (such as contrast or fronting). However, it avoids using them altogether in the utterances it generates, when trained on a larger corpus where the phenomena occur overall less frequently. We therefore propose a method of automatically labeling the style variants during training by augmenting the input MRs, and show that we can then successfully enforce the use of particular

discourse phenomena in the generated utterances using our stylistic labels in the inputs. Nevertheless, this stylistic control comes at the cost of semantic accuracy.

This motivates us to introduce a general tool, which we call the slot aligner, that can automatically find a semantic alignment between the slots in the input MR and their mentions in the corresponding utterance. The alignment information can be used for various purposes, including augmenting the training data, automatically evaluating the semantic accuracy of generated utterances, and ranking a pool of utterance candidates a model produces. The slot aligner can be customized to new domains, while taking advantage of the commonalities with other specific domain corpora.

As an alternative to the slot aligner for the task of utterance ranking, we develop a semantically attention-guided decoding method (SEA-GUIDE) for neural encoder-decoder models. This novel approach utilizes the model's own knowledge acquired from training in a way that enables it to automatically track the semantic accuracy during inference and rerank generated utterance candidates accordingly. SEA-GUIDE is domain-independent and requires no model or training data modifications, and can thus be used out of the box. We show on multiple datasets that it has an ability to dramatically reduce semantic errors in model outputs, while maintaining their overall quality and fluency.

Finally, we systematically explore MCTS as a way to simultaneously optimize both semantic accuracy and stylistic diversity during inference. We show that standard inference methods, such as beam search or sampling, achieve a high accuracy while reducing the diversity, or vice versa. Our proposed Batch-MCTS method combines sampling and informed tree search to find varied utterance candidates that are, nevertheless, fluent and semantically correct. It takes advantage of

parallelization to explore as many candidates in as short of a time as possible, so the performance of this inference method scales with the hardware performance. Batch-MCTS depends on a strong utterance evaluation metric to guide its search. For this purpose, we propose a new referenceless automatic metric that assesses the quality of an utterance based solely on the corresponding input MR and the utterance itself. The metric uses a pretrained language model and an existing reference-based neural metric, such as BERTScore or BLEURT, to compare an utterance against a pseudo-reference created from its corresponding MR in an automated fashion. As a result, the metric requires no training and can readily be used for other tasks where texts need to be evaluated, given a structured input, without access to human-written utterances.

In addition to these contributions, we also created and released a new parallel corpus, ViGGO, with more varied dialogue acts and more conversational utterances than previous corpora (Juraska et al., 2019). The dataset consists of 6,900 utterances spanning 9 dialogue act types that could support a dialogue system chatting about and recommending video games, a domain previously unexplored in the fields of data-to-text NLG and dialogue systems. While the ViGGO corpus is crowdsourced, we put significant effort into making it clean by correcting both syntactic and semantic errors, so that the data could serve for effective training of neural NLG models despite its relatively small size. Since we collected three reference utterances for each MR, automatic evaluation of models trained on the ViGGO corpus should also be more accurate than on datasets with a single reference per input. We use ViGGO in most of our experiments, as it enables us to evaluate our proposed methods more broadly.

1.6 Overview of the Thesis

In this chapter we have summarized related work on both stylistic diversity and semantic accuracy in order to motivate our own work. We will also discuss additional related work in more detail throughout the thesis where it is relevant.

In Chapter 2 we describe in detail the three datasets that we use for experimentation: (1) the E2E dataset that was used in the E2E NLG Challenge (Novikova et al., 2016, 2017b), (2) the MultiWOZ dataset which has been broadly used in data-to-text generation (Budzianowski et al., 2018; Eric et al., 2020; Zang et al., 2020), and (3) ViGGO, the dataset that we developed to address some of the limitations of the other datasets (Juraska et al., 2019).

Chapter 3 introduces our heuristic slot aligner tool, which we use extensively for various tasks throughout the entire thesis, ranging from semantic accuracy evaluation, to candidate utterance reranking, to training data augmentation.

In Chapter 4 we review the deep learning sequence-to-sequence models that we train for the data-to-text generation task on the datasets introduced in Chapter 2. We show how methods for adaptive delexicalization can improve the performance of such models, as well as the benefits of transfer learning for models trained on small datasets.

In Chapter 5 we discuss the goal of generating diverse utterances in NLG for dialogue systems, and explore various methods for stylistic control of generated utterances, such as corpus manipulation and token supervision.

In Chapter 6 we dive deeper into the transformer-based encoder-decoder models that we use, and show that attentional state of the models indicates that they are actually aware of whether the output is semantically accurate. We propose

a novel method that utilizes the attention mechanism in a way that tracks slot realizations during inference, which we call semantically attention-guided decoding (SEA-GUIDE), and show that it can dramatically improve semantic accuracy.

In Chapter 7 we explore the use of Monte-Carlo Tree Search (MCTS) for NLG inference, as an alternative to standard methods like greedy search, beam search or nucleus sampling. We discuss the potential of an MCTS-based approach to generate diverse outputs, while maintaining or improving faithfulness to their inputs. We describe in detail our proposed modification, Batch-MCTS, that takes advantage of parallelization in order to make the inference feasible in real time, and explain the need for a referenceless automatic evaluation metric to guide the search algorithm.

Thus in Chapter 8, we propose using standard off-the-shelf automatic metrics such as BLEU, BERTScore and BLEURT with pseudo-references generated automatically from MRs, as a referenceless approach to estimating semantic accuracy of generated utterances. We systematically explore the interaction of these automatic metrics with a number of different pseudo-reference types to determine both the best metric and the most effective pseudo-reference format. We show that the neural metrics BERTScore and BLEURT are the most robust in this setting. We then use these results to define two versions of a referenceless metric that can rank generated utterances based on their quality. It consists of two components: a general-purpose language model to evaluate fluency, combined with either BERTscore/BLEURT calculated using pseudo-references, or SER calculated using our slot aligner from Chapter 3.

In Chapter 9 we then use this referenceless metric to guide the Batch-MCTS inference introduced in Chapter 7. We evaluate our experiments using a set

of diversity metrics and other automatic metrics, as well as a human evaluation. The results show that the trade-off between diversity and semantic accuracy is very much present even in the Batch-MCTS method, but to a lesser degree, allowing this inference method to produce utterances with semantic accuracy on par with standard inference methods yet significantly higher diversity, or vice versa.

Finally, Chapter 10 summarizes our findings, discusses the limitations of our work and describes future work.

Chapter 2

Data-to-Text NLG Corpora

Deep learning models, regardless of the specific task they are applied to, depend on large amounts of data they can be trained on. Data collection for NLG models is often particularly time-consuming and expensive because it requires humans to label text and, for tasks like data-to-text generation, to actually write text. Therefore, the selection of publicly available corpora has been limited despite the widespread adoption of deep learning methods in NLG over the past few years.

Thanks to their unprecedented size in data-to-text NLG, the **E2E** (Novikova et al., 2017b) and the **WebNLG** (Gardent et al., 2017) dataset have become the most popular benchmark datasets for new models. Aside from their domains, the primary difference between these two parallel corpora lies in the representation of the input data. While the E2E dataset maps structured MRs onto natural-language utterances, WebNLG uses lists of RDF triples extracted from DBpedia instead of MRs. Since in our work we focus exclusively on language generation from MRs, the E2E dataset serves as the primary resource for training and evaluating our own models.

Although the E2E restaurant dataset is particularly well-suited for bench-

	TV	Laptop	E2E	MultiWOZ	ViGGO
|Training set|	4,221	7,944	42,061	55,951	5,103
|Validation set|	1,407	2,649	4,672	7,286	714
|Test set|	1,407	2,649	4,693	7,293	1,083
Total size	7,035	13,242	51,426	70,530	6,900
Domains	1	1	1	7	1
DA types	14	14	1	13	9
Slot types	16	20	8	27	14

Table 2.1: Dataset partition sizes and other dataset statistics, including the total number of dialogue act (DA) and slot types. For MultiWOZ, the numbers are calculated across system turns only.

marking new models against others, since many of the state-of-the-art models have been evaluated on it, it contains data from a single domain and uses just one dialogue act (DA) type. **MultiWOZ 2.1** (Eric et al., 2020) is a corpus of 10K human-human written conversations covering several domains, fully annotated with task-oriented dialogue systems in mind, and featuring thus a number of DAs. For the purposes of data-to-text generation, we extract system turns only, together with their MR annotations, along the lines of Peng et al. (2020) and Kale and Rastogi (2020). This results in a parallel corpus larger than E2E, and spanning 7 domains. We use MultiWOZ mainly for benchmarking purposes, to compare some of our proposed methods against state-of-the-art models that were trained and evaluated on this dataset.

In addition to E2E and MultiWOZ, we conducted our earlier experiments on the RNNLG toolkit's **TV** and **Laptop** datasets (Wen et al., 2016). Being significantly smaller than E2E, we made use of them primarily when evaluating our models against pre-E2E models, and we thus do not report results on these two datasets in our evaluation.

In order to be able to better evaluate our automatic slot aligner (Chapter 3) and for our domain adaptation experiments (Chapter 4), we also collect a new MR-

to-text corpus in the video game domain. A more detailed introduction of the dataset, **ViGGO**, along with its analysis, follows in Section 2.2. Before that, in Section 2.1, we describe the E2E dataset in more detail, as it will serve, together with ViGGO, as one of the primary datasets we evaluate our models on throughout the thesis. Table 2.1 summarizes the proportions of the training, validation, and test sets for all the datasets introduced above, along with additional statistics.

2.1 E2E Dataset

The E2E dataset was, until recently, the largest publicly available dataset for task-oriented language generation in the restaurant domain, now surpassed by YelpNLG (Oraby et al., 2019). With 50K examples, it offers almost 10 times more data than the San Francisco restaurant dataset introduced in Wen et al. (2015b). The reference utterances were crowdsourced mostly from the MRs themselves, though a portion was collected using pictorial representations, instead of the MRs, as the source of information for the crowdworkers. The latter was shown to inspire more natural utterances compared to textual MRs (Novikova et al., 2016), however, it introduced substantial noise in the form of missing and incorrect slot mentions in the utterances, which in turn has a negative impact on the performance of deep learning models trained on such data, as shown in Dušek et al. (2019). Nevertheless, overall the reference utterances in the E2E dataset exhibit superior lexical richness and syntactic variation, including more complex discourse relations. It aims to provide higher-quality training data for end-to-end NLG models to learn to produce more naturally sounding utterances. The dataset was released as part of the E2E NLG Challenge shared task.[1]

[1] https://www.macs.hw.ac.uk/InteractionLab/E2E/

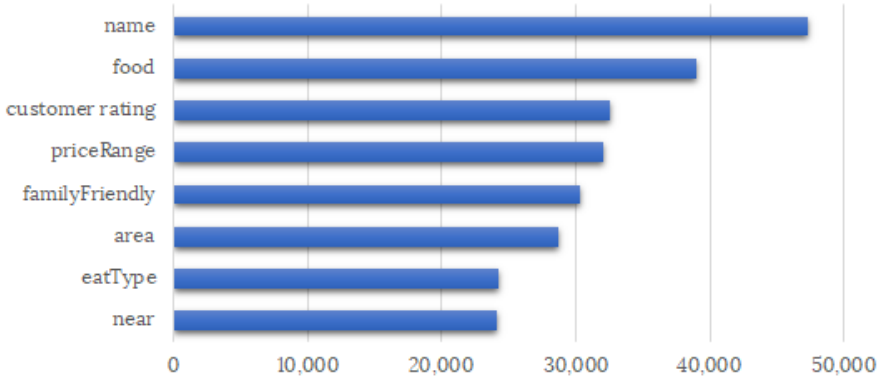


Figure 2.1: Distribution of slots in the E2E dataset.

Slots	3	4	5	6	7	8
Sentences	1.09	1.23	1.41	1.65	1.84	1.92
Proportion	5%	18%	32%	28%	14%	3%

Table 2.2: Average number of sentences in the reference utterance for a given number of slots in the corresponding MR, along with the proportion of MRs with specific slot counts in the E2E dataset.

The dataset contains only 8 different slot types, which are fairly evenly distributed (see Figure 2.1). The number of slots in each MR ranges between 3 and 8, but the majority of MRs consist of 5 or 6 slots. Even though most of the MRs contain many slots, the majority of the corresponding reference utterances, however, consist of one or two sentences only (Table 2.2), suggesting a reasonably high level of sentence complexity in the references.

Although the E2E dataset contains a large number of examples, each MR is associated on average with more than 8 different reference utterances. This means that the number of unique MRs in the training set is less than 5K. On the other hand, by explicitly providing multiple human-authored utterances for each MR, the dataset offers an NLG model examples of various possible ways of expressing the same information represented by an MR (see Table 1.1). In addition to that, multiple ground truths per MR play an important role in making automatic evaluation of a

confirm (NAME [**Hellblade: Senua's Sacrifice**], RELEASE_YEAR [**2017**], DEVELOPER [**Ninja Theory**])
Oh, do you mean the **2017** game from **Ninja Theory**, **Hellblade: Senua's Sacrifice**?
give_opinion (NAME [**SpellForce 3**], RATING [**poor**], GENRES [**real-time strategy, role-playing**], PLAYER_PERSPECTIVE [**bird view**])
I think that **SpellForce 3** is **one of the worst games** I've ever played. Trying to combine the **real-time strategy** and **role-playing** genres just doesn't work, and the **bird's eye view** makes it near impossible to play.
verify_attribute (NAME [**Little Big Adventure**], RATING [**average**], HAS_MULTIPLAYER [**no**], PLATFORMS [**PlayStation**])
I recall that you were **not that fond** of **Little Big Adventure**. Does **single-player** gaming on the **PlayStation** quickly get boring for you?

Table 2.3: Examples of MRs and corresponding reference utterances in the ViGGO dataset. The DA of the MRs is indicated in italics, and the slots in small caps. The slot mentions in the utterances are bolded.

model's outputs more objective, as different utterance structures generated by the model can thus correctly be accepted as good realizations of the given MR.

The popularity of the E2E dataset has since its release spawned new variants of the dataset, such as one augmented with utterances in different personality styles (Reed et al., 2018), or a cleaner version with the noise from the data collection removed (Dušek et al., 2019).

2.2 ViGGO: A Conversational Data-to-Text Corpus

Due to our dissatisfaction with existing corpora, we created a new data-to-text corpus, different from the existing resources. ViGGO is a smaller but more comprehensive dataset in the video game domain, introducing several generalizable DAs, making it more suitable for training versatile and more conversational NLG models.[2] The dataset provides almost 7K pairs of MRs and human-authored utterances about more than 100 video games. Table 2.3 lists three examples.

[2]The ViGGO corpus is available for download at: https://nlds.soe.ucsc.edu/viggo

Video games are a vast entertainment topic that can naturally be discussed in a casual conversation, similar to movies and music, yet in the dialogue systems community it does not enjoy popularity anywhere close to that of the latter two topics (Fazel-Zarandi et al., 2017; Li et al., 2017b; Moghe et al., 2018; Shah et al., 2018; Khatri et al., 2018). Restaurants have served as the go-to topic in data-to-text NLG for decades, as they offer a sufficiently large set of various attributes and corresponding values to talk about. While they certainly can be a topic of a casual conversation, the existing restaurant datasets (Stent et al., 2004; Gašić et al., 2008; Mairesse et al., 2010; Howcroft et al., 2013; Wen et al., 2015a; Nayak et al., 2017) are geared more toward a task-oriented dialogue where a system tries to narrow down a restaurant based on the user's preferences and ultimately give a recommendation. Our new video game dataset is designed to be more conversational, and to thus enable neural models to produce utterances more suitable for a multi-domain dialogue system.

Even the most recent additions to the publicly available restaurant datasets for data-to-text NLG, the E2E dataset described in Section 2.1 and YelpNLG, both suffer from the lack of a conversational aspect. E2E became popular, thanks to its unprecedented size and multiple reference utterances per MR, for training end-to-end neural models, yet it only provides a single DA type. And so does YelpNLG. In contrast with these two datasets, ViGGO presents utterances of 9 different DAs.

Other domains have been represented by task-oriented datasets with multiple DA types, for example the Hotel, Laptop, and TV datasets (Wen et al., 2015b, 2016). Nevertheless, the DAs in these datasets vary greatly in complexity, and their distribution is thus heavily skewed, typically with two or three similar DAs comprising almost the entire dataset. In our video game dataset, we omitted simple DAs,

DA	Slot range	Mandatory slots	Additional common slots
inform	3–8	NAME, GENRES	RELEASE_YEAR, DEVELOPER, ESRB, GENRES, PLAYER_PERSPECTIVE, HAS_MULTIPLAYER, PLATFORMS, AVAILABLE_ON_STEAM, HAS_LINUX_RELEASE, HAS_MAC_RELEASE
confirm	2–3	NAME	
give_opinion	3–4	NAME, RATING	
recommend	2–3	NAME	
request	1–2	SPECIFIER	
request_attribute	1		
request_explanation	2–3	RATING	
suggest	2–3	NAME	
verify_attribute	3–4	NAME, RATING	

Table 2.4: Overview of mandatory and common possible slots for each DA in the ViGGO dataset. There is an additional slot, EXP_RELEASE_DATE, only possible in the *inform* and *confirm* DAs. Moreover, RATING is also possible in the *inform* DA, though not mandatory.

in particular those that do not require any slots, such as greetings or short prompts, and focused on a set of substantial DAs only.

2.2.1 Dataset Overview

ViGGO features more than 100 different video game titles, whose attributes were harvested using free API access to two of the largest online video game databases: IGDB[3] and GiantBomb[4]. Using these attributes, we generated a set of 2,300 structured MRs. The human reference utterances for the generated MRs were then crowdsourced using vetted workers on the Amazon Mechanical Turk (MTurk) platform (Buhrmester et al., 2011), resulting in 6,900 MR-utterance pairs altogether. With the goal of creating a clean, high-quality dataset, we strived to obtain reference utterances with correct mentions of all slots in the corresponding MR through post-processing.

[3] https://www.igdb.com/
[4] https://www.giantbomb.com/

Meaning representations. The MRs in the ViGGO dataset range from 1 to 8 slot-value pairs, and the slots come from a set of 14 different video game attributes. Table 2.4 details how these slots may be distributed across the 9 different DAs. The *inform* DA, represented by 3,000 samples, is the most prevalent one, as the average number of slots it contains is significantly higher than that of all the other DAs. Figure 2.2 visualizes the MR length distribution across the entire dataset. The slots can be classified into 5 general categories:

1. **Boolean** – binary value, such as "yes"/"no" or "true"/"false" (e.g., HAS_MULTIPLAYER),
2. **Numeric** – value is a number or contains number(s) as the salient part (e.g., RELEASE_YEAR),
3. **Scalar** – values are on a distinct scale (e.g., RATING or ESRB),
4. **Categorical** – takes on virtually any value, typically coming from a certain category, such as names or types (e.g., NAME or DEVELOPER),
5. **List** – similar to categorical, where the value can, however, consist of multiple individual items (e.g., GENRES or PLAYER_PERSPECTIVE).

The first 4 categories are common in other NLG datasets, such as E2E, Laptop, TV, and Hotel, while the list slots are unique to ViGGO. There are no restrictions as to whether the values are single-word or multi-word in any of the categories.

Utterances. With neural language generation in mind, we crowdsourced 3 reference utterances for each MR so as to provide the models with the information about how the same content can be realized in multiple different ways. As we argued earlier, this also allows for a more reliable automatic evaluation by comparing the generated utterances with a set of different references each, covering a broader

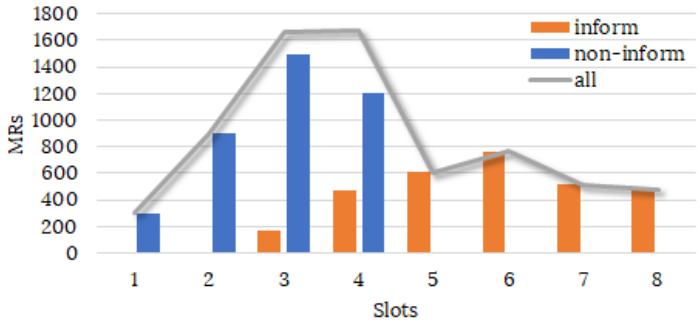


Figure 2.2: Distribution of the number of slots across all types of MRs, as well as the *inform* DA separately and non-*inform* DAs only.

spectrum of correct ways of expressing the content given by the MR. The raw data, however, contains a significant amount of noise, as is inevitable when crowdsourcing. We therefore created and enforced a robust set of heuristics and regular expressions to account for typos, grammatical errors, undesirable abbreviations, and unsolicited information. Moreover, using the automatic slot aligner described in Chapter 3 we fixed most of the missing and incorrect slot realizations too.

2.2.2 Data Collection

The crowdsourcing of utterances on MTurk took place in three stages. After collecting one third of the utterances, we identified a pool of almost 30 workers who wrote the most diverse and natural-sounding sentences in the context of video games. We then filtered out all utterances of poor quality and had the qualified workers write new ones for the corresponding inputs. Finally, the remaining two thirds of utterances were completed by these workers exclusively.

For each DA we created a separate task in order to minimize the workers' confusion. The instructions contained several different examples, as well as counter-examples, and they situated the DA in the context of a hypothetical conversation. The video game attributes to be used were provided for the workers in the form of

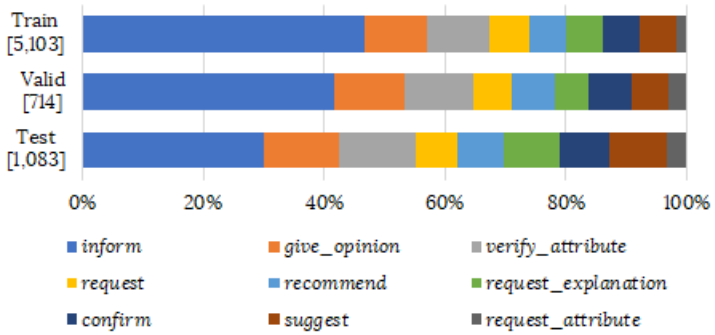


Figure 2.3: Distribution of the DAs across the training/validation/test split. For each partition the total number of examples is indicated.

a table, with their order shuffled so as to avoid any kind of bias.

2.2.3 Training/Validation/Test Split

Despite the fact that the ViGGO dataset is not very large, we strived to make the test set reasonably challenging. To this end, we ensured that, after delexicalizing the NAME and the DEVELOPER slots, there were no common MRs between the train set and either of the validation or test set. We maintained a similar MR length and slot distribution across the three partitions. The distribution of DA types, on the other hand, is skewed slightly toward fewer *inform* DA instances and a higher proportion of the less prevalent DAs in the validation and test sets (see Figure 2.3). The final ratio of examples is approximately 7.5 : 1 : 1.5, with the exact partition sizes indicated in the diagram.

2.3 ViGGO vs. E2E

Our new dataset was constructed under different constraints than the E2E dataset. First, in ViGGO we did not allow any omissions of slot mentions, as those are typically undesirable in data-to-text generation with no previous context as

	Size	Unique MRs	Unique delex. MRs	Vocab	Delex. vocab	Avg. 3-gram freq.	Ref/ MR	Slots/ MR	W/ Ref	W/ Sent	S/ Ref
E2E	51,426	6,039	5,963	2,878	2,818	18.70	8.1	5.43	22.41	14.36	1.56
ViGGO$_{inf}$	3,000	1,000	997	1,378	1,102	8.33	3	5.81	30.62	15.01	2.04
ViGGO	6,900	2,253	2,066	2,427	2,178	6.91	3	4.18	25.01	15.04	1.66

Table 2.5: Dataset statistics comparing the ViGGO dataset, as well as its subset of *inform* DAs only (ViGGO$_{inf}$), with the E2E dataset. The average trigram frequency was calculated on trigrams that appear more than once. "Ref" stands for references, "W" for words, and "S" for sentences.

input, and they make the evaluation ambiguous. Second, the MRs in ViGGO are grounded by real video game data, which can encourage richer and more natural-sounding reference utterances.

While ViGGO is only 13% the size of the E2E dataset, the lexical diversity of its utterances is 77% of that in the E2E dataset, as indicated by the "delexicalized vocabulary" column in Table 2.5. Part of the reason naturally is the presence of additional DAs in ViGGO, and therefore we also indicate the statistics in Table 2.5 for the *inform* examples only. The average *inform* utterance length in ViGGO turns out to be over 30% greater, in terms of both words and sentences per utterance.

Finally, we note that, unlike the E2E dataset, our test set does not place any specific emphasis on longer MRs. While the average number of slots per MR in the *inform* DAs are comparable to the E2E dataset, in general the video game MRs are significantly shorter. This is by design, as shorter, more focused responses are more conversational than consistently dense utterances.

2.4 Discussion

In Table 2.6 we demonstrate how the 9 DAs of the ViGGO dataset can support a natural multi-turn exchange on the topic of video games, as part of a longer

S:	. . . What about video games?
U:	Totally, I spend my entire evenings in front of my PS4.
S:	What is your favorite **genre** of video games? *request_attribute* (GENRES [])
U:	I guess I play a lot of shooter games.
S:	Have you played any **interesting** games lately? *request* (SPECIFIER [**interesting**])
U:	Tomb Raider is pretty good.
S:	Do you mean **Tomb Raider** from **2013**? *confirm* (NAME [**Tomb Raider**], RELEASE_YEAR [**2013**])
U:	Yes, that one with the epic E3 trailer.
S:	Do you also enjoy playing **first-person shooters**, such as **Half-Life 2**? *suggest* (NAME [**Half-Life 2**], GENRES [**shooter**], PLAYER_PERSPECTIVE [**first person**])
U:	I haven't played that one in a while. Would you recommend something else?
S:	Well, since you seem to be a fan of **M-rated PlayStation** games, I thought you would perhaps like **Max Payne 3**. Have you heard about that game? *recommend* (NAME [**Max Payne 3**], ESRB [**M (for Mature)**], PLATFORMS [**PlayStation**])
U:	No.
S:	**Max Payne 3** is a **multiplayer action-adventure shooter** from **2012**. *inform* (NAME [**Max Payne 3**], RELEASE_YEAR [**2012**], GENRES [**action-adventure; shooter**], MULTIPLAYER [**yes**])
U:	I see. So, you enjoyed it?
S:	Oh, I **love Max Payne 3**. When I see a game developed by **Rockstar Games**, I buy it without thinking. *give_opinion* (NAME [**Max Payne**], RATING [**excellent**], DEVELOPER [**Rockstar Games**])
U:	That sounds cool.
S:	Previously you told me that you **enjoyed Tomb Raider**. Is that the case with all games developed by **Crystal Dynamics**? *verify_attribute* (NAME [**Tomb Raider**], RATING [**good**], DEVELOPER [**Crystal Dynamics**])
U:	You bet.
S:	What is it that you **like** about **Crystal Dynamics** games? *request_explanation* (RATING [**good**], DEVELOPER [**Crystal Dynamics**])
U:	They usually have really good stories.
S:	. . .

Table 2.6: Example of a conversation about video games comprising utterances of DAs defined in ViGGO. "S" and "U" denote the system and user turns, respectively.

casual conversation on different topics. One caveat of using a language generator trained on this dataset in a dialogue system as-is is that multiple subsequent turns discussing the same video game would be repeating its full name. ViGGO was designed for grounded generation but without context, and therefore it is up to the dialogue manager to ensure that pronouns are substituted for the names whenever it would sound more natural in a dialogue. Alternately, the dataset can easily be augmented with automatically constructed samples which omit the NAME slot in the MR and replace the name with a pronoun in the reference utterance.

2.5 Summary

In this chapter, we provided an overview of existing data-to-text NLG corpora that we use throughout the thesis to train and test our models. We also presented a new parallel corpus for data-to-text NLG, ViGGO, which contains 9 dialogue acts, making it more conversational, rather than information seeking or question answering, and thus more suitable for a multi-domain dialogue system. The crowdsourced utterances were thoroughly cleaned in order to obtain high-quality human references. ViGGO and E2E will be our primary datasets in the thesis.

Chapter 3

Automatic Semantic Aligning

Models that perform data-to-text language generation using deep learning methods often find it challenging to learn exactly which part of the utterance they generate corresponds to a specific slot in the MR. As a consequence, it is not uncommon to see generated utterances with missing or duplicate information. Slots can be of different types, they can take on *many values* that often may be realized in *multiple synonymous ways*. A standard sequence-to-sequence neural model needs to encounter a large set of possible slot combinations and their realizations in reference utterances to be able to understand these variations and the associations in general.

The problem of many values typically concerns slots representing names, numbers/years, or other pieces of information that always get realized verbatim in the utterance and have no equivalent alternatives. The standard approach to handling this, as well as handling out-of-vocabulary (OOV) issues during inference, is through slot delexicalization (Mairesse et al., 2010; Henderson et al., 2014) or, more recently, through a copy mechanism (Vinyals et al., 2015; Gu et al., 2016; See et al., 2017). While the former is performed as a part of the pre- and post-processing stages, the latter is integrated directly into the model.

On the other hand, we address the possibility of multiple synonymous slot realizations by developing a domain-agnostic slot aligner. It serves to indirectly improve the performance of our neural model's capability to learn the correct slot realization rules, as well as to provide backup when the neural model itself produces a sub-optimal solution. To lift some of the burden off the neural model, we use the slot aligner to denoise the training samples, augment the training set with additional samples, rerank the top candidates generated by the model, and identify the most complete utterances.

In this chapter, we describe our slot aligner in detail, including the tasks it can be applied to. We defer the evaluation of its performance on these tasks to Section 4.5, after the description of our deep learning models which make extensive use of the slot aligner. There we assess the slot aligner's accuracy and benefits on the E2E dataset, as well as the newly collected ViGGO dataset, in order to demonstrate the slot aligner's scalability and robustness.

3.1 Heuristic Slot Aligner

For the purposes of the slot aligner, we classify slots into the same 5 general categories as are present in the E2E and the ViGGO dataset (i.e., *Boolean*, *numeric*, *scalar*, *categorical*, and *list*), covering most types of information MRs typically convey in data-to-text generation scenarios. Each of these categories has its own method for extracting a slot mention from an utterance, generalized enough to be applicable across all slots in the category. One major advantage of this design is that, whenever the NLG system is to be used in a new domain, the slot aligner merely needs to be indicated which of the five categories each of the slots in this

domain belongs to. Optionally, it can be supplied with a simple dictionary of common alternatives for specific slot values, which tends to increase the slot aligner's performance.

3.1.1 Boolean Slots

Boolean slots take on binary values, such as "yes"/"no" or "true"/"false". Their mention in an utterance thus typically does not contain the actual value of the slot, but instead a mention of the slot itself (e.g., "is a family-friendly restaurant" for FAMILYFRIENDLY[yes], or "not supported on Mac" for HAS_MAC_RELEASE[no]). Therefore, extracting a Boolean slot realization boils down to the following two steps: (1) finding a word or a phrase representing the slot, and (2) verifying whether the representation is associated with a negation or not.

The first step is straightforward, and it only requires a list of possible realizations for each Boolean slot. This list rarely contains more than one element, which is the "stem" of the slot's name (e.g., "linux" for "HAS_LINUX_RELEASE"). It can thus be populated trivially for most of the new Boolean slots. And if a Boolean slot can have multiple equivalent realizations (such as "child friendly" or "where kids are welcome" for the slot FAMILYFRIENDLY), they are typically not numerous and can be listed manually. Having a list of *stems* (we refer to all the equivalent realizations of a slot collectively as "slot stems"), the utterance is scanned for the presence of each of them in it. If one is found, we go to the second step.

A slot mention is decided to be negative if a negation cue is found to be modifying the slot stem, i.e., within a certain distance of the stem and without a contrastive cue in between, and it is decided to be positive if no negation cue is present or there is a contrastive cue in between.

#1	There's *no* Linux release or multiplayer, **but** there is Mac support.
#2	**Though** it's *not available* on Linux, it does have a Mac release as well.
#3	It is available on PC and Mac **but** *not* Linux, and it can be found on Steam.

Table 3.1: Examples of contrastive phrases involving Boolean slots. Underlined are the stems of the Boolean slots for which the polarity is questioned. Note that in all 3 examples the mention is positive, despite the presence of contrast and negation distractors.

3.1.2 Numeric Slots

Slots whose value is just a number are in general not handled in any special way, and the value is matched directly in the utterance. However, there are certain numeric slot types that benefit from additional preprocessing: (1) years, and (2) numbers with a unit (e.g., POWER_CONSUMPTION or SCREEN_SIZE in the TV dataset).

When a numeric slot represents a year, the slot aligner generates the common abbreviated alternatives for the year (e.g., "'97" for the value "1997") that it tries to match in case the original value is not found in the utterance. The only thing the slot aligner does differently for numeric slots with a unit than for simple numeric slots is that it strips away the unit (such as "watts" or "inches") and keeps the numeric value only, which it searches for in the utterance. Although this might appear as discarding essential information, in NLG for dialogue systems the utterances are only up to a few sentences long, and thus, it is unlikely that there would be two slots with the same numeric value in one MR, only differing in their units. Stripping the unit away, on the other hand, relieves the slot aligner of attempting to match the unit with its alternate expressions, such as abbreviations; which, in turn, relieves the user from defining unit equivalents for new domains.

3.1.3 Scalar Slots

Similar to Boolean slot aligning, scalar slot aligning consists of two steps. The first one is the same, i.e., finding a word or a phrase representing the slot (which we refer to as "stem" in this case too, in order to maintain consistency). In the second step, however, the slot aligner looks for the slot's value, or its equivalent, occurring within a reasonable distance from the slot stem. The optional soft alignment mode skips the second step as long as a slot stem is matched in the first step.

We assume that scalar slots, even across different domains, will often have values that can be mapped to each other, as long as they are on the same or a similar scale (see Table 3.2). For each scalar slot the slot aligner refers to a corresponding dictionary for possible alternate expressions of its value. With the above assumption, it is sufficient to have one dictionary per scale, or type of scale, which can be reused for similar scalar slots in different domains. The dictionaries can be quickly populated with synonyms of the values of a given scale (see the last column in Table 3.2), and thus do not necessarily require manual additions every time the system is used with a new domain. Some alternate expressions might be suitable for scalar slots in some domains better than others, but that will not be an issue in most cases, since they are not likely to cause conflicts (being synonymous), and the slot aligner will simply not encounter certain alternate expressions in certain domains.

An example of a scalar slot whose values are on a 3- or 4-point scale, but cannot be mapped to those of the video game RATING slot for the purposes of sharing the alternate expressions, is PRICERANGE, which, for instance, in the E2E restaurant domain takes on values "cheap", "moderate" and "high". These could, obviously,

customerRating (restaurant)	rating (video game)	Alternate expressions
low	poor	*bad, lacking, negative,...*
average	average	*decent, mediocre, okay,...*
-	good	*fun, positive, solid,...*
high	excellent	*amazing, fantastic, great,...*

Table 3.2: Example of value mapping between two similar scalar slots in the E2E restaurant domain and the ViGGO video game domain.

not be mapped to the alternate expressions in Table 3.2, and a separate dictionary would have to be created for it. This dictionary could then, however, be reused for similar price-related slots in other domains, such as hotel, laptop, or booking of a flight.

Although the slot aligner attempts to find the scalar slot's value in close proximity to the slot stem, it is often the case that the stem is not mentioned in the utterance at all (such as when the mention of the slot-value pair priceRange[cheap] in the E2E dataset is "a cheap restaurant", without the mention of "price"). For this scenario, the slot aligner remembers the leftmost match of any of the value's alternate expressions in the utterance, and uses it as a fallback when deciding about the value's mention without the occurrence of the slot stem.

3.1.4 Categorical Slots

Categorical slots can take on virtually any value, nevertheless, for each such slot the values typically come from a limited, although possibly large, set of values. For instance, in the E2E dataset, the food slot has seven possible values, such as "Italian" and "Fast food", but technically it could take on hundreds of different values representing all of the cuisines of the world. Some values can be single-word, while others can have multiple words (e.g., "restaurant" and "coffee

shop" as possible values for the EATTYPE slot). Due to this huge variety in possible values of categorical slots, the aligning methods need to remain very general.

Besides exact matching of the value in the utterance, the slot aligner can be instructed to perform the matching in three additional modes, increasing its robustness while maintaining scalability. The four modes of aligning the slot with its mention work as follows:

- **Exact** – slot mention is identified only if it matches (case-insensitive) the slot value verbatim,
- **All words** – slot mention is identified if each of the value's tokens is found in the utterance, though they can be in an arbitrary order and they can be separated by other words,
- **Any word** – slot mention is identified by matching any of the value's tokens in the utterance,
- **First word** – slot mention is identified by matching just the value's first token in the utterance.

Note that for single-word values all four modes give the same result. The three non-exact modes offer different approaches to soft alignment for categorical slots. The choice may depend on the particular slot, and the mode can thus be specified for each slot separately, while by default the slot aligner operates in the exact-matching mode.

Similar to Boolean and scalar slots, the slot aligner can search for alternate expressions of a value, if provided in the corresponding dictionary. The alternate matching is, however, more flexible here, as the alternatives in the dictionary can be multi-part, in which case the slot aligner tries to match all the parts

MR
inform(NAME [**BioShock**], DEVELOPER [**2K Boston**], GENRES [**action-adventure, role-playing, shooter**], HAS_MULTIPLAYER [**no**], PLATFORMS [**PlayStation, Xbox, PC**], HAS_LINUX_RELEASE [**no**], HAS_MAC_RELEASE [**yes**])
Reference utterance
Developed by **2K Boston**, **BioShock** is a **single-player shooter** game that will have you **role-playing** through a well constructed **action-adventure** narrative. It is **available for PlayStation**, **Xbox**, **Mac** and **PC**, but is **not available for Linux**.
Slot alignment
(**13:** DEVELOPER) (**25:** NAME) (**39:** HAS_MULTIPLAYER) (**53:** GENRES) (**174:** PLATFORMS) (**191:** HAS_MAC_RELEASE) (**228:** HAS_LINUX_RELEASE)

Table 3.3: Example from the ViGGO dataset that involves list slots. Notice how the individual value item mentions can be scattered across an entire sentence in a natural way. The bottom section indicates the determined slot alignment between the utterance and the MR. The position of a slot mention is given as the number of characters from the beginning of the utterance.

(words/tokens/phrases) provided in the form of a list.

3.1.5 List Slots

A list slot is similar to a categorical slot, the only difference being that it can have multiple individual items in its value. Two examples of a list slot, namely GENRES and PLATFORMS, can be seen in the sample from the ViGGO dataset in Table 3.3. Note that this slot type is not represented in the E2E dataset.

The aligning procedure for list slots thus heavily relies on that of categorical slots. In order to align a list slot with the corresponding utterance, the slot aligner first parses the individual items in the slot's value. It then iterates over all of them and performs the categorical slot alignment, as described in the previous section, with each individual item. Considering the items can be scattered over multiple sentences, the slot aligner considers the position of the leftmost mention of an item as the position of the corresponding list slot.

List slots allow for an additional level of soft alignment on top of the non-exact matching modes defined for categorical slots. The slot aligner can be parameterized so that not all items in the value need to be matched in the utterance in order for a list slot to be aligned with its mention. This, combined with the categorical slot matching modes, as well as the simple but robust definition of the alternate expression dictionary, make even the addition of support for a new list slot straightforward, with plenty of flexibility and without compromising on scalability.

3.2 Aligning Tasks

The slot aligner described in Section 3.1 is a multi-purpose tool that finds several different uses throughout the entire NLG pipeline, from the data collection for a new domain to the selection of the best output utterance among multiple candidates. The following sections give examples of different practical tasks for the slot aligner, all of which are integrated into our NLG system described in Chapter 4. These sections also give insight into the challenges faced in each of the tasks and how the slot aligner tackles them.

3.2.1 Finding Slot Alignment

The first obvious task the slot aligner is designed to perform is to identify which slot is mentioned at which position in the utterance (see an example in Table 3.3). These slot mention position indications can by themselves be used for a general analysis of the dataset. Additionally, it can be useful in identifying discourse phenomena, which in turn can be exploited for training a neural model with increased stylistic variation in general, or with the ability to enforce a certain struc-

ture in the generated utterance. Some example uses would be emphasizing specific slots at the beginning of the utterance, or putting two slot mentions in contrast. We elaborate on this in Chapter 5.

3.2.2 Training Data Denoising

NLG datasets, especially the crowdsourced ones, are notoriously full of typos and missing information in the reference utterances, and thus a neural model can benefit from a significant reduction of such noise. Although our slot aligner works heuristically and, therefore, does not always find a correct alignment, nor does it recognize all possible slot realizations, it is able to catch the vast majority of these errors and clean up the training set by removing the non-aligned slots from the MRs.

3.2.3 Training Data Augmentation

The slot aligner can also be used for augmenting the training data, with the objective of providing additional relevant samples for the model to learn from. With the positional information about slot mentions provided by the aligner, it is straightforward to identify which slots are mentioned in which sentence of the utterance. With the help of a sentence tokenizer, we then create pseudo-samples with the individual sentences as reference utterances, and corresponding MRs containing only the slots mentioned in the individual sentences. When creating new pseudo-samples, we remove the slots whose mention is not found at all by the slot aligner in the utterance.[1] Otherwise, such slots would have to be randomly assigned to one of the new pseudo-samples, possibly causing more noise in the augmented dataset.

[1]To maintain training data consistency, the denoising is applied to both the new pseudo-samples and the original ones.

The motivation for the sample splitting is to provide the model with more granular information about the alignment of the MR's slots with their mentions in the utterance. Our hypothesis is that this might facilitate the model's learning process, wherein it is given a shorter context (utterance) to learn to associate segments of, with slots as their mentions. It should be noted that the slot aligner works in a soft alignment mode when splitting samples, which means that for certain slot types the slot aligner is more lax when matching their mention. This is in contrast with the other aligning tasks, in which the slot aligner expects an exact match of the value or each of the individual words having a match. The soft alignment is meant to help recognizing a match even if the slot mention is not in any of the expected forms, which is more often the case in the human-authored reference utterances than the system-generated ones.

Taking the idea of data augmentation further, a neural model could be made to learn from its own mistakes. To achieve this, we would modify the MRs accordingly whenever the model generates an utterance with a missing slot mention, and use the pair as a new training sample. A similar approach, combining such self-training with noise injection sampling, was used in Kedzie and McKeown (2019) to significantly improve the performance of their otherwise basic sequence-to-sequence model.

3.2.4 Cross-Domain Dataset Replication

For two datasets from two different domains it might be difficult to get any performance improvement on one of them by pretraining the model on the other one, if they are not sufficiently similar. The slot aligner can, however, indirectly help reduce the semantic gap between such two datasets. Following the work of Wen

et al. (2016), we find a mapping of the target-domain slots to the source-domain slots based on their type and semantic similarity, and substitute the former for the latter in the source dataset accordingly. At the same time, we replace the original slot values with sampled target-domain values of the corresponding substituted slots. The slot aligner's dictionary of alternate expressions is utilized here to identify the slot mentions in the source dataset, so the values could be replaced in the utterances as well. Without the corresponding slot mentions in the utterance, though very limited in variation, the replicated dataset would have little value for the neural model. Whenever a slot mention is not found, the original source-domain value of the slot is preserved so as to avoid target-domain noise.

Assuming the source dataset is substantially larger, this process results in a crude but large target-domain dataset replica. Although the majority of the utterances would not make sense at this point, the purpose of the replicated dataset is to provide the neural model with an abundant number of additional training samples from which it can learn that slots from the input need to be realized in the generated utterance. In Table 3.4 we present the chosen slot mapping between the E2E and the ViGGO dataset that we use to replicate a large video game dataset (target) using the E2E dataset (source), as described above. Note that the video game domain has more slots defined than the E2E dataset, therefore, certain E2E slots have multiple video game slots mapped to them.

3.2.5 Utterance Reranking

Utterance reranking is employed to provide additional guidance for the NLG system to decide which among the utterances generated by the neural model for a given input MR is the best. A sequence-to-sequence model can produce mul-

E2E (restaurant)		ViGGO (video game)
NAME	⟶	NAME
FOOD	⟶	RELEASE_YEAR, EXP_RELEASE_DATE
CUSTOMERRATING	⟶	RATING, ESRB
PRICERANGE	⟶	PLAYER_PERSPECTIVE
FAMILYFRIENDLY	⟶	HAS_MULTIPLAYER, AVAILABLE_ON_STEAM, HAS_LINUX_RELEASE, HAS_MAC_RELEASE
AREA	⟶	PLATFORMS
EATTYPE	⟶	GENRES
NEAR	⟶	DEVELOPER

Table 3.4: Mapping of slots from E2E's restaurant domain to ViGGO's video game domain, which we used to create a video game dataset replica out of the E2E dataset.

tiple candidate utterances that it deems best according to the statistical rules and associations learned during the training phase. Nevertheless, although the candidate with the highest probability may be the most fluent, it might be missing one or two slot realizations, or a realization might be incorrect (such as mentioning "excellent" instead of the value "poor" given in the MR). This is where the slot aligner can be of assistance.

The slot aligner scores each candidate utterance, taking the number of missed, incorrect, and hallucinated slots into account. The alignment scores are used as coefficients applied to the candidates' probabilities computed by the model, potentially causing thus a reranking of the candidate utterances. The new top candidate is then selected as the final utterance.

3.2.6 Evaluation of Slot Realization Accuracy

The slot aligning performed in this task is similar to the one in the utterance reranking, with just one difference: the output is not a score but a list of

	$\mathbf{SER_{SA}}$	**SER CI (95%)**	**Precision**	**IAA**
ViGGO	2.77%	$2.19 \pm 1.55\%$	97.37%	1.00
E2E	3.98%	$3.91 \pm 1.73\%$	100%	1.00
MultiWOZ	1.19%	$1.35 \pm 0.91\%$	94.89%	0.90

Table 3.5: Human evaluation of the slot aligner's performance on each dataset. The IAA column indicates the Krippendorff's alpha reliability coefficient.

incorrectly realized slots. This task is used for evaluating the *slot error rate* (SER) of utterances generated by an NLG model. SER is calculated as the proportion of failed slot realizations (i.e., missing, incorrect, or duplicate) out of all slots in the test set. Alternatively, it can be viewed as the weighted average of slot errors per utterance expressed in percentage form. Therefore, unlike most metrics, SER should be minimized, with the ideal value being zero.

3.3 Slot Aligner Evaluation

We put substantial effort into developing a highly accurate heuristic slot aligner to calculate the semantic accuracy of generated utterances. In this section, we evaluate how accurately it performs in practice on three different datasets, each in a different domain: ViGGO (video games), E2E (restaurants), and MultiWOZ (multiple domains).

To verify the slot aligner's performance, we take the generated utterances of one model per dataset for which it determined a relatively high slot error rate (indicated in the SER_{SA} column in Table 3.5). We then have one of the authors and an additional expert annotator manually label all of the errors as true or false positives. This corresponds to 38, 173 and 176 errors for ViGGO, E2E and MultiWOZ, respectively. From that we calculate the precision for each dataset, which turns out

to be above 94% for each of the datasets. The almost perfect inter-annotator agreement (IAA), besides validating the precision, also suggests that SER is an objective metric, and therefore well-suited for automation.

Furthermore, we take samples of 72 ($\approx 20\%$), 63 ($\approx 10\%$) and 290 ($\approx 4\%$) of the generated utterances on ViGGO, E2E and MultiWOZ, respectively, annotate them for all types of errors, and calculate the *actual* SER confidence intervals (column "SER CI" in Table 3.5). Their good alignment with the slot aligner SER scores (column "SER_{SA}"), together with the high error classification precision, leads us to the conclusion that the slot aligner performs similarly to humans in identifying semantic errors on the above datasets.

3.4 Summary

In this chapter, we proposed a domain-adaptable slot aligner capable of many supporting tasks in an NLG system. We will be using it extensively throughout the rest of the thesis, primarily for candidate utterance reranking and for calculating SER scores for generated utterances. In a human evaluation we established that the slot aligner's performance is nearly perfect on three datasets: ViGGO, E2E and MultiWOZ.

Chapter 4

Sequence-to-Sequence Models for NLG

Here we describe the deep learning sequence-to-sequence models that we train for the data-to-text generation task on the datasets introduced in Chapter 2. We improve the performance of the base models with an adaptive method for delexicalizing slots and their values, and by employing the heuristic slot aligner introduced in Chapter 3 for augmenting the training sets and for semantic reranking of generated utterances.

4.1 Encoder-Decoder Architecture

The standard approach to sequence learning in the recent years has relied on the *encoder-decoder* architecture (Sutskever et al., 2014; Cho et al., 2014). This architecture allows the model to produce sequences of arbitrary lengths from input sequences, which is essential in our NLG task where we want to generate a natural language sentence from an MR. The model achieves this by first using the encoder to produce a compressed intermediate representation of the input sequence, and subsequently using the decoder to interpret it and produce an output sequence. More

advanced versions of the encoder-decoder architecture, with an *attention* mechanism (Bahdanau et al., 2015; Luong et al., 2015), enable the decoder to look at the input sequence along with its intermediate representation, which allows the decoder to make a more informed choice at each step of the sequence generation.

To represent the MR as a sequential input for the model, we flatten the dictionary of slots and values into a simple list of tokens without any special separators. Since the structure of the MRs we work with contains no hierarchical elements, no information is lost in the process. Leaving the slot names in the input sequences poses no problem either, as the model learns to use these tokens as mere separators and indicators of what information the immediately following tokens represent. Before feeding the input tokens to the encoder, they are converted to a lower-dimensional continuous-vector space. This sequence of *embedding* vectors is more suitable for processing by the encoder.

The encoder and decoder themselves can be various deep learning models, such as recurrent neural networks (RNNs) with Long Short-Term Memory (LSTM) cells (Hochreiter and Schmidhuber, 1997) or Gated Recurrent Unit (GRU) cells (Cho et al., 2014). Most recently, purely attentional encoder-decoders, such as the transformer (Vaswani et al., 2017) and its variants, have been favored over RNN-based ones thanks to their convenient parallelizability and often superior performance. For our earlier experiments, we used an LSTM-based encoder-decoder model with attention, and we later switched to using a transformer. Consequently, some of our evaluations are carried out using different models than others.

4.1.1 Recurrent Neural Network With Attention

Our first model uses a RNN with LSTM cells for both its encoder and decoder, and is equipped with attention. The attention mechanism allows the decoder to learn what specific parts of the input sequence to pay attention to, given the output generated so far. It does so by accessing all the hidden states of the encoder at each time step of the decoding, rather than merely using the final encoder state to initialize the state of the decoder. To define this more formally, the probability of output u_t at time step t of the decoder, given the input sequence $\mathbf{w}$ and the outputs predicted so far, depends on a distinct context vector q_t – produced by the attention mechanism – in the following way:

$$P(u_t|u_1, \ldots, u_{t-1}, \mathbf{w}) = g(s_t) \ ,$$

where in the place of function g we use the softmax function over the size of the vocabulary, and s_t is a hidden state of the decoder LSTM at time step t, calculated as:

$$s_t = f_D(u_{t-1}, s_{t-1}, q_t) \ .$$

The context vector q_t is obtained as a weighted sum of all the hidden states $h_1, \ldots, h_L$ of the encoder:

$$q_t = \sum_{i=1}^{L} \alpha_{t,i} h_i \ ,$$

where $h_i = f_E(w_i, h_{i-1})$, and $\alpha_{t,i}$ corresponds to the attention score the t-th word in the output sentence assigns to the i-th item in the input MR.

We compute the attention score $\alpha_{t,i}$ using a simple feedforward neural network jointly trained with the entire system, along the lines of the work in Bahdanau

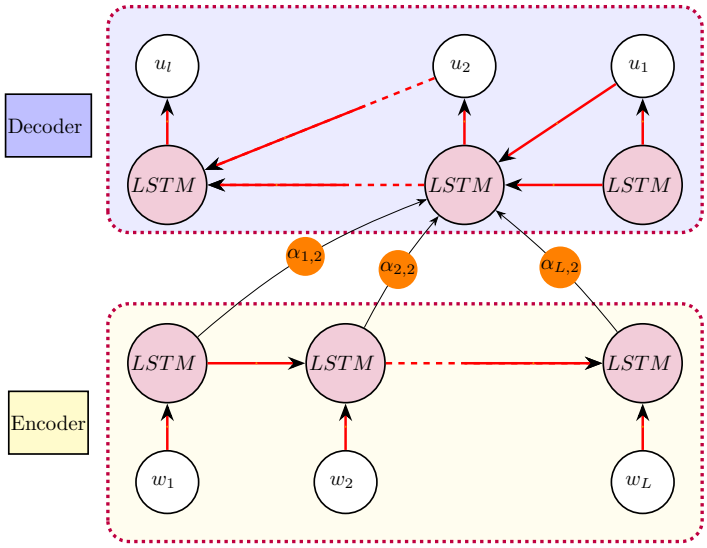


Figure 4.1: Standard architecture of a single-layer LSTM encoder-decoder model with a simplified illustration of the attention mechanism. For each time step t in the output sequence, the attention scores $\alpha_{t,1}, \ldots, \alpha_{t,L}$ are calculated. In this diagram, only attention scores for $t = 2$ are shown.

et al. (2015). The encoder's and decoder's hidden states at time i and t, respectively, are concatenated and used as the input to the neural network, namely:

$$\alpha_{t,i} = \text{softmax}\left(\mathbf{w}^T \tanh\left(W[s_t; h_i]\right)\right) \ ,$$

where W and $\mathbf{w}$ are the weight matrix and vector of the first and the second layer of the neural network, respectively. The learned weights indicate the level of influence of the individual words in the input sequence on the prediction of the word at time step t of the decoder. The model thus learns a soft alignment between the source and the target sequence.

4.1.2 Transformer

Even though, on the relatively small datasets we work with, we do not necessarily expect the transformer (Vaswani et al., 2017) model to perform better

than RNNs, we switched to running experiments with this model primarily for its significantly faster training, without sacrificing performance. Here we only give a high-level description of the transformer model and refer the reader to Vaswani et al. (2017) for full details of the architecture.

The primary difference of a transformer-based encoder-decoder from an RNN-based one is that it has no sequential dependencies, which is what gives it its superior parallelizable properties. Both the encoder and the decoder is a stack of multi-head self-attention blocks with normalization. The self-attention layer in the decoder blocks uses masking, and each block has an additional encoder-decoder attention layer that allows the decoder to peek at the relevant parts of the input sequence. Since the self-attention layers in the transformer do not know the concept of a sequence, the model uses positional encoding to inject sequential information into the input embeddings, representing thus the word order in the input sequence.

4.2 Adaptive Delexicalization

We enhance the ability of our models to generalize the learned concepts to unseen MRs by delexicalizing the training data. For each dataset we train our models on, we identify categorical and numeric slots whose values are always mentioned verbatim in the utterance, and replace the corresponding values in both the MR and the utterance with placeholder tokens. The placeholders are eventually replaced in the generated utterance in a post-processing step by copying the values from the original MR. Examples of such slots would be NAME in the E2E dataset, SCREENSIZE in the TV dataset, or DEVELOPER in ViGGO.

Previous work identifies categorical slots in general as good candidates for

delexicalization (Wen et al., 2015b; Nayak et al., 2017). However, we chose not to delexicalize those categorical slots whose values can be expressed in alternative ways, such as "less than $20" and "cheap", or "on the riverside" and "by the river". Excluding these from delexicalization may lead to a slightly increased number of incorrect realizations, but it encourages diversity of the model's outputs by giving it a freedom to choose among alternative ways of expressing a slot value in different contexts. This, however, assumes that the training set contains a sufficient number of examples exhibiting this type of alternation so the model could learn that certain phrases are synonymous.

As Nayak et al. (2017) point out, delexicalization affects the sentence planning and the lexical choice around the delexicalized slot value. For example, the realization of the slot FOOD[*Italian*] in the phrase "serves *Italian* food" is valid, while the realization of FOOD[*fast food*] in "serves *fast food* food" is clearly undesired. Similarly, a naive delexicalization can result in "a Italian restaurant", whereas the article should be "an". Another problem with articles is singular versus plural nouns in the slot value. For example, the slot ACCESSORIES in the TV dataset, can take on values such as "remote control", as well as "3D glasses", where only the former requires an article before the value.

We tackle this issue by defining different placeholder tokens for values requiring different treatment in the realization. For instance, the value "Italian" of the `food` slot is replaced by `⟨slot_vow_cuisine_food⟩`, indicating that the value starts with a vowel and represents a cuisine, while "fast food" is replaced by `⟨slot_con_food⟩`, indicating that the value starts with a consonant and cannot be used as a term for cuisine. The model thus learns to generate "a" before `⟨slot_con_food⟩` and "an" before `⟨slot_vow_cuisine_food⟩` when appropriate, as well as to avoid generating

the word "food" after FOOD slot placeholders that do not contain the word "cuisine". All these rules are general and can automatically be applied across different slots and domains.

4.3 Semantic Utterance Reranking

Rather than predicting a single most likely utterance, our models perform beam search during inference and produce thus a pool of possible utterances expressing the information in the given MR. While these utterance candidates have a probability score calculated by the model, we found that relying entirely on this score often results in a candidate being picked that is objectively worse than a lower scoring utterance (e.g., one missing a slot mention or realizing a slot incorrectly). We therefore adjust the score, multiplying it by the utterance's *alignment score* which takes the slot alignment into consideration:

$$s_{\text{align}} = \frac{N}{(N_n + 1) \cdot (N_d + 1)} \,,$$

where N is the number of all slots in the input MR, and N_n and N_d represent the number of non-aligned and duplicate slot mentions, respectively. As indicated in Section 3.2.5, our slot aligner can calculate N_n as the number of slots whose correct mentions it does not observe in the generated utterance (i.e., they are either missing or incorrect), while N_d is determined by identifying slots that are mentioned repeatedly.

4.4 Model Ensembling

In order to further improve the quality of the generated utterances, we ensemble multiple individually trained models and combine their predictions. We primarily use this technique with the LSTM-based models, two of which we put in an ensemble alongside a model with a convolutional neural network (CNN) encoder (LeCun et al., 1998; Gehring et al., 2017). The two LSTM models in the ensemble system are trained for a different number of steps each, but have otherwise the same parameters.

Since the models are in completely different states at a given time step during decoding, it is not possible to combine their predictions at token level. Therefore, we accumulate the top 10 fully generated utterances from each model using beam search, and allow the semantic reranker to rank all candidate utterances, as detailed in Section 4.3. Finally, our system outputs the utterance that receives the highest score after reranking.

4.5 Evaluation

Depending on the nature of the experiments, we make use of both automatic metrics and human evaluation to obtain a more accurate performance assessment. We report the results of the automatic evaluations on the standard NLG metrics listed in Table 4.1 along with their brief definitions. Although these metrics have repeatedly been shown to only inconsistently correlate with human judgments in NLG tasks that involve longer and more creative texts (Liu et al., 2016; Wiseman et al., 2017; Novikova et al., 2017a), the results of the E2E NLG Challenge (Dušek et al., 2020) showed that the winning system according to human evaluation scored

Metric	Definition
BLEU (Papineni et al., 2002)	Geometric mean of N-gram precision scores (with N up to 4) with brevity penalty.
ROUGE-L (Lin and Och, 2004)	Sentence-to-sentence longest common subsequence F-score.
METEOR (Banerjee and Lavie, 2005)	Recall-heavy unigram F-score with stem and synonym matching, and word-order penalty.
CIDEr (Vedantam et al., 2015)	Cosine similarity of TF-IDF weighted N-grams (with N up to 4) with stem matching.

Table 4.1: Simplified definitions of reference-based automatic metrics used in our evaluations.

the highest in some of the automatic metrics as well (Dušek et al., 2018). In addition to the standard automatic metrics, we also evaluate our models' outputs on the slot error rate (SER), as defined in Section 3.2.6.

We begin by providing more details on our system configurations, and then evaluate the system performance on the E2E and the ViGGO dataset. We first evaluate our complete system, Slug2Slug (Juraska et al., 2018), in the context of the E2E NLG Challenge we participated in, and then we move on to analyze the individual effects of the various techniques and components described in Chapters 3 and 4. Finally we look into how effective transfer learning is from E2E to ViGGO.

Note that we did not switch to the transformer architecture until after the challenge, and therefore only our more recent experiments – in particular those on the ViGGO dataset – were performed using this model. All of our model's results are averaged over 3 independent runs, unless stated otherwise.

4.5.1 System Configuration

In our experiments with LSTM-based models, both the encoder and the decoder have 4 layers and 512 cells per layer, and the encoder is bidirectional and uses 512-dimensional input embeddings. The transformer, on the other hand, is a

	BLEU	METEOR	ROUGE	CIDEr	SER
TGen	0.659	**0.448**	**0.685**	2.234	*N/A*
Slug2Slug	**0.662**	0.445	0.677	**2.262**	*0.91%*

Table 4.2: Slug2Slug, our LSTM/CNN ensemble system trained with sample splitting, compared to TGen, the baseline system in the E2E NLG Challenge.

small 2-layer one with 8 heads and using 256-dimensional embeddings, which proved to perform on par, if not slightly better than the large LSTM model. While in the LSTM models we settled on a dropout of 0.1, the transformer performed best with the value set to 0.2. For training both types of models we used the Adam optimizer with a custom learning rate schedule including a brief linear warm-up and a cosine decay.

After experimenting with different beam search parameters, we settled on the beam size of 10. We employed length normalization of the beams as defined in Wu et al. (2016) in order to encourage the decoder to favor longer sequences. The brevity penalty providing the best results on the E2E dataset was 0.6, whereas for ViGGO it was 1.0.

4.5.2 E2E Dataset Experiments

In terms of automatic metrics (see Table 4.2), our system performed comparably to the strong baseline model, TGen (Dušek and Jurčíček, 2016), in the E2E NLG Challenge. Nevertheless, systems which score similarly according to automatic metrics can produce utterances that are significantly different because these metrics fail to capture many of the characteristics of natural sounding utterances. Therefore, for a better assessment of the quality of our system's generated outputs, we present the results of a comprehensive human evaluation of the models' outputs in terms of both naturalness and quality, carried out by the challenge organizers.

Human Evaluation. *Quality* examines the grammatical correctness and adequacy of an utterance given an MR, whereas *naturalness* assesses whether a generated utterance could have been produced by a native speaker, irrespective of the MR. To obtain these scores, crowdworkers ranked the outputs of 5 randomly selected systems from worst to best. The final scores were produced using the TrueSkill algorithm (Sakaguchi et al., 2014) through pairwise comparisons of the human evaluation scores among the 20 competing systems.

Our system achieved the highest quality score in the E2E NLG Challenge, and was ranked second in naturalness.[1] The system's performance in quality (the primary metric) was significantly better than the competition according to the TrueSkill evaluation, which used bootstrap resampling with a p-level of $p \leq 0.05$. Comparing these results with the scores achieved by the baseline model in quality and naturalness (5th and 6th place, respectively) reinforces our belief that models that perform similarly on the existing automatic metrics can exhibit vast differences in the structural complexity of their generated utterances.

4.5.2.1 Ensembling

Testing our ensembling approach reveals that reranking predictions pooled from multiple different models produces an ensemble model that is overall more robust than the individual submodels. The submodels fail to perform well in all metrics at once, whereas the ensemble system is more consistent across the different metric types (Table 4.3).[2] Most importantly, the ensemble model decreases the

[1] The system that surpassed ours in naturalness was ranked the last according to the quality metric.

[2] Since the reference utterances in the test set were kept secret for the E2E NLG Challenge, we carried out the evaluation using the validation set, which was, nevertheless, used neither for training nor tuning the model.

	BLEU	METEOR	ROUGE	SER
LSTM1	0.6661	0.4644	0.7018	0.116%
LSTM2	0.6493	0.4649	0.6995	0.145%
CNN	0.6636	0.4700	0.7107	0.232%
Ensemble	0.6576	0.4675	0.7029	0.087%

Table 4.3: Automatic metric scores of 3 different models and their ensemble, tested on the validation set of E2E. LSTM2 differs from LSTM1 in that it was trained longer.

		BLEU	METEOR	ROUGE	CIDEr	SER
LSTM	–	0.628	0.449	0.680	2.231	*3.26%*
	d	0.655	**0.454**	0.675	2.195	***0.00%***
	ss	0.654	**0.454**	0.680	2.225	***0.00%***
Transf.	–	0.591	0.453	0.672	2.035	*1.42%*
	d	**0.664**	0.453	0.681	**2.248**	***0.00%***
	ss	0.660	0.451	**0.683**	2.230	***0.00%***

Table 4.4: Our LSTM and transformer models evaluated on the E2E dataset, with no training data preprocessing (–), with denoising only (d), and with sample splitting as well (ss).

proportion of incorrectly mentioned slots compared to its individual submodels.

4.5.2.2 Sample Splitting and Denoising

As we noted in Section 3.2.3, our sample splitting technique for augmenting the training set automatically performs denoising as well, so as to eliminate slots that are not recognized by the slot aligner during the splitting instead of being assigned to an arbitrary pseudo-sample. In order to be able to discern the effect of the sample splitting itself we train the same model on (1) the original E2E training set, (2) the denoised training set, and (3) the denoised training set augmented with pseudo-samples via sample splitting. All three instances are then evaluated on the original E2E test set.

The results in Table 4.4 indicate that the sample splitting with denoising

significantly increases the performance of a sequence-to-sequence model on the E2E dataset, whether it is an LSTM-based one or a transformer. The improvement is most noticeable in the BLEU metric, but especially the SER, which dropped all the way to zero when training the models on a denoised training set (with or without sample splitting). However, the results also show that the denoising by itself has an equal effect. We therefore conclude that the addition of more granular pseudo-samples, created through sample splitting, to the training set provides, in fact, no benefit for a neural model. As a consequence of this finding, and considering the ViGGO dataset is already very clean in terms of slot mentions, we do not use sample splitting and denoising in our later experiments on ViGGO.

We also note that the small transformer model performs on par with the TGen and Slug2Slug models (see Table 4.2 for comparison). Compared to an individual LSTM model, the transformer in fact performs marginally better. An interesting observation is, however, that the transformer made significantly fewer errors than the LSTM when trained on the original training set, but was greatly outperformed on the other metrics. With training data denoising though, we were able to achieve zero slot errors with both an LSTM and a transformer model on the E2E test set, whereas the SER of the original outputs of the Slug2Slug system is almost 1%.

4.5.3 ViGGO Dataset Experiments

In the experiments on our recently collected ViGGO dataset, we use the transformer model exclusively. The baseline results for the ViGGO dataset are reported in Table 4.5. The automatic metric scores are in general lower than on the E2E dataset, which can most likely be attributed to the smaller number of reference utterances available.

		Without reranking					With reranking					SER
		B	**M**	**R**	**C**	**SER**	**B**	**M**	**R**	**C**	**SER**	**red.**
WOT	**Ao3**	0.518	0.384	**0.631**	2.480	*6.32%*	0.519	0.388	0.631	2.531	*2.55%*	2.48×
	Bo5	0.524	0.387	0.638	2.494	*6.71%*	0.521	0.391	0.638	2.545	*2.48%*	2.71×
WT	**Ao3**	**0.523**	**0.386**	0.630	**2.533**	***3.84%***	**0.527**	**0.389**	**0.634**	**2.584**	***1.80%***	2.14×
	Bo5	0.523	0.384	0.625	2.522	*3.94%*	0.527	0.388	0.631	2.581	*1.75%*	2.25×
	Ens	–	–	–	–	–	0.526	0.389	0.634	2.555	*1.46%*	–

Table 4.5: Results of our experiments on the ViGGO dataset. WOT and WT denotes models trained *without* and *with* transfer learning, respectively. Despite individual models (Bo5 – best of 5) and ensembles (Ens) often having better scores, we consider the Ao3 (average of 3) results the most conclusive.

Human Evaluation. We let two expert annotators with no prior knowledge of the ViGGO dataset evaluate the outputs of our model. Their task was to rate 240 shuffled utterances (120 generated utterances and 120 human references) each on *naturalness* and *coherence* using a 5-point Likert scale. We define naturalness as a measure of how much one would expect to encounter an utterance in a conversation with a human, as opposed to sounding robotic, while coherence measures its grammaticality and fluency. Out of the 120 MRs in each partition, 40 were of the *inform* type, with the other 8 DAs represented by 10 samples each. In addition to that, we had the annotators rate a sample of 80 utterances from the E2E dataset (40 generated and 40 references) as a sort of a baseline for the human evaluation.

With both datasets, our model's outputs were highly rated on both naturalness and coherence (see Table 4.6). The scores for the ViGGO utterances were overall higher than those for the E2E ones, which we understand as an indication of the video game data being more fluent and conversational. At the same time, we observed that the utterances generated by our model tended to score higher than the reference utterances, though significantly more so for the E2E dataset. This is likely a consequence of the ViGGO dataset being cleaner and less noisy than the E2E dataset.

	Naturalness		Coherence	
	Ref.	**Gen.**	**Ref.**	**Gen.**
E2E	4.48	4.67	4.57	4.77
ViGGO$_{inf}$	4.85	4.83	4.85	4.93
ViGGO	4.68	4.74	4.78	4.84

Table 4.6: Naturalness and coherence scores of our model's generated outputs compared to the reference utterances, as per the human evaluation. ViGGO$_{inf}$ corresponds to the subset of *inform* DAs only.

In an additional evaluation of ViGGO, we asked the annotators to classify the utterance samples into the 9 DA groups. For this task they were provided with a brief description of each DA type. The annotators identified the DA incorrectly in only 7% of the samples, which we interpret as a confirmation that our DAs are well-defined. Most of the mistakes can be ascribed to the inherent similarity of the *recommend* and the *suggest* DA, as well as to our model often generating *give_opinion* utterances that resemble the *inform* ones.

Qualitative Analysis Among all 9 DAs, the one posing the greatest challenge for our model was *give_opinion*, due to its high diversity of reference utterances. Despite the occasional incoherence, it learned to produce rich and sensible utterances, for instance "Little Nightmares is a pretty good game. Tarsier Studios is a talented developer and the side view perspective makes it easy to play.". An example of a *recommend* utterance our model has generated is: "Since you seem to be a fan of sport racing simulators on the PC, I thought you might like F1 2014 too. Have you heard about the game?".

Since our baseline model does not implement any form of a copy mechanism, it fails on instances with out-of-vocabulary terms, such as the values of the SPECIFIER slot in the test set. These, in fact, account for almost half of the errors

indicated by the SER metric in Table 4.5. Therefore, more robust models have good potential for improving on our scores.

4.5.3.1 Utterance Reranking

The results in Table 4.5 confirm the positive impact of generated utterance reranking on the SER. The reduction of erroneous slot mentions by the factor of up to 2.5 is a significant benefit, especially considering it is not at the expense of the scores in the other metrics. In fact, we observe a clear trend of CIDEr scores being boosted when utterance reranking is used.

4.5.3.2 Transfer Learning

In order to improve the performance of our model on the ViGGO dataset, we experimented with the pretraining of our model on a large replicated video game dataset, as described in Section 3.2.4. While the benefits of transfer learning were not clearly visible when training models on the entire ViGGO dataset (see the bottom half of Table 4.5), the SER along with CIDEr score were still significantly improved. On the other hand, when we fine-tuned the pretrained model on the *inform*-only subset, the improvement was huge across all metrics, including more than 2 BLEU points, and a drop in SER from 1.83% to 0.83%, with a further decrease to 0.53% achieved with an ensemble model (see Table 4.7).

We thus showed that the performance increase on the entire dataset comes primarily from the *inform* DAs. However, constituting less than 30% of the test set, it leads to an overall small boost. We therefore conclude that pretraining on a large out-of-domain dataset is only effective on DAs of the same type, although it is possible that a positive effect could be observed on other DAs too in an extremely

		BLEU	METEOR	ROUGE	CIDEr	SER
WOT	**Ao3**	0.609	0.434	0.681	3.236	*1.83%*
	Bo5	0.608	0.434	0.681	3.332	*1.42%*
WT	**Ao3**	**0.631**	**0.440**	**0.696**	**3.316**	***0.83%***
	Bo5	0.645	0.442	0.709	3.404	*0.71%*
	Ens	0.629	0.440	0.699	3.317	*0.53%*

Table 4.7: Results of our experiments on the subset of the ViGGO dataset with *inform* DAs only. WOT and WT denote models trained *without* and *with* transfer learning, respectively.

data-starved scenario.

We also note that the model's scores on the *inform* DAs of the video game dataset are comparable to those achieved on the full E2E dataset, despite there being nearly 20 times less training data, more slots, and the reference utterances being on average significantly longer.

4.6 Summary

Building on the successful attentional encoder-decoder framework for sequence-to-sequence learning, we developed a robust NLG system that outperformed all competing systems in a data-to-text NLG shared task. We achieved this by using model ensembling, adaptive delexicalization of input data, and the many benefits of the automatic slot aligner. We further tested our model on ViGGO, setting a baseline for this new dataset, and used it to demonstrate the benefits of transfer learning for domain transferability. We showed that by pretraining our model on a modified version of the E2E dataset and subsequently fine-tuning it on ViGGO, we can achieve a significantly better performance and higher semantic accuracy on ViGGO than by training the model on ViGGO alone.

Chapter 5

Stylistic Control

Both a benefit and a pitfall of neural NLG models is that they are good at reducing noise in the training data. When they are trained on a sufficiently large dataset, they learn to generalize and become capable of applying the acquired knowledge to unseen inputs. The more data the models are trained on, the more robust they become, which minimizes the effect of noise in the data on their learning. However, the higher amount of training data can also drown out interesting stylistic features and variations that may not be very frequent in the data. In other words, a deep learning model, being statistical, will prefer producing the most common sentence structures, i.e., those which it observed most frequently in the training data and is thus most confident about.

In this chapter, we explore different ways of making language generated by neural models more natural and varied. To this end, we first develop text analysis methods that systematically characterize types of sentences in the training data. We then automatically label the training data with the help of the slot aligner described in Chapter 3, and a handful of domain-independent rules for discourse marker extraction, in order to allow us to conduct two kinds of experiments with

our neural language generator: (1) we test the effect of training the system with different stylistic partitions and quantify the effect of smaller, but more stylistically controlled training data; (2) we propose a method of labeling the style variants during training using auxiliary input tokens, and show that we can modify the style of the output using our stylistic labels. We contrast these methods, showing how they vary in terms of semantic quality and stylistic control. These methods promise to be usable with any sufficiently large corpus as a simple way of producing stylistic variation.

Being one of the largest datasets of its kind, we consider E2E the best available resource to perform the above experiments on. As pointed out earlier in Section 2.1, due to its size, as well as the data collection methods used, the E2E dataset manifests more interesting stylistic variations in the crowdsourced utterances. We also take advantage of the fact that it offers multiple alternative ways of expressing the information in each MR – which implies different styles too – when selecting the subset of examples for training with a particular purpose of stylistic variation.

5.1 Stylistic Selection

We note that the E2E dataset is significantly larger than what is needed for a neural model to learn to produce syntactically and semantically correct utterances in this domain. Thus we seek a way to help the model learn more than just to be correct; we strive to achieve higher diversity of the utterances generated by the model through stylistic selection of the training examples. We start by characterizing variation in the crowdsourced dataset and detecting what opportunities it offers for

Domain	Utterance
Video game	An excellent adventure RPG you can play from *either* a bird's eye view *or* in third person is Final Fantasy VII. It's available on Steam for PC, and you can play it on your PlayStation *too*.
TV	You might like the Dionysus 44 television that has an a+ eco rating and 720p resolution, *while only using 32 watts in power consumption.*
Laptop	*For the price of 449 dollars*, you could purchase the Satellite Hypnos 38 laptop.
People	*Born in the London Borough of Havering,* Alex Day started performing in 2006.
Food	Sago is the main ingredient in binignit, *but* sweet potatoes are *also* used in it.

Table 5.1: Examples of utterances in different datasets/domains, exhibiting interesting discourse phenomena.

the model to learn more advanced sentence structures. Table 5.2 illustrates some of the stylistic variation that we observe, which we describe in more detail below. We then assess the level of desirability of specific discourse phenomena in our context, and devise rules based on the utterance's dependency parse to extract examples that exhibit those stylistic phenomena. This gives us the ability to create subsets of examples with an arbitrary combination of stylistic features that we are interested in. We then explore the extent to which we can make the model's outputs demonstrate these stylistic features.

5.1.1 Stylistic Variation in the E2E Dataset

This section gives an overview of different discourse phenomena in the E2E dataset that we consider relevant in the context of a task-oriented dialogue in the restaurant domain. The majority of these would, however, generalize to other domains too, since they appear not only in summaries of restaurants, but also, for example, in those of TVs, laptops (Wen et al., 2016), people, food (Gardent et al., 2017), as well as video games (see examples in Table 5.1). The extraction rules we

Category	Utterance
Aggregation	Located in the city centre is a family-friendly coffee shop called Fitzbillies. It is **both inexpensive and highly rated**.
Contrast	The Rice Boat is a Chinese restaurant in the riverside area. It has a customer rating of 5 out of 5 **but** is not family friendly.
Fronting	**With a 1 out of 5 rating** Midsummer House serves Italian cuisine in the high price range, found not far from All Bar One.
Subordination	Wildwood pub is serving 5 star food **while keeping their prices low**.
Exist. clause	In the city center, **there** is an average priced, non-family-friendly, Japanese restaurant called Alimentum.
Imperative/ modal	In Riverside, **you'll** find Fitzbillies. It is a passable, affordable coffee shop which interestingly serves Chinese food. **Don't bring** your family though.

Table 5.2: Examples of the categories of discourse phenomena extracted from E2E utterances.

have implemented can thus be widely used in data-to-text language generators. We split the sentence features into the following six categories (an example of each is given in Table 5.2):

- **Aggregation:** Discourse phenomena grouping information together in a more concise way. This includes specifiers such as "both" or "also", as well as apposition and gerunds. Another type of aggregation uses the same quantitative adjective for characterizing multiple different qualities (such as "It has a *low customer rating and price range.*").

  Note that some of the following categories contain other markers that also represent aggregation.
- **Contrast:** Connectors and adverbs expressing concession or contrast between two or more properties, such as "but", "despite", "however", or "yet".
- **Fronting:** Fronted adjective, verb and prepositional phrases, typically highlighting properties of the entity (e.g., restaurant) before its name is given. In

this category we also include specificational copular constructions, which are formulations with inverted predication around a copula, bringing a particular property of the entity in the front (e.g., "*A family friendly option is* The Rice Boat.").

- **Subordination:** Clauses introduced by a subordinating conjunction (such as "if" or "while"), or by a relative pronoun (such as "whose" or "that").
- **Existential clause:** Sentences formulated using the expletive "there".
- **Imperative and modal verb:** Sentences involving a verb in the imperative form or a modal verb, making the utterance sound more personal and interactive.

5.1.2 Discourse Marker Weighting

Many human-authored utterances naturally contain multiple of the discourse phenomena described in Section 5.1.1. We would thus also prefer our system to be able to generate such utterances, as opposed to ones only containing a single discourse phenomenon of interest, especially if it is a common one, such as the existential clause. We therefore devise a weighting schema for different groups of discourse markers, whose purpose is to represent the markers' general desirability in the output utterances, as well as to counteract the sparsity of some of the markers compared to others. In other words, the weighting is supposed to ensure that all the most desirable utterances are picked from the training set during the selection, but some that only contain less interesting, and typically more prevalent, discourse phenomena would be omitted in favor of the more complex ones. Our reasoning behind this is that the greater the proportion of the most desirable discourse phenomena in the stylistically selected training set, the more confidently the model is expected to

Category	Subset of discourse markers	Proportion	Weight
Aggregation	"also, both, neither,...", quantitative adjectives	1.8%	3
	apposition	4.6%	2
	gerund	11.2%	2
Contrast	"but, however, despite, although,..."	5.4%	3
Fronting	fronted adjective/prepositional/verb clause	14.5%	2
Subordination	subordinating conjunction	2.9%	2
	relative pronouns	19.3%	1
Existential clause	expletive "there"	10.0%	1
Imperative/ modal	imperative	1.0%	2
	modal verb	4.1%	2

Table 5.3: The weighting schema for different discourse markers for each introduced category of discourse phenomena. For each set of markers we indicate the heuristically determined proportion of reference utterances in the training set they appear in.

generate utterances in which they are present.

For illustration, let us assume there are eight different reference utterances for an MR. All of them will be scored based on the discourse markers they contain, but only those that score above a certain threshold will be selected, while the rest will be ignored. The purpose of that is to encourage the model to learn to use, say, a contrastive phrase if there is an opportunity for it in the MR, and not be distracted by other possible realizations of the same MR, which are not as elegant. Thus, we can set the weighting schema in such a way that sentences containing only, for example, "which" or an existential clause, will not be picked. However, if there is no high scoring utterance for an MR, the utterance with the highest score is selected regardless so that the model would not miss the opportunity to learn from any MR examples.

Our final weighting schema – determined through a combination of the discourse markers' frequency in the dataset, their intra-category variation, as well

as their general desirability in the particular domain of our task – is specified in Table 5.3. When there are discourse markers from multiple subsets present in the utterance, the weights are accumulated. It is then the total weight that is used to determine whether the utterance satisfies a given stylistic threshold or should be eliminated. The weights can be tuned for any new domain according to the above or any other factors.

5.2 Input Data Annotation

5.2.1 Contrastive Relation

One of the discourse phenomena whose actualization could benefit from explicit indication of when it should be applied, is the contrastive relation between two (or more) slot mentions in the utterance. There are several reasons why such a comparison of specific slots would be desired in the restaurant domain. One of them is to provide emphasis that one attribute is positive, whereas the other is negative. Another natural reason in dialogue systems could be to indicate that the closest match to the user's query that was found is a restaurant that does not satisfy one of the requested criteria. A third instance is when the value of one attribute creates the expectation of a particular value of another attribute, but the latter has in reality the opposite value.

Some of the above could presumably be learned by the model if sufficient training data was available. However, they involve fairly complex sentence constructs with various potentially confusing rules for the neural network. The slightly more than 2K examples with a contrasting relation may easily be considered noise by the model among the tens of thousands of examples in the E2E dataset. Hence,

we augment the input MRs given to the model with the information about which particular slots should be put into a contrastive relation. We hypothesize that this explicit indication will help the model to learn to apply contrasting more easily despite the small proportion of training examples exhibiting the property.

In order to extract the information from the training utterance as precisely as possible, but in an automated fashion, we make use of our slot aligner yet again, this time using the positional information to identify two slots that are in a contrastive relation. For the relation we only consider the two scalar slots (priceRange and customerRating), plus the Boolean slot familyFriendly. Whenever a contrastive relation appears to the aligner to involve a slot other than the above three, we discard it as an undesirable utterance formulation. Depending on the values of the two identified slots, we assign the example either of the following labels:

- **Contrast:** If the slots have different values on a 3-point positivity scale that they can be mapped to (familyFriendly is only mapped to $\{1, 3\}$). An example would be customerRating being "low" ($\rightarrow 1$) and familyFriendly having the value "yes" ($\rightarrow 3$),
- **Concession:** If the slots have an equivalent value. For instance, customerRating being "5 out of 5" ($\rightarrow 3$) and priceRange having the value "cheap" ($\rightarrow 3$).

The label is added in the form of a new auxiliary slot in the MR, containing the names of the two corresponding slots as its value, such as `<contrast> [priceRange customerRating]`.

User query	Is there a **family-friendly Indian** restaurant nearby?
Response with no emphasis	*The Rice Boat* in city centre near Express by Holiday Inn is serving Indian food at a high price. It is **family-friendly** and received a customer rating of 1 out of 5.
Response with emphasis	A **family-friendly** option is *The Rice Boat*. This Indian cuisine is priced on the higher end and has a rating of 1 out of 5. They are located near Express by Holiday Inn in the city centre.

Table 5.4: Example of emphasizing the information about family-friendliness in an utterance conveying the same content.

5.2.2 Emphasis

Another property that it might be desirable in practice to have enforced in generated utterances is emphasis. Through fronting discourse phenomena, such as specificational copular constructions or fronted prepositional phrases, certain information about the entity can be emphasized at the beginning of the utterance. This could be used to make the dialogue system's responses sound more context-aware and thus natural. Consider the following example in the restaurant domain. Assume the user asks the system for a recommendation of a family-friendly Indian restaurant (see Table 5.4). Considering the user explicitly specifies the "family-friendly" requirement in the query, it is arguably more natural for the system response to be in the form of the second example in the table rather than the first.

We argue that the order of the information given in a response matters, depending on the context of the conversation, and should not be entirely arbitrary. That motivated us to identify instances in the training set where some information about the restaurant is provided in the utterance before its name. Using the slot aligner we extract the information about which slot(s) the opening segment of the utterance represents. Subsequently, we augment the corresponding flattened input to the model with additional `<emph>` tokens immediately before the slots that should

be emphasized in the generated utterance. This additional annotation will give the model an incentive to learn to realize such slots at the beginning of the utterance when desired. From the perspective of the dialogue manager in a dialogue system, it simply needs to indicate slots to emphasize along with the generated MR whenever applicable.

5.3 Evaluation

5.3.1 Style Subsets

In the initial experiments, we trained a model on the reduced training set, which only contains utterances filtered out based on the weighting schema defined in Table 5.3. Setting the threshold to 2, we obtained a training set of 17.5K examples, which is approximately 40% of the original training set. Although this heavily reduced training set had a higher concentration of more desirable reference utterances, it was not quite sufficient to achieve the desired effect. However, many of the discourse relations, including contrast, apposition, and fronting, appeared multiple times in the utterances generated from the test set, which was not the case for a model trained on the full training set.

Therefore, our next step was to verify whether our model is capable of learning all the concepts of the discourse phenomena individually and apply them in generated utterances. To this end, we repeatedly trained a model on subsets of the E2E dataset, each containing only examples with a specific group of discourse markers, as listed in the second column of Table 5.3. We then evaluated the outputs on the correspondingly reduced test set, using the same method we used for identifying examples with specific discourse markers, as described in Section 5.1.1. In

other words, we identified what proportion of the generated utterances did exhibit the desired discourse relation.

The results show that the model is indeed able to learn how to produce various advanced sentence structures that are, moreover, syntactically correct despite being trained on a rather small training set (in certain cases less than 2K examples). In all of the experiments, 97–100% of the generated utterances conformed to the style the model was trained to produce. Any occasional incoherence that we observed (e.g., "It has a high customer rating, but *are* not kid friendly.") was actually picked up from poor reference utterances in the training set. The only exception in the syntactic correctness was the *Imperative/modal* category. Since this is one of the least represented categories among the six, and due to the particularly high complexity and diversity of the utterances, the model trained exclusively on the examples in this category generated a significant proportion of slightly incoherent utterances.

5.3.2 Data Annotation

The first set of experiments we performed with the data annotation involved explicit indication of emphasis in the input (see Section 5.2.2). As the results in Table 5.6 show, the model trained on data with emphasis annotation reached an almost 98% success rate of generating an utterance with the desired slots emphasized.[1] In order to get a better idea of the impact of the annotation, notice that the same model trained on non-annotated data does not produce a single utterance with emphasis; instead it produces utterances in the usual rigid style, which always starts with the name of the restaurant (see Table 5.5).

[1] There were 3,309 slots across all the test MRs that were labeled as to-be-emphasized.

MR	NAME [Wildwood], **EATTYPE [coffee shop]**, **FOOD [English]**, PRICERANGE [moderate], **CUSTOMERRATING [1 out of 5]**, **NEAR [Ranch]**
Reference	A **low** rated **English** style **coffee shop** around **Ranch** called Wildwood has moderately priced food.
No emph. annotation	Wildwood is a **coffee shop** providing **English** food in the moderate price range. It is located near **Ranch**.
With emph. annotation	There is a **low** rated **English coffee shop** near **Ranch** called Wildwood. It has a moderate price range.

Table 5.5: Examples of generated utterances with or without emphasis annotation. The MR's slots to be emphasized before the restaurant name, and the corresponding slot realizations, are in bold.

	Emph. realiz.	**SER**
Reference	100.00%	8.48%
No emph.	0.00%	3.45%
With emph.	97.85%	5.82%

Table 5.6: Comparison of the emphasis realization success rate (precision) and the slot error rate (SER) in the generated outputs using data annotation against the reference utterances, as well as the outputs of the same model trained on non-annotated data.

We notice that the error rate of the slot realization rises (from 3.45% to 5.82%) when the annotation is introduced. Nevertheless, it is still lower than the error rate among the reference utterances in the test set, in which over 8% of slots have missing mentions. Thus we find it acceptable considering the desired stylistic improvement of the output utterances.

The experiments with contrastive relation annotation also show a significant impact of the added labels on the style of the utterances generated by our model. However, the success rate of the realization of a contrast/concession formulation was only 49.12%, and the slot error rate jumped up to 8.34%. The contrast and concession discourse relations being syntactically more complex, and at the same time being less prevalent among the training utterances, it is understandable that it is more difficult for the model to learn how to use them properly.

priceRange	customerRating	Frequency
less than £20	low	2,153
£20–25	3 out of 5	919
moderate	3 out of 5	1,282
more than £30	high	1,329
more than £30	5 out of 5	921

Table 5.7: Combinations of the slot values for which aggregation would be possible. Note that only the combinations with a non-zero frequency are listed.

5.3.3 Aggregation

One of the aggregation discourse markers that we identified as contributing to the stylistic variation in an interesting way is, unfortunately, very sparsely represented in the E2E dataset. It is the last aggregation type described in the category overview in Section 5.1.1. Its scarcity in the training set would not make it feasible to train a successful neural model on the subset of the corresponding examples only.

Nevertheless, we analyze the potential for this aggregation in the training set. Since there are only two scalar slots in this dataset, priceRange and customerRating, we obtain the frequencies of their value combinations. Both of these take on values on a scale of 3, however, the values are different for each of the slots. Moreover, there are two sets of values for both slots throughout the dataset. We observed, however, that the values between the two sets are used somewhat interchangeably in the utterances, e.g., "low" seems to be a valid expression of the "less than £20" value of the priceRange slot, and vice versa.

As can be seen in Table 5.7, the potential for this type of aggregation is rather limited. Although the 6,604 examples in which a feasible value combination can be found corresponds to over 15% of the training set, due to the values not matching exactly between the two slots, aggregation was not elicited in the human-authored utterances. Moreover, a high value in the customerRating means it is

a positive attribute, while in the priceRange slot it indicates a negative attribute. We conjecture this might have also deterred the crowdworkers who produced the utterances from aggregating the values together.

5.4 Summary

In this chapter, we presented two different methods of giving a neural language generation system greater stylistic control. Our results indicate that the data annotation method has a significant impact on the model being able to learn how to use a specific style and sentence structures, without an unreasonable impact on the semantic error rate. Both methods are a convenient way for achieving the goal of stylistic control when training a neural model with an arbitrary existing large corpus.

Chapter 6

Semantically Attention-Guided Decoding for Data-to-Text NLG

This chapter picks up the semantic accuracy thread started in Chapter 3. As we could see in our experiments in Chapter 5, increased diversity in generated utterances tends to come at the cost of their faithfulness to the input. With a reliable way of automatically detecting errors in model outputs, however, we may be able to optimize for both at the same time.

Several different approaches to enhancing semantic accuracy of neural end-to-end models have been proposed for data-to-text NLG over the years. The most common approach to ensuring semantic quality relies on over-generating and then reranking candidate outputs using criteria that the model may not have been explicitly optimized for in training. Reranking in NLG models is typically performed by creating an extensive set of rules, or by training a supplemental classifier, that indicates for each input slot whether it is present in the output utterance (Wen et al., 2015a; Dušek and Jurčíček, 2016; Juraska et al., 2018; Agarwal et al., 2018; Kedzie and McKeown, 2020; Harkous et al., 2020).

Wen et al. (2015b) proposed an extension of the underlying LSTM cells of their sequence-to-sequence model to explicitly track, at each decoding step, the information mentioned so far. The coverage mechanism (Tu et al., 2016; Mi et al., 2016; See et al., 2017) penalizes the model for attending to the same parts of the input based on the cumulative attention distribution in the decoder. Chisholm et al. (2017) and Shen et al. (2019) both introduce different sequence-to-sequence model architectures that jointly learn to generate text and reconstruct the input facts. An iterative self-training process using data augmentation (Nie et al., 2019; Kedzie and McKeown, 2019) was shown to reduce semantic NLG errors on the E2E dataset. Among the more recent efforts, the jointly-learned segmentation and alignment method of Shen et al. (2020) improves semantic accuracy while simultaneously increasing output diversity. Kedzie and McKeown (2020) use segmentation for data augmentation and automatic utterance planning, which leads to a reduction in semantic errors on both the E2E and ViGGO datasets.

Also related to our problem of controlling for semantic accuracy is the line of work researching controllable neural language generation, in which the constrained decoding strategy is often used, rescoring tokens at each decoding step based on a set of feature discriminators (Ghazvininejad et al., 2017; Baheti et al., 2018; Holtzman et al., 2018). Nevertheless, this method is typically used with unconditional generative LMs, and hence does not involve input-dependent constraints.

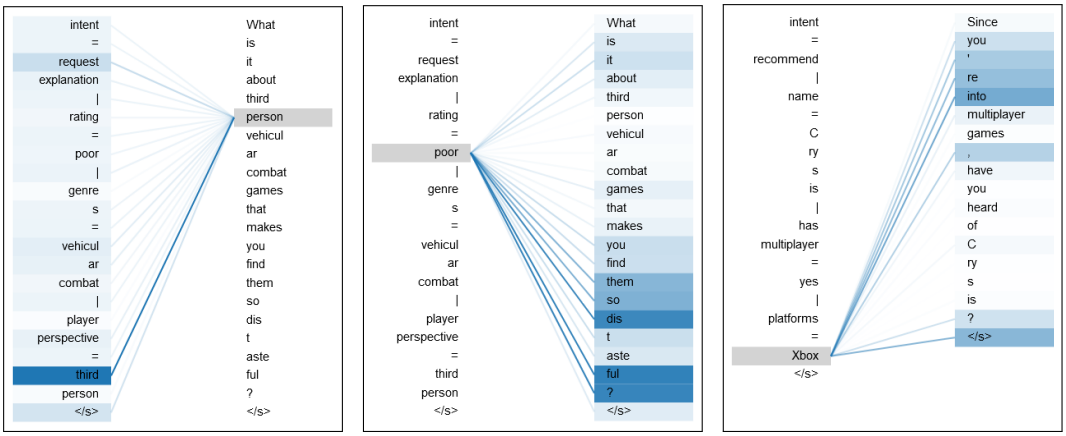
In this chapter, we study the behavior of attention in large pretrained language models (LMs) fine-tuned for a data-to-text NLG task, and how it relates to the semantic accuracy in their outputs. We show that encoder-decoder models equipped with cross-attention (i.e., an attention mechanism in the decoder looking back at the encoder's outputs) are, in fact, aware of the semantic constraints, yet standard

decoding methods do not fully utilize the model's knowledge. The method we propose extracts interpretable information from the model's cross-attention mechanism at each decoding step, and uses it to infer which slots have been correctly realized in the output. Coupled with beam search, we use the inferred slot realizations to rescore the beam hypotheses, preferring those with the fewest missing or incorrect slot mentions.

In contrast to previous work, our approach does not rely on model modifications, data augmentation, or manual annotation. Our method is novel in that it utilizes information that is already present in the model itself to perform semantic reranking. Although our approach also exploits the cross-attention between the encoder and the decoder, it differs from the coverage mechanism in that it does not require any modifications to the language generation model itself. In Section 6.1.1 we also demonstrate certain nuances of cross-attention, which the coverage mechanism may not be able to account for.

6.1 Semantic Attention-Guided Decoding

While we will evaluate the SEA-GUIDE method on ViGGO, E2E, and MultiWOZ, we develop the method by careful analysis of the cross-attention behavior of different pretrained generative LMs fine-tuned on the ViGGO dataset. The motivation for selecting ViGGO for developing the method was that it is the smallest dataset, but it provides a variety of DA and slot types (as shown in Table 2.1 in Chapter 2). The models used for the analysis were the smallest variants of T5 (Raffel et al., 2020) and BART (Lewis et al., 2020). We saved the larger variants of the models, as well as the other two datasets, for the evaluation.



(a) Verbatim slot mention (1^{st} layer).

(b) Paraphrased slot mention (3^{rd} layer).

(c) Unrealized slot mention (4^{th} layer).

Figure 6.1: Visualization of cross-attention weight distribution for the 6-layer T5-small (trained on the ViGGO dataset) in 3 different scenarios. The left column in each corresponds to the input tokens, and the right to the tokens generated by the decoder. The darker the blue background shade, the greater the attention weight. Note that the weights are aggregated across all attention heads by extracting the maximum.

6.1.1 Interpreting Cross-Attention

Attention (Bahdanau et al., 2015; Luong et al., 2015) is a mechanism that was introduced in encoder-decoder models (Sutskever et al., 2014; Cho et al., 2014) to overcome the long-range dependencies problem of RNN-based models. It allows the decoder to effectively condition its output tokens on relevant parts of the encoder's output at each decoding step. The term *cross-attention* is primarily used when referring to the more recent transformer-based encoder-decoder models (Vaswani et al., 2017), to distinguish it from the *self-attention* layers present in both the encoder and the decoder transformer blocks. The cross-attention layer ultimately provides the decoder with a weight distribution at each step, indicating the importance of each input token in the current context.

Our results below will show that visualizing the attention weight distribution for individual cross-attention layers in the decoder – for many different inputs

– reveals multiple universal patterns, whose combination can be exploited to track the presence, or lack thereof, of input slots in the output sequence. Despite the differences in the training objectives of T5 and BART, as well as their different sizes, we observe remarkably similar patterns in their respective cross-attention behavior. Below, we describe the three most essential patterns (illustrated in Figure 6.1) that we use in SEA-GUIDE.

6.1.1.1 Verbatim Slot Mention Pattern

The first pattern consistently occurs in the lowest attention layer, whose primary role appears to be to retrospectively keep track of a token in the input sequence that the decoder just generated in the previous step. Figure 6.1a shows an example of an extremely high attention weight on the input token "third" when the decoder is deciding which token to generate after "What is it about *third*" (which ends up being the token "person"). This pattern, which we refer to as the *verbatim* slot mention pattern, can be captured by maximizing the weight over all attention heads in the decoder's first layer.

6.1.1.2 Paraphrased Slot Mention Pattern

Paraphrased slot mentions, on the other hand, are captured by the higher layers, at the moment when a corresponding token is about to be mentioned next. Essentially, as we move further up the layers, the cross-attention weights gradually shift towards input tokens that correspond to information that is most likely to *follow next* in the output, and capture increasingly more abstract concepts in general. Figure 6.1b shows an example of the RATING slot's value "poor" paraphrased in the generated utterance as "distasteful"; the first high attention value associated

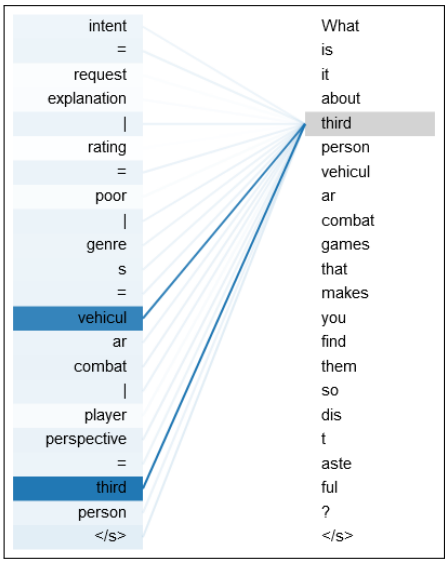


Figure 6.2: Example of the decoder paying equal attention (in the 5^{th} layer of the 6-layer T5-small) to two slots in the input sequence when deciding what to generate next after "What is it about".

with the input token "poor" occurs when the decoder is about to generate the "dis" token.

At certain points during generation, however, the attention in the uppermost layers is distributed fairly evenly among multiple slots, because any of them could lead to a coherent continuation of the sentence. For example, the generated utterance in Figure 6.2 could have started with "What is it about vehicular combat games played from a third-person perspective that…", where the GENRES slot is output before the PLAYER PERSPECTIVE slot.

In order to recognize a paraphrased mention, without incorrectly capturing other slots considered, we propose averaging the cross-attention weights, using only the bottom half of the layers (e.g., layers 1 to 3 in the T5-small model).

6.1.1.3 Unrealized Slot Mention Pattern

The third pattern alleviates any undesired side effects of identifying paraphrased mentions using the second pattern, i.e., slots incorrectly assumed to be mentioned. Figure 6.1c illustrates an unrealized slot (PLATFORMS) being paid attention to in several decoding steps. The cross-attention weight distribution for the "Xbox" token in the 4th layer, shows that the decoder considered mentioning the slot at step 5 (e.g., "Since you're *an Xbox fan* and like multiplayer games,..."), as well as step 8 (e.g., "...into multiplayer games *on Xbox,*..."). The second pattern, depending on the sensitivity setting (see Section 6.1.2), might infer the PLATFORMS slot as a paraphrased mention at step 5 and/or 8.

However, the PLATFORMS slot's value is also paid attention to when the decoder is about to generate the EOS token and, importantly, without any high attention weights associated with other slots at this step. This suggests that the model *is aware* that it omitted that slot. However, at that point, the decoder is more confident ending the sentence than realizing the missed slot after generating a question mark. This *unrealized* slot mention pattern is most likely to occur in the higher cross-attention layers, but not necessarily, so it is more effective to capture it by averaging the attention weights over all layers (at the last decoding step).

Note on Boolean Slots. With any of the three patterns described above, Boolean slots, such as HAS MULTIPLAYER in Figure 6.1c, typically have a high attention weight associated with their name rather than the value. This observation leads to a different treatment of Boolean slots, as described below.

6.1.2 Slot Mention Tracking

We use the findings of the cross-attention analysis for automatic slot mention tracking in the decoder. During decoding, for each sequence, the attention weights associated with the next token to be generated are aggregated as per Section 6.1.1. Using configurable *thresholds*, the aggregated weights are then binarized, i.e., set to 1 if above the threshold, and 0 otherwise. This determines the sensitivity of the pattern recognition. Optionally, all but the maximum weight can be set to 0, in which case only a single input token will by implied even if the attention mass is spread evenly across multiple tokens. Finally, the indices of binarized weights of value 1, if any, are matched with their corresponding slots depending on which slot-span in the input sequence they fall into.

To automatically extract slot spans, we parse the input MRs on-the-fly – which is trivial given the structured nature of MRs – as each batch is being prepared for inference, and create a list of slot spans for each MR in the batch.[1] In fact, we indicate the spans for slot names and slot values separately, and for list-values down to individual list elements, for a higher specificity. Since Boolean slot mentions are tracked by their name rather than value, we also indicate for each slot whether it is Boolean or not. This information can be provided explicitly to the data loader, otherwise it is automatically inferred from the dataset's ontology based on all the possible values for each slot.

[1] This is done on token level, and the result varies thus from model to model depending on its tokenizer.

	Verbatim	Paraphrased	Unrealized
Layer agg.	1st layer only	avg. over bottom half of layers	avg.
Head agg.	max.	max.	max.
Bin. threshold	0.9	0.4 (T5-small) 0.3 (BART-base)	0.1
Bin. max.	yes	no	no

Table 6.1: Final configuration of parameters used in each of the 3 mention-tracking components. The "Bin. max." row indicates whether only the maximum weight is kept during binarization, or all above the threshold.

6.1.2.1 Mention-Tracking Components

The three mention-tracking components, each of which operates on different attention layers and uses a different weight aggregation and binarization strategy, are summarized in Table 6.1. These components are executed in sequence and update one common slot-tracking object.

The first component, which tracks verbatim mentions, operates on the first attention layer only, with a high binarization threshold. Slot mentions identified by this component are regarded as high-confidence. The second component tracks paraphrased mentions, which are identified as slot mentions with low confidence, due to the partial ambiguity in mention detection using the second pattern (see Section 6.1.1.2). The third component only kicks in when the EOS token is the most probable next token. At that point, it identifies – with high sensitivity – slots that were not realized in the sequence (e.g., the PLATFORMS slot in Figure 6.1c), and removes the corresponding mention record(s). Only low-confidence mentions can be erased, while high-confidence ones are final once they are detected.

6.1.3 Semantic Reranking

Combining the slot mention tracking with beam search, for each input MR we obtain a pool of candidate utterances along with the semantic errors inferred at decoding time. We then rerank the candidates and pick the one with the fewest errors, resolving ties using the length-weighted log-probability scores determined during beam search.

6.2 Evaluation

Besides ViGGO, which we use for fine-tuning the decoding (slot-tracking) parameters of the proposed SeA-GuiDe method, we evaluate its effectiveness for semantic error reduction on two *unseen* and *out-of-domain* datasets: E2E and MultiWOZ. Table 2.1 in Chapter 2 gives an overview of all three datasets' sizes and properties.

6.2.1 Data Preprocessing

When preprocessing input MRs before fine-tuning a pretrained model or running inference with such a model, we convert the MR's structured format into a more natural language. This turned out to significantly increase the performance compared to using the raw MRs, which we attribute to these large LMs being pretrained on textual data in natural language, without seeing much structured data. Below, we briefly describe the process.

We first parse the DA types, if present, and all slots and their values from the dataset-specific format into an intermediate list of slot-and-value pairs, keeping the original order. Although typically indicated in the MR differently from slots,

we treat the DA type as any other slot (with the value being the DA type itself, and assigning it the name "intent"). Next, we rename any slots that do not have a natural-language name (e.g., "priceRange" to "price range", or "has_mac_release" to "has Mac release"). Slot values are left untouched. We perform this slot name conversion to take advantage of the pretrained LMs' ability to model the context when the input contains familiar words, as opposed to feeding it code names with underscores and no spaces. It changes them into natural-language phrases which the LM will understand better, otherwise it would not be able to associate them with any of the words and phrases it encountered during pretraining. Finally, we convert the updated intermediate list of slots and their values to a string. The '|' symbol is used for separating slot-and-value pairs from each other, while the '=' is used within each pair to separate the value from the slot name. The result for an MR from ViGGO can look as follows:

```
intent = request explanation | rating = poor | genres = vehicular combat
| player perspective = third person
```

6.2.2 Experimental Setup

In our experiments, we fine-tune T5 and BART models of varying sizes on the above datasets' training partitions, select the best model checkpoints based on the BLEU score they achieve on the respective validation set, and evaluate them on the test sets while using different decoding methods for inference. For beam search decoding, including when used as part of SEA-GUIDE, we use beam size 10 and early stopping, unless stated otherwise. All of our results are averaged over 3 runs with random initialization.

The pretrained models that we fine-tuned for our experiments are the Py-

	Layers	Heads	Hidden state size	Total parameters	Batch size	Learning rate	Epochs
T5-small	6+6	8	512	≈ 60M	32/64/64	2×10^{-4}	20/20/30
BART-base	6+6	12	768	≈ 139M	32/32/32	1×10^{-5}	20/20/25
T5-base	12+12	12	768	≈ 220M	16/16/ –	3×10^{-5}	20/20/ –
BART-large	12+12	16	1024	≈ 406M	16/ – / –	4×10^{-6}	20/ – / –

Table 6.2: Overview of the model specifications and the training parameters used in our experiments. Batch size and the number of epochs are indicated per dataset (ViGGO/E2E/MultiWOZ).

Torch implementations in the Hugging Face's Transformers library[2] (Wolf et al., 2020). The model sizes are indicated in Table 6.2. We trained all models using a single Nvidia RTX 2070 GPU with 8 GB of memory and CUDA version 10.2. The training parameters too are summarized in Table 6.2. For all models, we used the AdamW optimizer with a linear decay after 100 warm-up steps. The maximum sequence length for both training and inference was set to 128 for ViGGO and E2E, and 160 for MultiWOZ.

6.2.3 Automatic Evaluation Metrics

We evaluate our trained models' performance with the standard NLG metrics BLEU, METEOR, ROUGE-L, and CIDEr. To ensure a fair comparison with the MultiWOZ baselines (Peng et al., 2020; Kale and Rastogi, 2020), we additionally report BLEU scores calculated using the RNNLG evaluation script[3], which their respective authors used in their own evaluation. We denote it BLEU_R in our results tables.

In order to measure the proposed decoding method's performance in se-

[2] https://huggingface.co/transformers/
[3] https://github.com/shawnwun/RNNLG/

mantic error reduction, we use our slot aligner from Chapter 3. We use it to count missing, incorrect, and duplicate slot mentions, and determine the slot error rate (SER) as the percentage of these errors out of all slots in the test set's MRs (see Section 3.2.6). The slot aligner is rule-based and took dozens of man-hours to develop, but it is robust and extensible to new domains, so it works on all three test datasets. With the slot aligner we can calculate SER automatically for all our model outputs across all datasets and configurations tested, which would be infeasible to have human annotators do.

Besides SER evaluation, we use the slot aligner for beam search candidate reranking as one of our baselines. Due to the handcrafted and domain-specific nature of the slot aligner, beam search with this reranking has a distinct advantage over SEA-GUIDE, which can be used for any domain out of the box. We therefore consider the results when using the slot-aligner reranking to be an upper bound for SEA-GUIDE in terms of SER, rather than a baseline.

We also note that Kale and Rastogi (2020) calculated SER on utterance level, rather than slot level, and that using *exact* slot value matching in the utterance. We thus wrote a script to also perform this type of naive SER evaluation, in addition to our slot aligner-based SER evaluation. We report its results as SER_{E}.

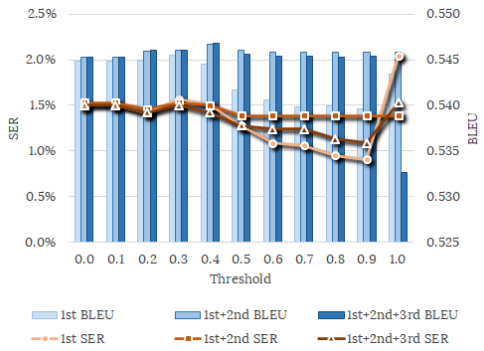
6.2.4 SEA-GUIDE Parameter Tuning

Each of the three mention-tracking components described in Section 6.1.2.1 has four configurable parameters, which we tuned by testing T5-small and BART-base, fine-tuned on the ViGGO dataset and equipped with SEA-GUIDE for inference. The parameter optimization was based on the insights obtained in Section 6.1.1 and a subsequent grid search, with results in Table 6.1.

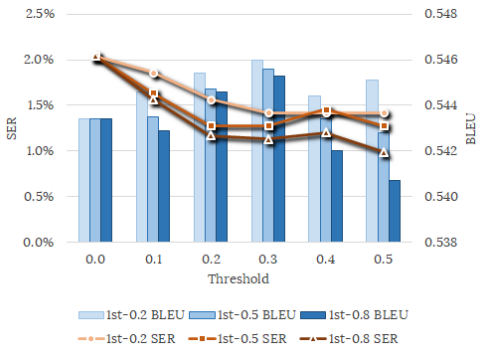
For attention weight aggregation, we experimented with summing, averaging, maximizing, and normalizing. We determined *averaging* over layers and *maximizing* over heads to be the best combination for all three components. As for the binarization thresholds, Figure 6.3 shows the most relevant slice of the grid search space for each component, leading to the final threshold values.

To show the effect of each slot-tracking component, we perform an ablation study with individual components disabled.[4] As the plot in Figure 6.3a demonstrates, the 1st component by itself reduces the SER the most, but at the expense of the BLEU score, which decreases as the SER does – to the point where BLEU drops below 0.54 when the SER is at its lowest (0.91%), that is with a threshold of 0.9. For reference, the SER and the BLEU score achieved with beam search only are 2.04% and 0.543, respectively. Adding the 2nd component brings the BLEU score up to above 0.545, nevertheless the SER jumps to 1.39%. Finally, enabling the 3rd component too has a negligible negative effect on BLEU, but reduces the SER to 1.09%.

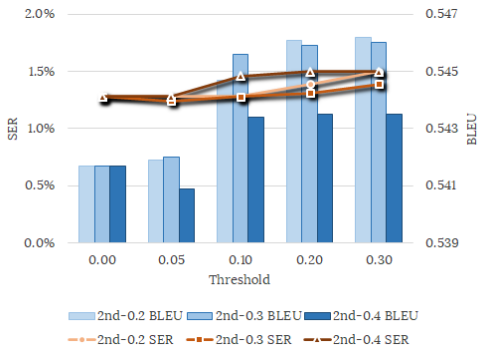
Figure 6.3b shows that the 2nd component gives optimal performance when its threshold is set to around 0.3. This setting maximizes BLEU, while keeping SER low. Beyond 0.3 the BLEU score starts dropping fast, and with a threshold of greater than 0.5, the 2nd component has barely any effect anymore. Similarly, Figure 6.3c shows the threshold value of 0.1 to be optimal in the 3rd component, when optimizing for both metrics. Thresholds higher than 0.3 cut off almost all aggregated weights in this component, virtually disabling it.



(a) Threshold optimization for the 1st component (verbatim mentions), with the other components enabled or disabled. When enabled, the 2nd component's threshold was fixed at 0.3, and that of the 3rd at 0.1. Note that the threshold of 1.0 is equivalent to the 1st component being disabled, as attention weights are in the $[0.0, 1.0]$ range.

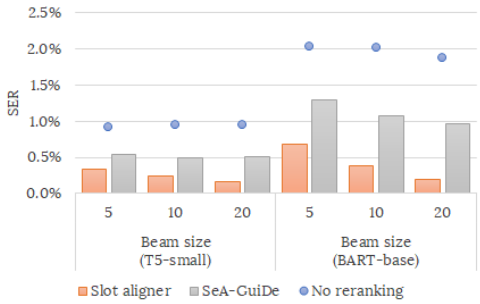


(b) Threshold optimization for the 2nd component (paraphrased mentions), with the 1st component's threshold of 0.2, 0.5 and 0.8, and that of the 3rd component fixed at 0.1.

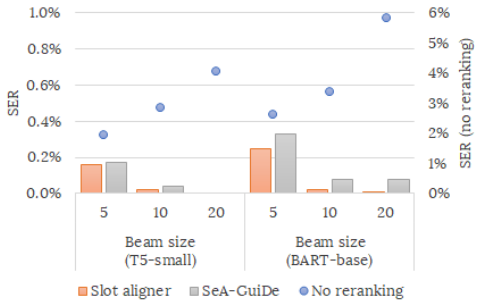


(c) Threshold optimization for the 3rd component (unrealized mentions), with the 2nd component's threshold of 0.2, 0.3 and 0.4, and that of the 1st component fixed at 0.5.

Figure 6.3: Effects of different parameter configurations of the 3 mention-tracking components on SER and BLEU of utterances generated by BART-base fine-tuned on ViGGO.



(a) ViGGO. With greedy search decoding, the SER is 1.65% and 2.70% for T5 and BART, respectively.



(b) E2E. With greedy search decoding, the SER is 1.60% and 1.97% for T5 and BART, respectively.

Figure 6.4: Effect of different beam sizes on the SER using different reranking methods on the ViGGO and E2E datasets.

6.2.5 Effects of Beam Size on SeA-GuiDe

Since SeA-GuiDe uses beam search to generate the pool of candidates that it later reranks, we analyzed the effect of increasing the beam size on the SER of the final utterances. As Figure 6.4a shows for the ViGGO dataset, SeA-GuiDe certainly benefits from increasing the beam size from 5 to 10, but the benefit shrinks substantially (or disappears entirely, in case of T5-small) when further increased to 20.

On the E2E dataset, decoding using SeA-GuiDe is even more effective in reducing SER than on ViGGO. Across all beam sizes, its performance is comparable to beam search with slot aligner reranking, and there is also only a limited gain from increasing the beam size to 20 (see Figure 6.4b). It is worth noting that, using beam search with no reranking, the SER dramatically increases with the increasing beam size. This is likely caused by the relatively heavy semantic noise in the E2E training set, resulting in more slot errors in the generated utterances the less greedy the decoding is. Some form of semantic guidance is thus all the more important for the

[4]The 3rd component has no effect without the 2nd, so we do not consider the combination where only the 2nd is disabled.

model in this scenario.

6.3 Results

To maximize the performance of the models using SEA-GUIDE, the binarization thresholds (and possibly other parameters of the mention-tracking components) can be optimized for each model and dataset on the validation set. In our evaluation, however, we focused on demonstrating the effectiveness of this decoding method out of the box. That being said, even common decoding methods, such as simple beam search or nucleus sampling (Holtzman et al., 2019), usually benefit from parameter optimization (e.g., beam size, or the p-value) whenever used with a different model or dataset.

6.3.1 SEA-GUIDE Performance

While developing the SEA-GUIDE method we analyzed the behavior of cross-attention on both the T5-small and the BART-base model; interestingly, the decoding performs best for both with nearly the same configuration. The only difference is the 2$^{\text{nd}}$ component's binarization threshold (see Table 6.1), accounting for the fact that BART-base has 50% more attention heads than T5-small, which causes the attention weights to be more spread out.

The upper half of Table 6.3 compares the two models' performance with SEA-GUIDE vs. other decoding methods, as well as against three state-of-the-art baselines. As the results show, both models, when using SEA-GUIDE, significantly reduce the number of semantic errors in the generated outputs compared to using greedy search ($\approx$ 3.4 and 2.5 times in case of T5 and BART, respectively) or sim-

ple beam search ($\approx$ 1.9 times both). As expected, the slot-aligner (SA) reranking achieves even better results thanks to the handcrafted rules it relies on. In addition, the overall high automatic metric scores suggest that the fluency of utterances generated using SeA-GuiDe does not suffer.

Finally, compared to the baseline models, T5-small performs on par with the state-of-the-art DataTuner in terms of automatic metrics, yet maintains a 3.4-times lower SER. This corresponds approximately to K&M baseline's SER, whose automatic metrics, however, are significantly worse. BART-base outperforms T5-small according to most metrics, but its SER is more than double.

6.3.2 Cross-Model Robustness

In addition to T5-small and BART-base, we fine-tune a larger variant of each of the models, namely, T5-base and BART-large (see Section 6.2.2 for model specifications), on the ViGGO dataset, and evaluate their inference performance when equipped with SeA-GuiDe. We do not perform any further tuning of the decoding parameters for these two models, only slightly lower the binarization thresholds (as we did for BART-base) to account for the models having more attention heads and layers. The thresholds we use for the 2^{nd} and 3^{rd} components are $\langle 0.3, 0.1 \rangle$ and $\langle 0.2, 0.05 \rangle$ for T5-base and BART-large, respectively.

The results in the lower half of Table 6.3 show that these two larger models, fine-tuned on ViGGO, benefit from SeA-GuiDe beyond just the effect of beam search. T5-base performs significantly better across the board than its smaller T5 variant when using greedy search decoding, so there is less room for improvement to begin with. In fact, the SER using greedy search is so low (0.61%, in contrast to T5-small's 1.65%) that beam search causes it to increase. Nevertheless, SeA-GuiDe

Model		BLEU	METEOR	ROUGE	CIDEr	SER ↓
S2S		0.519	0.388	0.631	2.531	2.55%
DT		**0.536**	**0.394**	**0.640**	**2.700**	1.68%
K&M		0.485	0.380	0.592	2.454	***0.46%***
T5-small	GS	0.519	0.387	0.631	2.647	1.65%
	BS	0.540	0.392	0.636	2.685	0.95%
	SA	**0.541**	**0.393**	**0.637**	**2.695**	**0.24%**
	SG	**0.541**	**0.393**	**0.637**	**2.695**	0.49%
BART-base	GS	0.524	0.386	0.635	2.629	2.70%
	BS	0.544	0.393	**0.639**	2.679	2.02%
	SA	**0.547**	**0.394**	**0.639**	**2.704**	**0.39%**
	SG	0.545	0.393	**0.639**	2.698	1.07%
T5-base	GS	0.527	**0.394**	**0.639**	**2.682**	0.61%
	BS	0.534	**0.394**	0.636	2.664	0.66%
	SA	**0.536**	**0.394**	0.637	2.672	**0.19%**
	SG	**0.536**	**0.394**	0.637	2.670	0.46%
BART-large	GS	0.508	0.378	0.616	2.452	5.50%
	BS	0.535	0.391	0.628	2.612	1.78%
	SA	**0.538**	**0.394**	**0.631**	**2.659**	**0.27%**
	SG	0.533	0.391	0.627	2.613	1.41%

Table 6.3: Models tested on the ViGGO dataset using different decoding methods: greedy search (GS), beam search with no reranking (BS), beam search with slot-aligner reranking (SA), and SEA-GUIDE (SG). Baselines compared against are Slug2Slug (Juraska et al., 2019) (S2S), DataTuner (Harkous et al., 2020) (DT), and Kedzie and McKeown (2020) (K&M). The best results are highlighted in bold for each model. SER scores of baselines reported by the authors themselves, rather than calculated using our slot aligner, are highlighted in italics, and they do not correspond exactly to our SER results.

Model		BLEU	METEOR	ROUGE	CIDEr	SER ↓
S2S		0.662	0.445	0.677	2.262	0.91%
S_1^R		**0.686**	**0.453**	**0.708**	**2.370**	N/A
K&M		0.663	**0.453**	0.693	2.308	***0.00%***
T5-small	GS	0.670	**0.454**	0.692	2.244	1.60%
	BS	0.667	0.453	**0.694**	**2.361**	2.85%
	SA	**0.675**	0.453	0.690	2.341	**0.02%**
	SG	0.675	0.453	0.690	2.340	0.04%
BART-base	GS	0.667	**0.454**	0.694	2.276	1.97%
	BS	0.670	**0.454**	**0.701**	**2.372**	3.39%
	SA	**0.680**	0.453	0.695	2.350	**0.02%**
	SG	**0.680**	0.453	0.695	2.347	0.08%
T5-base	GS	0.668	**0.459**	0.692	2.282	1.85%
	BS	0.667	0.453	**0.697**	**2.387**	3.94%
	SA	**0.682**	0.454	0.691	2.375	**0.03%**
	SG	**0.682**	0.454	0.691	2.374	0.05%

Table 6.4: Models tested on the E2E dataset, compared against the following baselines: Slug2Slug (Juraska et al., 2018) (S2S), S_1^R (Shen et al., 2019), and Kedzie and McKeown (2020) (K&M).

improves on both, while slightly boosting the other automatic metrics as well.

The almost twice-as-large BART-large model performs rather poorly in our experiments, in fact, significantly underperforming its smaller variant.[5] We therefore refrain from drawing any conclusions for this model, although SEA-GUIDE offers a definite improvement in SER over simple beam search.

6.3.3 Domain Transferability

We achieve similar results when evaluating across domains. Table 6.4 shows that using SEA-GUIDE with all three models fine-tuned on E2E reduces the SER

[5]We observed that it frequently misrepresents names, such as "Transportal Tycoon" instead of "Transport Tycoon", which we think may be the consequence of the extremely small size of the ViGGO training set relative to the model's size.

Model		BLEU	BLEU_R	METEOR	SER ↓	SER_E ↓
SCG		N/A	0.308	N/A	***0.53%***	N/A
K&R		N/A	**0.351**	N/A	N/A	1.27%
T5-small	GS	**0.367**	**0.351**	**0.325**	1.15%	1.36%
	BS	0.359	0.344	0.323	1.06%	1.19%
	SA	0.360	0.344	0.323	**0.41%**	**0.63%**
	SG	0.360	0.344	0.323	0.60%	0.85%
BART-base	GS	**0.372**	**0.356**	**0.326**	1.18%	1.17%
	BS	0.363	0.346	0.323	1.12%	1.02%
	SA	0.364	0.347	0.324	**0.40%**	**0.60%**
	SG	0.363	0.347	0.323	0.63%	0.72%

Table 6.5: Models tested on MultiWOZ, compared against the following baselines: SC-GPT (Peng et al., 2020) (SCG) and Kale and Rastogi (2020) (K&R).

down to almost zero, with performance for the other metrics comparable to the state-of-the-art baseline.[6] In fact, SEA-GUIDE is nearly as effective at reducing errors in this dataset as the heuristic slot aligner (SA). Table 6.5 compares our models against two recent baselines on the MultiWOZ dataset, where the effectiveness of SEA-GUIDE on SER reduction is comparable to that on the ViGGO dataset. All in all, on both the E2E and the MultiWOZ dataset, our models equipped with SEA-GUIDE for inference perform similarly to the best baselines for both SER and the other metrics *at the same time*, whereas the baselines individually perform well according to one at the expense of the other.

6.3.4 Slot Error Detection Examples

Table 6.6 shows several utterances generated for corresponding input MRs in the video game domain, along with the errors SEA-GUIDE detected, if any. In the first example, all slots are correctly mentioned, and SEA-GUIDE agrees. This

[6]We were unable to successfully train BART-large on E2E due to the memory limitations of our computational resources.

utterance was ultimately selected during reranking over the beam search's choice, "The Room is an excellent first person point-and-click puzzle game.", which has one of the genres omitted.

The second example again showcases a successful identification of all slot mentions by SEA-GUIDE, this time in an utterance where our heuristic slot aligner incorrectly found an error in the HAS_LINUX_RELEASE slot mention (presumably confused by the negation shortly before the keyword "Linux").

The third example shows that SEA-GUIDE is not confused by the difference between "excellent" (input slot value) and "amazing" (the corresponding slot realization), and it correctly identifies that the PLAYER_PERSPECTIVE slot is not mentioned.

In the fourth example, SEA-GUIDE fails to recognize the incorrect mention of the HAS_LINUX_RELEASE slot. This is the most common mistake our decoding method makes.

The final example does not contain any errors, yet the heuristic slot aligner was confused by the negation formulation (as "I wish it was") and identified the AVAILABLE_ON_STEAM slot as incorrect, while SEA-GUIDE did not. This demonstrates the robustness of SEA-GUIDE beyond exact matching. Yet despite the individual slots being mentioned semantically correctly, the generated utterance as a whole does not sound quite right. A better utterance would be something along the lines of "Mirror's Edge Catalyst is a disappointment, I really wish they released it on Steam as well.".

All in all, SEA-GUIDE chooses semantically correct utterances that are fluent and adequate, except for the rare case like in the last example.

give_opinion (NAME[The Room], RATING[excellent], GENRES[indie, point-and-click, puzzle], PLAYER_PERSPECTIVE[first person])
The Room is an excellent indie point-and-click puzzle game. The first person perspective makes it easy to play. **Errors:** N/A
verify_attribute (name[Small World 2], RATING[poor], HAS_LINUX_RELEASE[yes])
You said earlier that you disliked Small World 2. Are you generally not fond of Linux games? **Errors:** N/A
request_explanation (RATING [excellent], GENRES [adventure, hack-and-slash, indie], PLAYER_PERSPECTIVE [third person])
What is it about indie hack-and-slash adventure games that you find so amazing? **Errors:** PLAYER_PERSPECTIVE
inform (NAME[RollerCoaster Tycoon], RATING[good], GENRES[simulation, strategy], PLATFORMS[Xbox, PC], AVAILABLE_ON_STEAM[no], HAS_LINUX_RELEASE[yes], HAS_MAC_RELEASE[no])
RollerCoaster Tycoon is a good simulation strategy game for Xbox and PC. It is not available on Steam, Linux, or Mac. **Errors:** N/A (incorrect)
give_opinion (NAME [Mirror's Edge Catalyst], RATING [poor], AVAILABLE_ON_STEAM [no])
Mirror's Edge Catalyst is a pretty bad game, I really wish it was available on Steam. **Errors:** N/A

Table 6.6: Examples of slot-mention tracking results using SEA-GUIDE. Each gray row is the input MR for the corresponding utterance in the subsequent row.

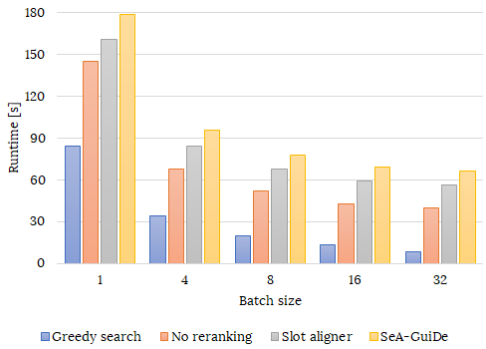
6.4 Discussion

In the previous section, we showed that SEA-GUIDE is highly effective at reducing semantic errors across different models and domains, and that without compromising on the generated utterances' fluency. On datasets other than E2E, it does not quite match the performance of beam search combined with our slot aligner-based reranking, but then again, the slot aligner is a hand-crafted tool with complex rules, requiring a good deal of domain knowledge, and suffering thus significantly in scalability. While these two decoding methods have a lot in common – both being based on beam search and subsequent candidate reranking – their difference lies in the identification of slot mentions; SEA-GUIDE identifies them *automatically* during the decoding, utilizing the model's cross-attention weights at each step, as opposed to relying on string-matching rules post decoding, which need to be extended for any new domains.

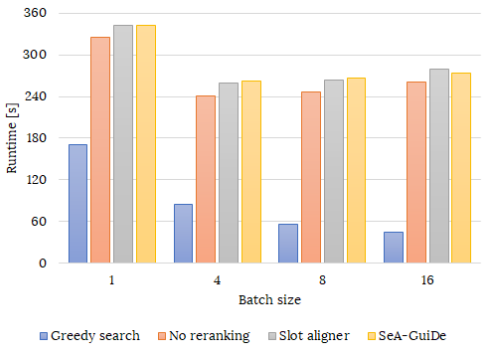
6.4.1 Inference Performance

Despite working conveniently out of the box, SEA-GUIDE would ideally not come with a significant computational overhead caveat. In order to assess this, we measure the inference running time of the T5-small model fine-tuned on ViGGO. For all beam search-based methods (including SEA-GUIDE), the beam size was set to 10, and early stopping was enabled.

The results in Figure 6.5a show a distinct but expected overhead across all batch sizes when running inference on a GPU. The overall increase in runtime is 11–18% over beam search with slot aligner-based reranking, which is the method computationally most similar to SEA-GUIDE, as it too involves reranking on top



(a) Inference using a GPU (RTX 2070 with 8 GB of memory).



(b) Inference using a CPU (8-core Ryzen 7 2700X with 32 GB of RAM).

Figure 6.5: Running time of T5-small performing inference on the ViGGO test set using different decoding methods and batch sizes. "No reranking" stands for simple beam search, while "Slot aligner" denotes beam search with slot aligner-based reranking. Model and data loading is excluded from the running times.

of beam search. The slot aligner-based reranking itself adds a constant amount of 16 seconds on top of simple beam search, which corresponds to an 11–40% increase for the range of batch sizes in the plot.

When performing the same inference on a CPU, on the other hand, the overhead SeA-GuiDe introduces to beam search is no greater than that of the slot aligner-based reranking (see Figure 6.5b). This suggests that further optimization of SeA-GuiDe for GPU, especially by minimizing the communication between the GPU and the CPU during the decoding, could bring the overhead of SeA-GuiDe inference on a GPU down to the same level as that of the slot aligner-based reranking.

Considering the large improvement in semantic accuracy the SeA-GuiDe method delivers in the tested models, we deem the observed computational overhead reasonable and acceptable.

6.4.2 Limitations of SeA-GuiDe

SeA-GuiDe's ability to recognize slot errors is limited to missing and incorrect slot mentions, which are the most common mistakes we observed models

to make on the data-to-text generation task. Duplicate slot mentions are hard to identify reliably because the decoder inherently pays attention to certain input tokens at multiple non-consecutive steps (such as in the example in Figure 6.1b). And arbitrary hallucinations are entirely beyond the scope of this method, as there is no reason to expect cross-attention to be involved in producing input-unrelated content, at least not in a foreseeable way.

As we see in example #4 in Table 6.6, Boolean slots occasionally give SEA-GUIDE a hard time, as the decoder appears not to be paying a great deal of attention to Boolean slots' values throughout the entire decoding in many cases. We plan to investigate if the performance can be improved for Boolean slots, perhaps by modifying the input format or finding a more subtle slot mention pattern.

6.5 Summary

In this chapter, we presented a novel decoding method, SEA-GUIDE[7], that makes a better use of the cross-attention component of the already complex and enormous pretrained generative language models to achieve significantly higher semantic accuracy for data-to-text NLG, while preserving the otherwise high quality of the output text. It is an automatic method, exploiting information already present in the model, but in an interpretable way. Here we summarize the strengths of SEA-GUIDE:

- It drastically reduces semantic errors in the generated text (shown on the E2E, ViGGO, and MultiWOZ datasets);
- It is domain- and model-independent for encoder-decoder architectures with

[7]Our SEA-GUIDE code is available at: https://github.com/jjuraska/data2text-nlg

cross-attention, as shown on different sizes of T5 and BART;

- It works out of the box, but is parameterizable, which allows for further optimization;
- It adds only a small performance overhead over beam search decoding;
- Perhaps most importantly, it requires no model modifications, no additional training data or data preprocessing (such as augmentation, segmentation, denoising, or alignment), and no manual annotation.

Although SEA-GUIDE is not as effective at detecting semantic errors as our slot aligner described in Chapter 3, it is a significantly more scalable method that offers large gains in model outputs' faithfulness to their respective inputs with minimal effort.

Chapter 7

Diversity-Promoting NLG Inference

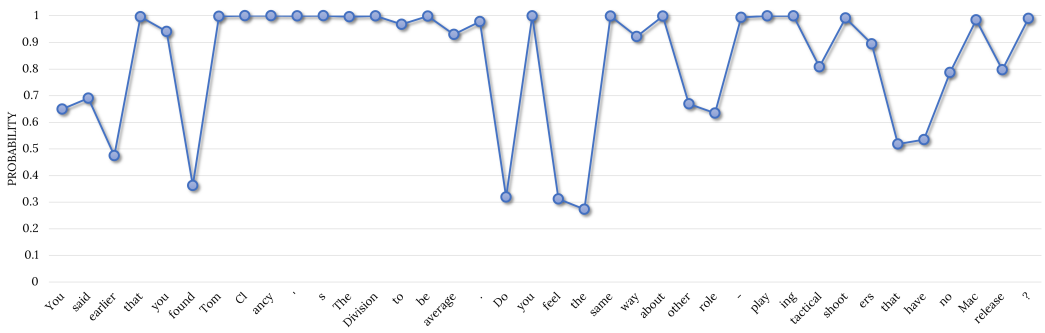
7.1 Motivation

In Chapter 5, we presented two different methods of giving a neural NLG system greater stylistic control. By limiting the training set to only examples that exhibit certain properties, we saw that a neural model was able to produce utterances with the same properties. Our experiments with the input data annotation method also showed a positive impact on the model's ability to generate utterances with a specific style or using a specific sentence structure, without an unreasonable impact on the semantic accuracy. This suggests that, when the model is trained on a large and varied training set, it *chooses* not to generate more interesting utterances, presumably because they are seen in fewer contexts during training and thus the model is less confident about them during inference. This is reflected by the lower probabilities the model assigns to less common words, phrases, and ultimately discourse relations.

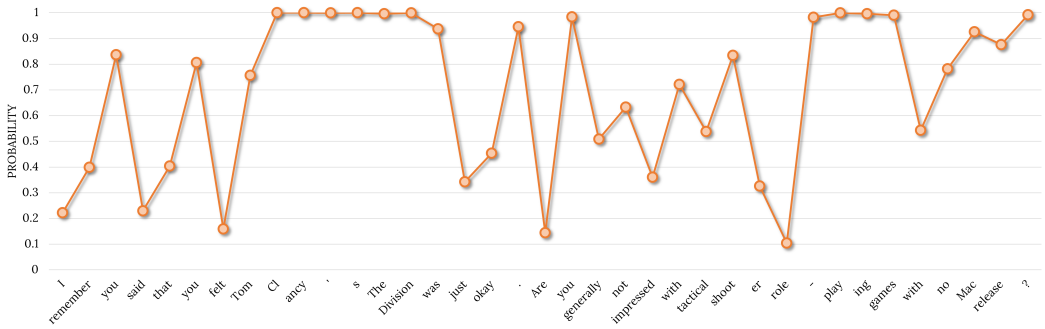
In other words, while a neural model may default to producing utterances that look rather generic, since it is most confident using language commonly seen

during training, it does not mean it is not capable of producing more complex and varied utterances, perhaps with the help of a different decoding strategy. If at any time step of the model's inference we look at the probability distribution of the next token, computed by the decoder, we would sometimes see that the model is considering multiple tokens that are more or less equally reasonable in the given context. Figure 7.1 shows three examples of a whole generated sequence with the probability of each token.

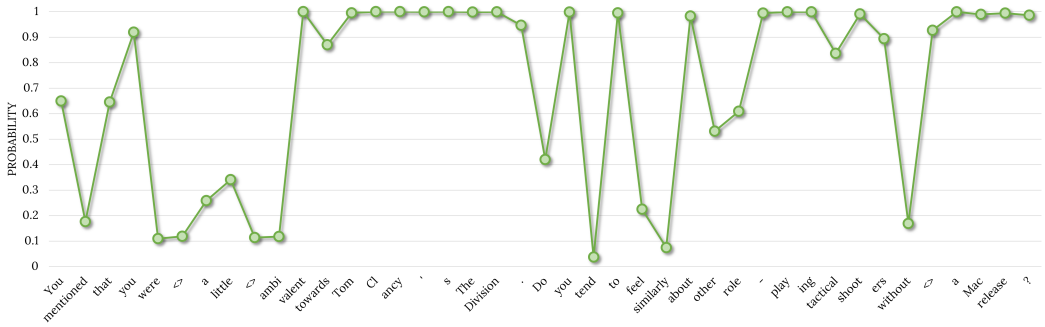
What we can immediately observe in the plots is that the decoder is highly confident about the next token whenever it is generating a common word or phrase, such as "role-playing" or the name of a game. In many other places, the probability – and with it the model's confidence – dramatically drops, as there are typically several words that could coherently continue the sentence in the given context. For example, in the first utterance (Figure 7.1a) after "Do you feel", instead of "the same way about" the decoder could have decided to continue with "kind of the same way", "similarly about" or "that way about", but also a more different formulation, such as "like most tactical shooters . . . are not that fun in general?". The probability of the token "the" as the next token is less than 30% in this context, as there are multiple perfectly valid tokens the decoder could have chosen instead. It is easy to see that the decisions the decoder makes at these points can entirely change the structure of the remainder of the utterance. Take, for instance, the very beginning of the third utterance (Figure 7.1c). It is with a much lower confidence that the model picked "mentioned" as the second token than the token "said" in the first utterance (0.18 vs. 0.69), yet it resulted in an entirely correct and coherent utterance, and that with a more interesting realization of the RATING[average] slot. Most standard inference algorithms would, nevertheless, prefer the first utterance over the other. And why



(a) Greedy search.



(b) Nucleus sampling with $p = 0.8$, example #1.



(c) Nucleus sampling with $p = 0.8$, example #2.

Figure 7.1: Token probabilities in 3 different utterances generated by a T5-small model for the same input MR with different inference methods. Note on T5's tokenizer: we use <> to denote T5's special token that represents a standalone word boundary (typically represented as a space once decoded), although T5's tokens often include a word boundary symbol at the beginning, which we omitted for better readability. For example, in the token pair ("shoot", "ers"), the former would have it while the latter would not, indicating that the two tokens should form one word.

is that?

The simplest approach to inference is *greedy* decoding, which picks the token with the highest probability at each time step until it reaches the end of the sequence (typically by generating the special end-of-sequence token). As is the case with greedy algorithms in general, this does not guarantee that the utterance is one with the overall highest probability though.[1] A more robust approach, and most commonly adopted one in sequence-to-sequence NLG models, is *beam search*. At each time step, it keeps a pool of n best candidate partial sequences, where n is typically less than or equal to 10. Though this strategy can help the decoder find sequences with a higher overall probability, presumably corresponding to better utterances, it still focuses on locally optimal choices. In other words, it is not able to anticipate a lower-probability partial sequence ending up being a more probable whole sequence than others, and often ignores it. In theory, errors from these short-sighted decisions, referred to as *myopic bias* (He et al., 2017), during inference could lead to a relatively poor final utterance even if we held a perfect NLG model.

With the now widespread use of encoder-decoder models in NLG, there has been certain effort to address the issue of myopic bias during inference in various NLG tasks in recent years. Li et al. (2016b) propose a modified beam search in which they penalize partial sequences that originate from the same ancestors, i.e., utterances that share the same prefix, whereby they enforce diversity among the final candidates. Li et al. (2017a) and He et al. (2017) both developed a reinforcement learning method for predicting the future BLEU score of a partial sequence, and use it to guide the decoding at each time step. Similarly, the decoding strategy

[1]The probability of an utterance is determined as the product of the individual token probabilities in the sequence, but since this can become an extremely small number, a sum of their log-probabilities is typically used instead.

proposed in Wang et al. (2018) estimates the future reward of a partially generated sequence, but based on its probability of appearing in a wider beam. These alternative approaches are shown to be effective to a certain degree in promoting greater diversity among the generated candidates in abstractive summarization, dialogue response generation, as well machine translation.

An alternative approach to decoding in NLG relies on sampling. There are various sampling methods that can be used, but they have a common objective – to produce significantly more varied outputs. With sampling, instead of choosing a token that maximizes the utterance's probability at each time step, the decoder may pick a less probable token as long as it satisfies certain constraints. A constraint can be that the token is, for example, among a certain number of the most probable tokens ($\sim$ top-K sampling, described in Fan et al. (2018)), or among the most probable tokens within a certain cumulative probability mass ($\sim$ top-p/nucleus sampling, introduced in Holtzman et al. (2019)). If the constraints are strict, the output will end up not being very different from using the methods from the previous paragraphs. On the other hand, if the constraints are loose, the outputs have a good chance of introducing semantic errors and becoming incoherent. Finding a middle ground that would work consistently, leading to diverse outputs without semantic errors across all the possible inputs, is typically very difficult, if not impossible.

An ideal inference method for NLG would promote diversity, while maintaining a high quality – not necessarily probability though – in terms of fluency, coherence and semantics. Revisiting Figure 7.1, we can see that with greedy search the lowest probability of a token is slightly below 0.3, while the token probabilities regularly dip well below 0.2 in the two nucleus sampling examples. The log-probability of the utterance produced using greedy search is -9.42, while that of the other two

is −19.02 and −25.56, respectively, yet all three of them are fluent and semantically equivalent.[2] This shows that, while striving for the most probable utterance is a safe strategy likely to avoid errors, there is an abundance of opportunities to produce more interesting utterances if we do not optimize for the probability. In data-to-text NLG, there are often countless perfectly valid paraphrases for any given utterance, and, as we see, the range of their probabilities can be relatively wide. At the same time, in the same probability range, for every good utterance we will find one that is incorrect or incoherent, because one error will not drag the probability of an otherwise strong utterance down substantially. We therefore argue that, in order to tap the diversity potential, it is not safe to rely solely on sampling; not without some sort of supervision that would guide it around the "land mines" in the field of potentially superior utterances.

To this end, in contrast to the above methods that modify the beam search, predict the quality of a sequence based on its prefix, or use sampling on its own, we propose an approach based on Monte-Carlo Tree Search (Coulom, 2006). By using it in conjunction with a neural NLG model whose inference can be parallelized, we intend to take advantage of this algorithm's superior sampling properties, and make its application to inference feasible in real time. The three primary benefits we expect to obtain using this approach are: (1) a greater output diversity thanks to its guided sampling, (2) the ability to optimize for an arbitrary metric, such as semantic accuracy, and (3) a better overall output quality as a result of this method operating on whole sequences to make decisions, as opposed to making final decisions in a left-to-right fashion during decoding. In the following section, we describe the

[2]The utterance probability inherently decreases with its increasing length. However, all three utterances in this example are almost equally long (in terms of the number of tokens), so the length is not the reason for the large log-probability differences observed.

Monte-Carlo Tree Search method and our proposed application of it to the NLG task at hand.

7.2 Batch Monte-Carlo Tree Search Inference

We start this section by introducing the original Monte-Carlo Tree Search (MCTS) algorithm, as defined in Coulom (2006) and Kocsis and Szepesvári (2006), followed by a description of our proposed modifications. MCTS works with a search tree data structure, where edges correspond to *actions*, and nodes represent *states* the actions lead to. In our language generation scenario, an action can be understood as adding a token, and a state is simply a sequence of tokens, i.e., a partial utterance. An *end state* thus corresponds to a whole utterance, and we denote such nodes in our diagrams with a cross.

7.2.1 MCTS Algorithm

Classic MCTS consists of four phases which are repeated until the algorithm is terminated (see Figure 7.2). The terminating condition is typically set as the longest acceptable running time, or a fixed number of iterations. In the first phase, starting from the root node, an action is iteratively selected based on the tree policy, until a not fully explored node, i.e., a node with at least one possible action that has not been previously taken from the node, is encountered. Subsequently, this selected node is expanded in the second phase, which means an action is randomly picked and a corresponding new leaf node is added to the tree.[3] This leaf node then serves as the starting point for the playout simulation in phase three.

[3]Note that an action can also be picked which leads to an already existing node, in which case no node is added.

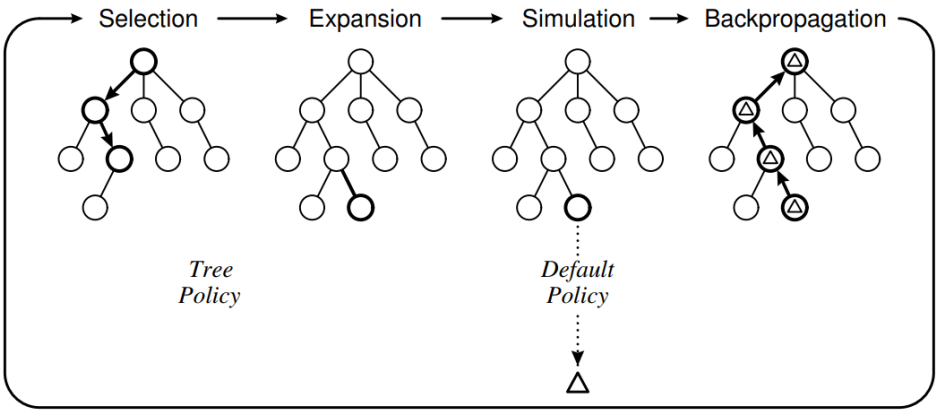


Figure 7.2: The 4 phases of the original MCTS, executed iteratively. Image credit: Browne et al. (2012).

A *light* playout is performed by simply choosing random actions until an end state is reached, whereas a *heavy* playout performs the action selections in an informed way, e.g., using heuristics. The playout's end state is then evaluated. Finally, in the fourth phase, the result is backpropagated from the leaf node where the simulation started back to the root. During the backpropagation, all nodes along the way to the root get updated based on the result, incrementing also their counters of how many times they have been visited. These two values serve then as an indicator of how good the subtree rooted in a given node is.

The *tree policy* used in the selection phase is very essential to MCTS. It is responsible for determining which child node is the most "urgent" and should thus be visited next. The policy is typically designed to balance exploitation and exploration. In other words, it guides the tree search to sample from subtrees with good potential, but not forgetting to visit unexplored branches of the tree too, as those can potentially lead to even better finds. The most popular choice of a tree policy is Upper Confidence Bound for Trees (UCT) (Kocsis and Szepesvári, 2006),

defined for a node v as follows:

$$\text{UCT} = \underset{v' \in v.\text{children}}{\text{argmax}} \frac{Q(v')}{N(v')} + c\sqrt{\frac{2 \ln N(v)}{N(v')}} ,$$

where $Q(v)$ is the accumulated reward in v, and $N(v)$ is its visit count. The first term in the formula corresponds to exploitation, as it is high for a child node with a high average reward. The second term is high for a child node with few visits compared to v, and corresponds thus to exploration. The exploration coefficient c then serves to balance the amount of exploitation and exploration to be performed in the tree search. The UCT tree policy picks the child node of v that maximizes the value calculated by the formula.

7.2.2 Batch Modification

Our proposed modification of the MCTS algorithm differs from the original version primarily in three aspects:

1. It performs a large batch of simulations at once,
2. It involves sampling in the tree policy so that the simulations would start from different nodes (see Figure 7.3),
3. Nodes corresponding to all states passed through in a playout are added to the tree instead of just the start node.

Additionally, due to the parallel nature of the search, the four stages are performed in a slightly different order. The following paragraphs describe each of the phases in more detail.

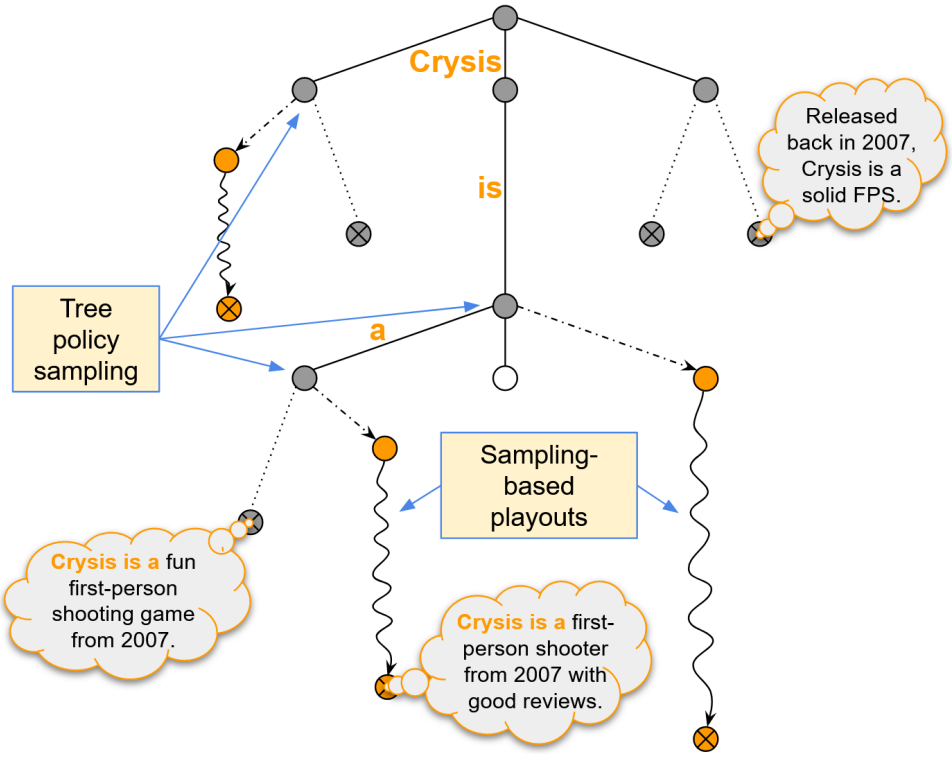
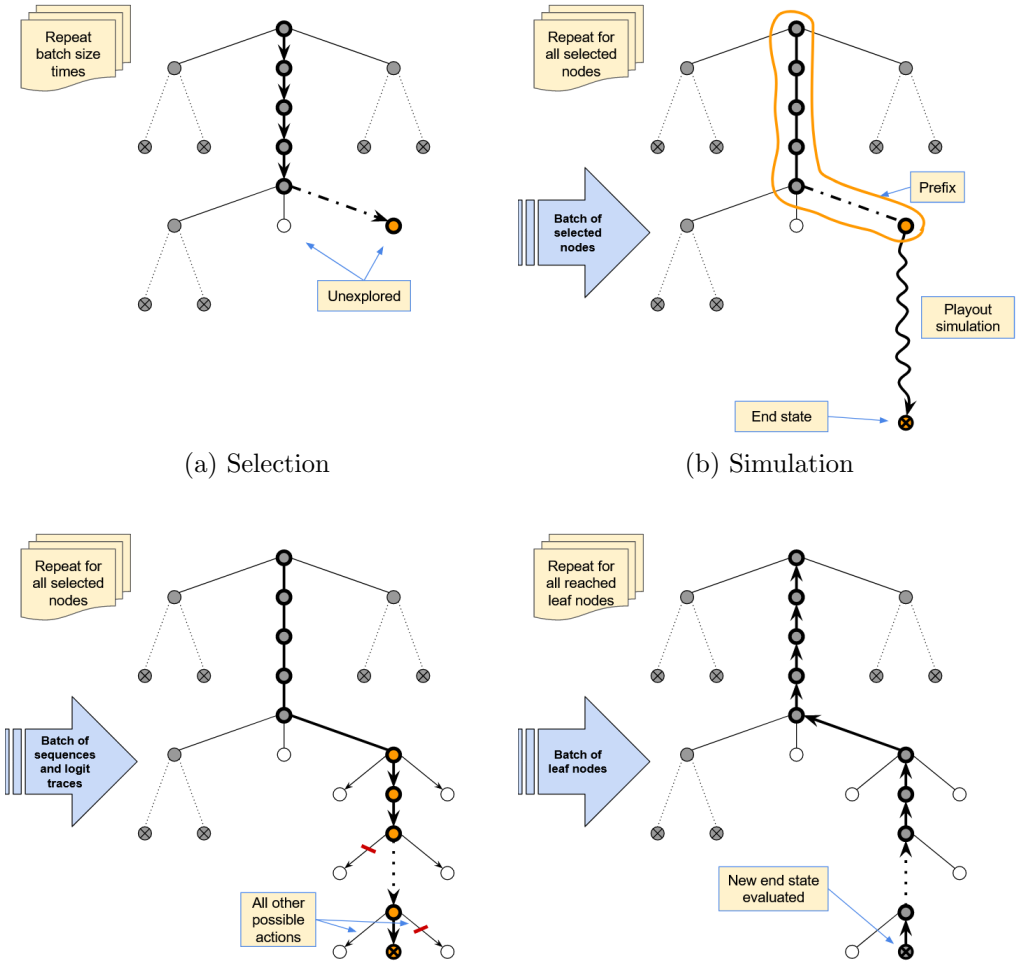


Figure 7.3: Illustration of Batch-MCTS sampling utterances from different parts of the search tree in parallel. Some of the nodes representing an end state have a possible utterance indicated according to the sequence prefix.

Selection. The first phase differs from the original only in that it is repeated B times, where B is the batch size. It identifies a batch of nodes in the tree that should be further explored. These can be the roots of promising subtrees (based on previous simulations), as well as unexplored nodes whose subtrees have not yet been visited during the search (see Figure 7.4a). In order to obtain a batch of various nodes, it is crucial that the tree policy involves sampling, or else the same node will be selected over and over B times. At the same time, it is okay for the same node to appear in the batch more than once, considering the subsequent simulations are assumed to involve sampling too and thus to return different sequences even when executed from the same start node.



(a) Selection (b) Simulation

(c) Expansion (d) Backpropagation

Figure 7.4: The 4 phases of our proposed Batch-MCTS algorithm. Nodes corresponding to an end state are crossed. Empty white nodes represent known states (i.e., the probability of the action leading to them is known) that have not yet been visited. Edges in (c) that are crossed out indicate pruned actions.

Simulation. For the batch of selected nodes (whose states correspond to sequence prefixes), this phase performs simultaneous heavy playouts until the end state is reached in each sequence (see Figure 7.4b). The heavy playouts are where the trained NLG model we run the inference for comes into play. It generates a batch of utterances for the same input, either from the very beginning or as a continuation of a partial utterance provided. The model thus needs to support the following: (1) output sequence prefix forcing, so as to be able to generate conditioned on an input MR as well as a given utterance beginning, (2) sampling in the decoder, in order to generate a batch of different utterances (this can be temperature-based, nucleus, or other sampling), and (3) batch decoding mode.

Expansion. Since our simulation performs heavy playouts, we add nodes corresponding to *all* actions taken during the playouts to the tree, starting from the selected nodes. Besides the actual actions taken in a playout, we also add their sibling nodes corresponding to other possible actions to the tree, since the batch decoding in the previous phase will have calculated the probability of all possible actions from each state during a playout. In this phase, the algorithm thus has an opportunity to prune the search space by removing – or rather not creating – nodes led to by actions with a probability below a certain threshold. If using top-K or nucleus sampling, the sibling nodes to add can simply correspond to the limited set of actions considered in the sampling. See Figure 7.4c for an illustration of the above concepts.

Backpropagation. At the beginning of the final phase, the candidate utterance in each of the new leaf nodes after the expansion is scored. The score can be calculated

by any automatic evaluation metric that solely relies on the generated utterance and its corresponding MR to assess the utterance's quality. Subsequently, all nodes between the leaf node and the root have their rewards and visit counts updated (see Figure 7.4d). This is repeated for each leaf node corresponding to the end state of a playout from the simulation phase.

Batch-MCTS iterates over these four phases a fixed number of times I, specified as an input parameter. Similarly, the batch size B is another input parameter. The algorithm keeps a pool of n-best candidate utterances discovered, and updates them at the end of each iteration. Unlike in the standard MCTS, where the tree is updated with each "run" (performing one playout), in Batch-MCTS the tree remains unchanged for all runs within the same batch; that is until the end of an iteration, when it changes considerably.

7.2.3 Discussion

Due to potential node access conflicts, only the first two phases can be fully parallelized. Nevertheless, it is the simulation phase that is the most compute-intensive, and all the others are expected to only take a fraction of the time. With an access to a GPU, the heavy playouts in the simulation phase can be executed concurrently, since they are entirely independent. Therefore, it is important for the underlying NLG model to support batch decoding. Depending on the GPU specifications, however, the batch parallelization will become less effective beyond a certain batch size. For optimal performance, it is therefore important to configure the batch size and number of iterations according to the hardware used.

Finally, we should note that the purpose of the Batch-MCTS inference,

unlike many other applications of MCTS, is not to merely determine what the best action might be in a given state based on sampled simulations, but to find a whole sequence of actions (i.e., a whole utterance) that is as good as possible in the large space of possible sequences. This is a consequence of the heavy playouts, which allow for a highly informed search rather than random, and the fact that our task is equivalent to a one-player game. The latter means that we do not need to account for adversarial moves following each of the selected actions. Instead, if the tree search discovers a strong candidate utterance, it is a real one, not a hypothetical one that could still be prevented from being generated.

7.3 Summary

This chapter motivated the need for a new NLG inference method that would be able to take advantage of more of the knowledge a model acquires from the training data. Standard inference methods typically either (1) focus on maximizing the probability of the generated sequence of tokens, which often results in dull and repetitive utterances, or (2) they try to stimulate diversity through sampling, which inevitably leads to incoherence and errors in utterances whose rate increases with the diversity. Since they perform the generation in a single left-to-right pass, their token selection at each time step is more or less final. For sampling methods, this means that once a mistake is made, it cannot be undone. For greedy and beam search decoding, this results in a bias towards the most common formulations at utterance beginnings, even if it leads to an overall less desirable utterance by the end of the decoding. Moreover, we showed there is, in fact, likely an abundance of opportunities among lower-confidence model outputs for more interesting utter-

ances that are semantically correct and perfectly fluent. Naturally, if the utterance probability drops too much, the output can be expected to contain disfluencies or be missing a fact or two.

We proposed a method based on Monte-Carlo Tree Search (MCTS) as an alternative that may be able to realize an NLG model's potential better. In contrast to the standard methods, it performs inference in multiple passes, evaluating the candidates at the end of each and improving on them in the next. It prompts the trained model to generate batches of largely adequate and varied candidates, and uses an automatic metric to find the best among them and guide the search to the most promising utterance prefixes. While model outputs in NLG are traditionally evaluated using reference-based metrics, such as BLEU, which compare the generated texts with human-authored reference utterances, for evaluating candidate utterances in MCTS we do not have the luxury of using references. We must use a referenceless metric that relies on the input MR and the utterance itself only. Taking advantage of parallelism, the proposed Batch-MCTS inference is expected to be computationally feasible in real time with the right hardware.

In the next chapter, we describe and test different ideas for an automatic referenceless metric that distinguishes between good and bad candidates among utterances generated from the same MR. The chapter after that then evaluates the performance of the Batch-MCTS inference with multiple variants of the referenceless metric integrated for state evaluation.

Chapter 8

Referenceless Automatic Evaluation Metric for Data-to-Text NLG

In Chapter 7, we described in detail a proposed inference method, Batch-MCTS, for data-to-text NLG. This method requires guidance in its search for good candidate utterances. In this chapter, we will discuss a novel referenceless metric that we incorporate into Batch-MCTS to guide it toward fluent and semantically accurate utterances.

A data-to-text NLG model is typically trained on a parallel corpus of inputs and corresponding reference texts. During inference, however, it only has a previously unseen input (MR in our case), and the knowledge of the language structure and correspondence to inputs that the model acquired from the training set, to guide its generation. It does not have access to reference utterances beyond training. That extends to Batch-MCTS too when it is used by the system to generate utterances. It is therefore essential for our proposed inference method that we develop a robust referenceless metric that can distinguish good utterances candidates from inadequate ones.

A sequence-to-sequence model learns during training to produce utterances that are relevant and of generally expected structure, i.e., utterances that have content corresponding to the input and mention it in an expected sense, which is represented by the generated utterance's conditional probability computed by the model. If the conditional probability is too low, the utterance is most likely incorrect and inadequate. However, even among the ones with a high probability according to the model we would still find substantial differences in various aspects, including if and how the input content is mentioned, and we would find utterances of various styles and with various sentence structures. At the same time, less common but perfectly valid utterance formulations will typically be generated by the model with a lower confidence because of their lower prevalence in the training data. As a consequence, such utterances will also have a lower conditional probability, but we do not want to ignore them. In fact, these are often more natural-sounding outputs than those the model predicts with the highest confidence. This is supported by the fact that humans do not strive to simply use the most predictable words in the given context when composing a sentence (Holtzman et al., 2019), instead they may try to use more varied language that avoids sounding repetitive, as we discussed at the beginning of Chapter 7.

It is our goal to promote this natural language diversity in generated utterances using Batch-MCTS inference, while maintaining the high adequacy as determined by the trained encoder-decoder model. In order to achieve that, we use the model's conditional probabilities to guide the tree search to largely adequate utterance candidates, and then eliminate any with flaws as identified using an appropriate evaluation metric. While for the evaluation of an NLG system's performance reference-based metrics are typically used – which compare outputs against

human-written references – during the inference we only have the generated utterance itself, along with the corresponding input, to determine how it compares against other highly probable candidates generated by the model.

There are several different approaches to designing such a referenceless metric that we consider and describe in the next section. We then proceed to thoroughly analyze the performance of a new method that we propose for automatic semantic accuracy evaluation, an important component of a robust referenceless metric for data-to-text NLG.

8.1 Referenceless Metric Components

The most common flaws in generated utterances with a high conditional probability are missing and incorrect slot realizations, as well as hallucinated content, the latter being a more frequent issue in large pretrained models that are nowadays commonly used for NLG tasks. Hence, the metric must account for this aspect and penalize utterances with any such semantic inaccuracies. However, optimizing aggressively for slot realization accuracy might lead to preferring utterances with deficiencies in fluency and coherence. Especially if we lower our expectations of the conditional probability – with the goal of encouraging even more diverse utterances – the "largely adequate" candidates may even contain occasional syntactic errors. Therefore, a second useful component of the metric would be penalizing utterances for poor grammar and language in general. In this section, we provide more details and ideas on how these two components of a referenceless metric can be implemented.

8.1.1 Syntactic Fluency

We start by discussing the component responsible for syntactic fluency, which is completely independent of the input MR, and only focuses on the language aspect of the utterance. To assess an utterance's fluency and overall coherence, we propose using a language model (LM). By calculating the model's perplexity (PPL) on generated utterances, we can evaluate how confident the model is about the text being "good". The LM's notion of "goodness" depends on what data it was trained on, i.e., if it is trained on a small domain-specific dataset, it will only find those texts good that closely resemble those in the dataset, and any other text will have a high perplexity. On the other hand, if it is trained on millions of human-written documents (articles, books, web pages, etc.), the LM is expected to become very good at recognizing virtually any text being disfluent or incoherent, as long as it is in the same language as the data it was trained on.

Whenever there is a grammatical error or an incoherent phrase in a text, the LM will have a low probability associated with the offending word, which translates to a high word perplexity. The perplexity of a text $T = (w_1, w_2, \ldots, w_n)$ is formally defined as the exponentiated average negative log-probability of a sequence of words:

$$\mathrm{PPL}(T) = \exp\left\{-\frac{1}{n}\sum_{i=1}^{n}\log p_\theta(w_i|w_{<i})\right\} ,$$

where $p_\theta(w_i|w_{<i})$ is the probability of the i-th word in the context of all previous words $w_{<i}$ in the text, and these conditional probabilities are calculated by an LM with parameters θ. Internally, LMs typically work with tokens, which may or may not correspond to words, and are often smaller units, such as syllables or single characters.

We consider the following three ways of utilizing an LM for the purposes of our evaluation metric for syntactic fluency:

1. **Use a large pretrained language model.** This approach would make the metric domain-agnostic (at least as far as the syntactic fluency component is concerned), but it may suffer in accuracy when applied to domain-specific data. The LM may end up scoring well-formed utterances lower if they contain rare, domain-specific terms, phrases or formulations. This could possibly be balanced out by combining its perplexity with the NLG model's own perplexity scores, which will reflect the expected structure of utterances and terminology learned from the training set.

2. **Train a small in-domain model.** This is an opposite extreme of the previous approach, where an LM would be trained on a (typically small) in-domain dataset only, thus ensuring that the model does not penalize what is considered a good utterance in the given domain. Two major drawbacks of this approach are that (1) the sentence probabilities determined by the LM would not differ significantly from the generator model, which is essentially a conditional LM, as it would be trained on the same dataset, and (2) it would make the metric less scalable by requiring the LM to be retrained for every new domain/dataset.

3. **Fine-tune a large pretrained model on in-domain data.** Combining the benefits of both of the above approaches, this might provide the most adequate sentence scores. And while it addresses the first drawback described in the "small in-domain LM" approach, the second one would apply here as well, limiting the usability of the metric by requiring the LM to be fine-tuned

for different domains. Perhaps this could be, however, turned into a simple, streamlined procedure that would be executed as a one-time initialization of the metric before the first use in a new domain.

As for the model implementations, LMs pretrained on huge amounts of textual data in an unsupervised way have become ubiquitous over the past few years in the field of NLG, so we can take advantage of one of the numerous off-the-shelf models trained on hundreds of millions of sentences. We will primarily investigate the utility of transformer-based general-purpose pretrained LMs, such as GPT-2 (Radford et al., 2019) or BERT (Devlin et al., 2019). While it is possible that the fact that BERT taking advantage of bidirectional context encoding makes it unsuitable for an NLG task, it has been successfully adapted to sentence scoring by using masking appropriately, as in Shin et al. (2019) for automatic speech recognition candidate reranking.

8.1.2 Semantic Accuracy

Semantic accuracy of an utterance in data-to-text generation is a very important aspect in deciding whether the utterance is good or not. Some data-to-text generation tasks might involve content selection, wherein the model has to correctly decide which content from the input to use and which to ignore, depending on the context. In our case though, all of the information in the input MR is supposed to be realized in the corresponding generated utterance, as no additional context is provided.

Determining whether all slots in an MR were correctly realized in an utterance may, however, not always be an easy task for a model without a deeper understanding of the language and context. Table 8.1 shows a few examples of how

MR
verify_attribute (NAME [Little Nightmares], RATING [average], HAS_MULTIPLAYER [**no**])
Possible utterance #1
I know you found Little Nightmares merely okay. Would you say that is the case with all games that **don't have a multiplayer mode**?
Possible utterance #2
So you thought Little Nightmares was just an average game. I wonder, are you in general not a big fan of games **without multiplayer**?
Possible utterance #3
You mentioned earlier you weren't that fond of Little Nightmares. Do you feel the same way about most games that are **single-player only**?
Possible utterance #4
Hey, I remember you telling me that Little Nightmares was just meh. Do you **wish it** at least **had multiplayer** so you could play it with friends?

Table 8.1: MR with multiple utterance examples, each of which correctly mentions the HAS_MULTIPLAYER slot (and its negative polarity) in a very different way.

the mention of a particular slot may vary for the same MR. One can imagine it takes fairly complex rules to capture certain slot mentions reliably using a rule-based system, yet statistical and neural models that are trained to identify slot mentions on one particular dataset may also have a hard time correctly recognizing mentions in utterances that deviate from the distribution of formulations in the training set. The situation can become even trickier when, for example, multiple Boolean slots with different polarities get mentioned together or aggregated in the utterance, such as in "... was released for Xbox and PC, but not Steam, with multiplayer support. While the game has a Mac release, a Linux version is not available.".

There are four automatic methods that we consider to measure the accuracy of slot realization in an utterance given an input MR:

1. **Heuristic slot aligner.** Our automatic slot aligner, described in Chapter 3, is designed to identify words and phrases in an utterance that correspond to mentions of slots in the input MR. Iterating thus over all the slots in the MR, we can easily determine if all of them were correctly realized in a generated

utterance. However, the slot aligner will not recognize hallucinations in an utterance, i.e., information not grounded in the input MR. Although in theory we could modify the slot aligner to recognize information in an utterance that corresponds to a slot not present in the MR – by reversing and adjusting its rules and semantic dictionary for certain slot types – it would be limited to just the dataset's ontology and it would still not be capable of recognizing general hallucinations. And hallucinations are more likely to be encountered in this era of large pretrained language models than a mention of an extra slot from the ontology. In fact, hallucination detection is in general a rather difficult problem, unless the dataset's language is very rigid or template-based. Aside from the lack of ability to identify hallucinations, the main drawback of the heuristic slot aligner is the need for manually populating and extending a semantic dictionary before using it in a new domain.

2. **Statistical aligner.** Training a statistical aligner, such as one of those popular in machine translation before deep learning models took over (Brown et al., 1993; Dyer et al., 2013; Gelling and Cohn, 2014), could make this component more scalable. A related approach introduced in Wiseman et al. (2018) might also be applicable; they use a neural hidden semi-Markov model to associate phrases with the latent states that frequently generate them. Nevertheless, for each new domain a statistical aligner would have to be retrained.

3. **Neural text-to-data model.** A natural attempt to improve on a statistical aligner would be to build a deep learning model with the same purpose. However, instead of finding a word/phrase alignment, the model could be trained to predict the MR, or simply a list of slot names, from an utterance. In this

way the semantic accuracy of an utterance would be determined by comparing the generated MR with the original input MR. This is related to the concept of speaker-listener models, which was applied to the data-to-text NLG task in Shen et al. (2019). Although a neural model might be able to achieve a superior accuracy, it would make the metric even more cumbersome than a light-weight statistical aligner that can be trained faster and in a deterministic way. Additionally, both the statistical and the neural model's performance may suffer on data that does not resemble the distribution in the training set, which may be the case with utterances containing unusual formulations or disfluencies as a result of sampling being used in the model's inference method.

4. **Reference-based metrics with pseudo-references.** We propose a novel approach that takes advantage of existing reference-based metrics such as BLEU or BERTScore, i.e., metrics that compare a generated text with a reference text (or a set of them), typically written by humans, in order to assess how good an utterance produced by a model is. However, instead of human-authored references we compare the generated utterances directly against the input MRs, though in a modified format. There are various ways an MR could be linearized and presented to metric scorers as such a pseudo-reference, with the simplest being a concatenation of the slot values with spaces between them. While the metrics are expected to produce rather low scores on such text pairs, it is the relative score difference between a correct utterance and one that has a semantic error that matters. If a metric can reliably score good utterances higher against a pseudo-reference than utterances with errors, then we can use it as a heuristic to rank a pool of candidate utterances by semantic

accuracy. We provide further details on this approach, along with experiments and a thorough analysis of the suitability of various existing metrics for this purpose in Section 8.2.

These four approaches to automatically assessing semantic accuracy of utterances generated from MRs cover the spectrum of scalability, from predominantly domain-specific but highly accurate, to domain-agnostic but only approximate. Each of them has its pros and cons, as discussed in the above paragraphs, and each will have its own merit in a different setting, depending on the data-to-text NLG task at hand, the number of domains that should be supported, or where and how the generated texts are going to be used.

What they do have in common is that they all focus solely on the information in an MR being correctly realized in the corresponding utterance. Depending on the model (or the reference-based metric in the last approach), they may be able to take advantage of contextual language information, but none of them would identify an incoherent utterance with grammatical errors as long as it mentions all the slots. Luckily, this is exactly where the syntactic fluency component described in Section 8.1.1 comes into play.

8.1.3 Other Aspects of Generated Utterances

While we consider syntactic fluency and semantic accuracy to be the two most essential characteristics of a good generated utterance in the data-to-text NLG setting, there are certainly many other aspects it might be desirable to enforce in generated outputs and thus to evaluate them for. In the legal and medical domains, it might be the formalness of an utterance, whereas in dialogue systems it might be the style or personality an utterance exhibits and which should remain consistent

throughout a conversation. Nevertheless, these are domain- and task-specific and therefore, in this work, we focus on the two that are universally important.

8.2 Reference-Based Metrics With Pseudo-References

In this section we describe in detail our novel referenceless metric for assessing semantic accuracy of an utterance introduced in Section 8.1.2, and evaluate its effectiveness experimentally. This method uses standard reference-based metrics (e.g., BLEU, METEOR, or BERTScore) to evaluate the semantic quality of model-generated utterances by comparing them to *pseudo-references* automatically created from the corresponding MRs.

Pseudo-references can be composed from MRs in various ways. Since we will be using metrics that are meant to be used with natural-language texts, we need to make the pseudo-references look as much like a natural text as possible. At the same time, we need to keep the process simple and avoid introducing domain- or DA-specific phrases as "glue" between the slots. In fact, when DA type indications are present in a dataset's MRs, we drop them when creating pseudo-references, as they do not provide content information that could be matched by reference-based metrics.

Considering the MRs we work with are flat (i.e., not hierarchical)[1], the linearization into a pseudo-reference is straightforward and amounts to concatenating the slots names and/or values. Nevertheless, there are three aspects we experiment with in this process of creating pseudo-references:

1. Using slot values only vs. prepending slot names to the corresponding values.

[1]MRs with DA types encapsulating slot-value pairs, such as is the case in ViGGO and MultiWOZ, become flat after removing the DA types.

2. Comma vs. space as the value (or slot-value pair) separator.
3. Boolean slot handling, since their value is not meaningful on its own, and it would not be natural if a value like "yes" or "true" simply followed the slot name. We therefore opt for the following formulation of Boolean slots in a pseudo-reference:
    - When the slot value is positive, only the slot name is used in the pseudo-reference.
    - When the slot value is negative, we prepend "not" to the slot name, e.g., "not multiplayer".

    For Boolean slots we follow this format independently of whether the remaining slots have their names included in the pseudo-reference.

Table 8.2 provides a few examples of pseudo-reference formats we experiment with. Notice in example #3 that we verbalize the slot names when they are used in the pseudo-reference. That means we convert them to natural-language words or phrases. This is very dataset-dependent but can often be achieved by simply removing underscores from multi-word slot names and using single-word slot names without any conversion. For Boolean slots we opt for verbalizations that keep just the most salient part of the slot name, typically just a single word, e.g., "multiplayer" instead of "has multiplayer" for the slot name HAS_MULTIPLAYER. This in general results in more fluent negations when prepending "not" to the verbalized slot name.

Any slot name conversions we do here are identical to those that we already perform during MR preprocessing for model fine-tuning, since pretrained language models benefit from verbalized slot names in our experience, leading to a better

MR
inform (NAME [SpellForce 3], RELEASE_YEAR [2017], GENRES [real-time strategy, role-playing], PLAYER_PERSPECTIVE [bird view], PLATFORMS [PC], AVAILABLE_ON_STEAM [yes], HAS_LINUX_RELEASE [no], HAS_MAC_RELEASE [no])
Pseudo-reference variant #1 (space-separated, slot values only)
SpellForce 3 2017 real-time strategy, role-playing bird view PC Steam not Linux not Mac
Pseudo-reference variant #2 (comma-separated, full Boolean slot names)
SpellForce 3, 2017, real-time strategy, role-playing, bird view, PC, available on Steam, not has Linux release, not has Mac release
Pseudo-reference variant #3 (comma-separated, slot names included)
name SpellForce 3, release year 2017, genres real-time strategy, role-playing, player perspective bird view, platforms PC, Steam, not Linux, not Mac
Reference utterance example
SpellForce 3 is a 2017 role-playing real-time strategy with the traditional bird view. It was released for PC only. Although available on Steam, this game does not run on Linux or Mac.

Table 8.2: MR with multiple pseudo-reference examples, each using different composition rules, indicated in parentheses. For comparison with an actual utterance, the table also shows one reference utterance example from the ViGGO dataset.

performance. In the pseudo-utterance variant #2 in Table 8.2, it is easy to see how using full Boolean slot names may lead to disfluent formulations, as mentioned above.

8.2.1 Pseudo-Reference Perturbations

In order to evaluate the ability of reference-based metrics to detect semantic errors in generated utterances when comparing them to pseudo-references proposed above, we run a series of experiments with perturbed pseudo-references. By performing small modifications to the pseudo-references, we can simulate various errors that may occur in utterances generated by an NLG model, such as fact omissions or hallucinations. This will help us evaluate how sensitive different metrics are to such semantic errors when using pseudo-references instead of proper reference utterances. We perform the evaluation, using the following steps:

1. Calculate baseline metric scores by using a dataset's reference utterances as

predictions, and pseudo-references generated from corresponding MRs as references.

2. Create a set of systematically perturbed pseudo-utterances (see a detailed overview below), calculate metric scores using these as references, and measure the relative changes from the baseline scores achieved when using original pseudo-references.
3. Verify that the metrics produce very low scores when using pseudo-references generated from random MRs.

There are several common types of semantic errors that can be simulated by perturbations to the pseudo-reference (as opposed to modifying the utterance), avoiding thus the possibility of introducing incoherence in the utterance. Perturbations are performed by modifying one or more slots in the MR before concatenating it into a pseudo-reference. Below is a definition of all the perturbations we use in our evaluation, and which semantic error each of them corresponds to (Table 8.3 shows an example for each of them):

- **Insertion:** picks up to k random slots from the dataset's ontology that are not present in the MR[2], along with a random value for each of them, and inserts them at random positions in the MR.
  - Simulates a **fact omission** in the generated utterance.
- **Substitution:** replaces the value of k randomly chosen slots in the MR with a different value that the slots could possibly take on according to the dataset's ontology.
  - Simulates an **incorrect mention** in the generated utterance.

[2]The number of slots inserted may be less than k in case there are fewer than k slots in the ontology that can be added to the MR without repeating one of the MR's slot.

- **Deletion:** deletes up to k randomly chosen slots in the MR.[3]
  - Simulates a **hallucination** in the generated utterance.
- **Boolean negation:** inverts the polarity of k Boolean slots in the MR, if any are present.
  - Simulates an **incorrect polarity of a Boolean slot mention**, which tends to be a common mistake models make, especially when multiple Boolean slots are aggregated in the utterance.
- **Slot shuffle:** randomly reorders the slots in the MR.
  - There is no semantic difference the slot order in an MR implies. While MR-to-text datasets often come with slots in a fixed order within MRs, the task typically does not enforce any particular order in which the slots should be mentioned in the corresponding utterance. Figuring out a natural order of mentioning the slots is normally a part of the MR-to-text generation task, i.e., something NLG models are expected to learn along with *how* to mention them.

If k is greater than the number of slots and Boolean slots in an MR, the substitution and Boolean negation perturbations modify *all* the slots and Boolean slots, respectively. In our evaluation we limit k to 1, as modifying more than one slot at a time only makes the task of detecting slot errors easier for the metrics.

We also note that there is no straightforward automatic way to simulate duplication errors (i.e., repeated slot mentions) in the generated utterance without possibly affecting the fluency and grammaticality of the utterance, and we therefore omit this error type from our analysis. Nevertheless, a duplication can be loosely

[3] If the MR has n slots, we delete at most $n-1$ slots, even if $k \geq n$, otherwise the pseudo-reference would become empty.

Pseudo-reference	SpellForce 3 2017 real-time strategy, role-playing bird view PC Steam not Linux not Mac
Insertion	SpellForce 3 2017 **excellent** real-time strategy, role-playing bird view PC Steam not Linux not Mac
Substitution	SpellForce 3 2017 **adventure** bird view PC Steam not Linux not Mac
Deletion	SpellForce 3 2017 real-time strategy, role-playing PC Steam not Linux not Mac *("**bird view**" deleted)*
Boolean negation	SpellForce 3 2017 real-time strategy, role-playing bird view PC Steam not Linux Mac *("**not**" before "Mac" not present)*
Slot shuffle	Steam 2017 not Mac SpellForce 3 bird view real-time strategy, role-playing not Linux PC

Table 8.3: Examples of perturbed pseudo-references, with $k = 1$ (slot shuffle affects all slots irrespective of k). Perturbations are highlighted in blue bold font, or explained in parentheses, as appropriate.

considered a variant of a hallucination error, and results from the *deletion* perturbation experiments are therefore expected to be at least partially representative of the metrics' ability to recognize duplicate slot mentions.

Now that we have the perturbations in pseudo-references defined, let us have a look at how they correspond to errors in utterances and how they will allow us to measure a reference-based metric's performance in estimating semantic accuracy. What we would expect from a good semantic accuracy metric is a combination of several properties that it demonstrates consistently. Such a metric should be sensitive to fact omissions (i.e., demonstrate a high recall), but at the same time sensitive to hallucinations (i.e., demonstrate a high precision). It should be insensitive to slot mention order and robust to synonymous expressions, not scoring an utterance lowly for not using the words in the pseudo-utterance verbatim. A full list of desired properties and how to assess them by comparing utterances with perturbed pseudo-references can be seen in Table 8.4.

Desired metric property	Corresponding pseudo-reference criterion
Sensitivity to fact omissions (recall)	Score should drop significantly for insertions
Sensitivity to incorrect mentions (recall + precision)	Score should drop significantly for substitutions
Sensitivity to hallucinations (precision)	Score should drop significantly for deletions
Insensitivity to slot mention order	Score should not change significantly when the order of slots in the pseudo-reference changes
Robustness to Boolean slot mention polarity	Score should drop when the utterance mentions the opposite value of a Boolean slot than indicated in the pseudo-reference, regardless of the given polarity being positive or negative
Robustness to synonyms	Score should not change significantly if a semantically equivalent phrase is used in the utterance to express information in the pseudo-reference
Ability to clearly recognize complete unrelatedness	Score should drop far more (ideally close to zero) when evaluating an utterance using a random pseudo-reference

Table 8.4: An overview of the ideal semantic accuracy metric behavior and how it translates to a metric's performance when comparing an utterance with a perturbed pseudo-reference.

8.2.2 Evaluation

Metrics. We evaluate the effectiveness of the proposed method to determine the semantic accuracy of an utterance, introduced earlier in this section, by testing it with several reference-based metrics: BLEU, METEOR, ROUGE-N, ROUGE-L, BERTScore and BLEURT. Some of them were already introduced earlier in Table 4.1, as we use them for standard reference-based automatic evaluation of our model outputs, and the remaining ones are described in Table 8.5. For ROUGE-N we report the results for $N \in \{1, 2\}$, and we report the precision, the recall and the F1 version of BERTScore. We enable baseline rescaling when calculating BERTScore in order for the metric to produce values in a wider range and thus for us to see score changes more clearly.[4]

[4]Before rescaling, BERTScore tends to output values in a rather limited range, such as between 0.8 and 1.0. After rescaling, the values are typically between 0.0 and 1.0, although they may fall slightly outside this range in extreme cases. See Zhang et al. (2020a) for more details.

Metric	Definition
ROUGE-N (Lin and Hovy, 2003)	N-gram recall, i.e., number of N-grams co-occurring in a candidate and a reference divided by the total number of N-grams in the reference.
BERTScore (Zhang et al., 2020a)	Average of maximum pairwise cosine similarities between a candidate's and a reference's tokens' contextual embeddings. Offers precision, recall and F1-score versions.
BLEURT (Sellam et al., 2020)	A learned metric that utilizes contextual embeddings to score a candidate on how well it conveys the meaning of a reference. Pretrained on large amounts of synthetic sentence-pair data with automatic metric scores, and further fine-tuned on data with human ratings.

Table 8.5: Simplified definitions of additional reference-based automatic metrics that we employ for semantic accuracy evaluation. They operate on a pair of texts: a candidate (the one being evaluated) and a reference (typically human-authored, but in our case a pseudo-reference automatically created from an MR).

Experimental setup. Since most of the perturbations modify a randomly chosen slot in the MR before creating a pseudo-reference, we evaluate the metrics across 5 independent runs. Within each run, all metric scores are calculated on the same set of perturbed pseudo-references, so as to make the comparison as fair as possible. We perform all experiments on the validation partition of the ViGGO dataset, which contains 238 unique MRs, each with 3 reference utterances, for a total of 714 examples in each run.

Results format. Along with absolute metric scores, we report relative score changes with respect to non-perturbed pseudo-references. Relative scores are more useful for recognizing how a metric reacts to a perturbation in pseudo-references, yet absolute scores also offer insights into a metric's behavior, especially across different sets of experiments, such as using different pseudo-reference formats. For all perturbations except for slot shuffle we want the metrics to maximize the *negative* difference. As mentioned earlier, the slot order in the pseudo-reference does not

matter, therefore the relative change for the slot shuffle perturbation would ideally remain as close to zero as possible.

Throughout the rest of this section, we present and discuss the results of experiments using 4 different pseudo-reference formats – the combinations of using vs. not using slot names, and comma vs. space as the value (or slot-value pair) separator. We also look into the effects of data lowercasing on the metric scores. Finally, we try different model sizes for the two neural metrics, BERTScore and BLEURT, to see if it has a significant impact on the metrics' performance.

8.2.2.1 Performance of Reference-Based Metrics Using Pseudo-References

We start the evaluation by looking at how different metrics are affected by different perturbations in pseudo-references. Figure 8.1 shows a comprehensive overview of the results across all the above-mentioned metrics using the simplest pseudo-reference format (variant #1 from Table 8.2).

The first thing we observe is that most scores are in the metrics' lower range (the full range being 0 to 1), which makes perfect sense considering we are not comparing utterances with other utterances but with pseudo-references, which lack the fluency of natural language and only contain content words, making them substantially different from the utterances. One exception is ROUGE-1, which is a unigram recall metric and thus may benefit from the reference only having salient words which are most likely to be present in the utterance as well.

Fact omissions, substitutions, and hallucinations. Looking at the relative score changes in the bottom half of the table in Figure 8.1, we can instantly see that different types of metrics follow different trends for different semantic errors. Sub-

Perturbation	BLEU	METEOR	ROUGE-1 (recall)	ROUGE-2 (recall)	ROUGE-L (F1)	BERTScore			BLEURT
						precision	recall	F1	
None	0.1341	0.4701	0.8804	0.4295	0.3886	0.2016	0.5161	0.3344	0.4663
Average of 5 runs									
Insertion	0.1341	0.3967	0.6971	0.3402	0.3639	0.2013	0.3810	0.2819	0.4251
Substitution	0.1068	0.3422	0.6360	0.3020	0.3103	0.1446	0.3649	0.2415	0.3860
Deletion	0.1007	0.4116	0.8767	0.4248	0.3276	0.0915	0.4712	0.2448	0.3975
Bool negation	0.1339	0.4650	0.8703	0.4245	0.3859	0.1955	0.5030	0.3255	0.4608
Slot shuffle (≈)	0.1336	0.4710	0.8804	0.4318	0.3415	0.1734	0.4818	0.3038	0.4504
Ref. shuffle (within DA)	0.0122	0.0554	0.1365	0.0354	0.0684	-0.0207	0.0703	0.0220	0.1899
Ref. shuffle	0.0048	0.0333	0.0936	0.0153	0.0465	-0.0567	0.0150	-0.0224	0.1729
Relative change									
Insertion	0.01%	-15.62%	-20.82%	-20.79%	-6.36%	-0.17%	-26.17%	-15.69%	-8.84%
Substitution	-20.37%	-27.20%	-27.76%	-29.68%	-20.15%	-28.28%	-29.30%	-27.78%	-17.23%
Deletion	-24.88%	-12.44%	-0.42%	-1.10%	-15.69%	-54.61%	-8.70%	-26.79%	-14.75%
Bool negation	-0.15%	-1.08%	-1.15%	-1.17%	-0.71%	-3.05%	-2.54%	-2.67%	-1.19%
Slot shuffle (≈)	-0.40%	0.18%	0.00%	0.54%	-12.12%	-13.98%	-6.64%	-9.14%	-3.41%
Ref. shuffle (within DA)	-90.92%	-88.21%	-84.49%	-91.76%	-82.41%	-110.28%	-86.38%	-93.42%	-59.28%
Ref. shuffle	-96.41%	-92.91%	-89.37%	-96.45%	-88.03%	-128.11%	-97.09%	-106.69%	-62.92%

Figure 8.1: Scores calculated by various metrics comparing human-written utterances from the ViGGO validation set to **perturbed pseudo-references** created from the corresponding MRs (using **space** as the separator and **no slot names**). The upper half of the table shows the absolute values of metric scores averaged over 5 independent runs, while the bottom half shows the relative score change with respect to the no-perturbation baseline (yellow row labeled "None" at the top). Greater negative changes (good, except for slot shuffle) are highlighted in an increasingly darker shade of blue, and non-negative changes (bad, except for slot shuffle) are highlighted in red.

stitutions are generally recognized well by all the metrics, with a 17–30% negative relative change on average. However, it is the recall-based metrics, i.e, ROUGE-N and BERTScore-recall, that most clearly identify fact omissions ($\sim$ insertion perturbation), while hallucinations ($\sim$ deletion perturbation) are something precision-based metrics excel at detecting, i.e., BLEU and BERTScore-precision. METEOR, ROUGE-L and BERTScore-F1, being F1-based metrics, tend to perform relatively well on all of these three semantic errors. That being said, the ability of ROUGE-L to detect fact omissions is poor compared to that of METEOR and BERTScore, while BERTScore recognizes hallucinations much more clearly than the other two. Although not corresponding to any of the three metric categories, BLEURT behaves most similarly to F1-based metrics, i.e., able to recognize all of the above three semantic errors in general, yet not as clearly as BERTScore.

Boolean negations. Boolean slots are different from the other slot types in that they always require their salient part (usually corresponding to a part of the slot name) to be mentioned in the utterance, which typically has to be accompanied by a negation when the value is "false" or "no".[5] This makes it arguably more difficult for automatic metrics to identify the cases when these slots are mentioned incorrectly, i.e., with the opposite polarity. Out of the 714 examples in the ViGGO validation set, only 309 contain one or more Boolean slots, so the relative score changes for the Boolean negation perturbation are in general lower, since they are averaged over the whole set. BLEU and ROUGE-L only demonstrate a negative relative change of less than 1%, while BERTScore, with a 2.67% negative change, significantly outperforms all the other metrics (see Figure 8.1). In fact, we notice

[5]It is possible in some cases, however, to use a different word/phrase instead of a negation, such as "single-player" for the HAS_MULTIPLAYER slot with a value "no".

that both the precision and the recall component of BERTScore achieve a similar negative difference in the 2.5–3% range, which suggests that BERTScore can detect Boolean negations regardless of being true-to-false or false-to-true: a missing negation in the utterance is presumably detected by the recall component, and an extra negation by the precision component. The ratio of Boolean slots with a true value and a false value in the validation set is nearly 1 : 1, with 219 and 213 slots, respectively.

Slot shuffle. Since metric scores should not be affected by the slot order in a pseudo-reference, there should ideally be no relative score change for the slot shuffle perturbation at all. Nevertheless, as Figure 8.1 shows, this turns out to be the case for the ROUGE-1 metric only. ROUGE-2, BLEU and METEOR all show a change of less than 0.6% in either direction, which is certainly acceptable. ROUGE-L, however, scores pseudo-references with shuffled slots significantly lower (−12.12%), which is a consequence of this metric looking for the longest common subsequence between the two texts – an operation where the slot mention order matters.[6] Neural metrics are also greatly affected by the slot order. While BLEURT scores decrease on average by 3.41%, BERTScore drops by more than 9% after shuffling slots. This drastic fall, while undesirable from the perspective of a semantic accuracy metric, can probably be explained away by the change in the tokens' contextual embeddings that happens when the slots get shuffled. Considering most of the words in a pseudo-reference are salient words, their context changes dramatically with any word order modification. However, as we show in our later experiments, including slot names in pseudo-references – for BERTScore in particular – drastically reduces the relative

[6] Apparently, slots are often mentioned in the reference utterances in a similar order to that in the corresponding MRs, since the ROUGE-L score drops after shuffling the slots.

change for this perturbation (down to 3 times lower levels), and that without a negative impact on the relative changes for the other perturbations.

Reference shuffle. Reference shuffling is a perturbation performed on pseudo-reference level, as opposed to slot level. The experiments with this perturbation serve as a sanity check that the metrics recognize a pseudo-reference being completely unrelated to an utterance. The "within DA" variant shuffles pseudo-references within the same DA type only, which helps partially preserve the pseudo-reference length distributions. As we can see in the bottom two rows in Figure 8.1, all metrics pass the test, reporting negative relative score differences between 62% and 97%, and even over 100% in case of BERTScore, since it is possible for the scores of this metric to become slightly negative. Perhaps even more relevant is the fact that all the metrics, except for BLEURT, score random pseudo-utterances close to zero. BLEURT, being a metric trained on text pair ratings, presumably does not drop to zero unless one of the texts is an empty string, and thus even two completely unrelated texts still get a score well above zero. BLEURT's practical lower bound being significantly higher than zero appears to also be the primary reason for its relative score changes for all perturbations being less prominent than those of BERTScore.

8.2.2.2 Qualitative Evaluation of Boolean Negations and Synonyms

Here we perform a manual analysis of how the metrics handle two phenomena that cannot be evaluated easily in an automated fashion.

Boolean slot negations. We saw earlier that, based on relative score changes averaged over the whole validation set, BERTScore appeared to be the best metric at recognizing Boolean slot negations. Nevertheless, the score changes across all met-

	Utterance	Pseudo-reference
1a	Driver is an average driving simulation game. You play in third person, and there is no multiplayer mode.	Driver average driving/racing, simulation third person not multiplayer
1b	Driver is an average driving simulation game. You play in third person, and there is **no multiplayer mode**.	Driver average driving/racing, simulation third person multiplayer

	BLEU	METEOR	ROUGE-1 (recall)	ROUGE-2 (recall)	ROUGE-L (F1)	BERTScore			BLEURT
						precision	recall	F1	
1a	0.0611	0.2778	0.7778	0.2500	0.5185	0.2363	0.4137	0.3205	0.4536
1b	0.0611	0.3086	0.8750	0.2857	0.5385	0.1114	0.3374	0.2150	0.4431

	Utterance	Pseudo-reference
2a	Driver was an average driving simulation game with a third person perspective. It didn't have multiplayer.	Driver average driving/racing, simulation third person not multiplayer
2b	Driver was an average driving simulation game with a third person perspective. It **didn't have multiplayer**.	Driver average driving/racing, simulation third person multiplayer

	BLEU	METEOR	ROUGE-1 (recall)	ROUGE-2 (recall)	ROUGE-L (F1)	BERTScore			BLEURT
						precision	recall	F1	
2a	0.0699	0.4227	0.7778	0.2500	0.5385	0.3097	0.4496	0.3778	0.4389
2b	0.0699	0.4709	0.8750	0.2857	0.5600	0.1385	0.3813	0.2489	0.4189

	Utterance	Pseudo-reference
3a	Driver is a third person driving simulation game with average ratings. It is single-player only.	Driver average driving/racing, simulation third person not multiplayer
3b	Driver is a third person driving simulation game with average ratings. It is **single-player only**.	Driver average driving/racing, simulation third person multiplayer

	BLEU	METEOR	ROUGE-1 (recall)	ROUGE-2 (recall)	ROUGE-L (F1)	BERTScore			BLEURT
						precision	recall	F1	
3a	0.0602	0.4276	0.6667	0.1250	0.2400	0.1715	0.3549	0.2580	0.4390
3b	0.0602	0.4769	0.7500	0.1429	0.2500	0.1085	0.3282	0.2095	0.4531

Figure 8.2: Metric score changes for 3 different utterances when a Boolean slot is negated in the pseudo-utterance ("not multiplayer" → "multiplayer"). Each of the three examples shows both versions of the pseudo-reference, with the incorrect slot mention highlighted in blue in the utterances with perturbed pseudo-references. Metric scores for the pairs with a perturbation are highlighted in green when they reduce substantially, yellow when they reduce slightly, and red when they stay the same or increase.

rics, including BERTScore, were too small to make a sound conclusion. Therefore, we inspect how the metrics react for a sample of utterances when a Boolean slot in their corresponding pseudo-reference is negated. Figure 8.2 shows an example of three different utterances for the same MR. Each of these utterances demonstrates a different realization of the Boolean slot HAS_MULTIPLAYER[NO], verbalized in the pseudo-utterances as "not multiplayer", but none of the utterances contains this phrase verbatim. For overlap-based metrics this means the slot mention is only partially matched (because the word "not" has no exact match in the utterances), so their score remains unchanged or actually incorrectly increases on the perturbed pseudo-reference which omits the "not". Both of the neural metrics, however, handle the situation appropriately, at least in case of 1b and 2b. The third utterance mentions the Boolean slot as "single-player only", which apparently confuses BLEURT but not BERTScore. In all three cases, BERTScore drops significantly, clearly indicating that the utterances {1,2,3}b are inferior in semantic accuracy with respect to their pseudo-references. That being said, there are scenarios that even BERTScore fails to handle correctly, such as those involving aggregations of multiple Boolean slots in an utterance.

Slot mention paraphrases. The behavior of the metrics on different but semantically equivalent slot mentions is difficult to evaluate automatically. We therefore again turn to studying a sample of utterances for which we systematically manipulate the pseudo-reference in different ways that will help us see if the metrics correctly understand paraphrased mentions. We re-evaluate the utterances using these perturbed pseudo-references, and analyze how the metric scores change. An example of two different utterances for the same MR, along with their respective pseudo-

	Utterance	Pseudo-reference
1a	RollerCoaster Tycoon is a good **single-player** simulation strategy game in which you play from a **top down view**.	RollerCoaster Tycoon good simulation, strategy bird view not multiplayer
1b	RollerCoaster Tycoon is a good **single-player** simulation strategy game in which you play from a top down view.	RollerCoaster Tycoon good simulation, strategy not multiplayer
1c	RollerCoaster Tycoon is a good **single-player** simulation strategy game in which you play from a top down view.	RollerCoaster Tycoon good simulation, strategy side view not multiplayer

	BLEU	METEOR	ROUGE-1 (recall)	ROUGE-2 (recall)	ROUGE-L (F1)	BERTScore			BLEURT
						precision	recall	F1	
1a	0.0533	0.3188	0.6667	0.2500	0.4286	0.3005	0.5819	0.4276	0.4335
1b	0.0509	0.3897	0.7143	0.3333	0.3846	0.2127	0.6131	0.3827	0.4851
1c	0.0533	0.3188	0.6667	0.2500	0.4286	0.3166	0.5985	0.4440	0.4869

	Utterance	Pseudo-reference
2a	RollerCoaster Tycoon is a good simulation strategy game. You play with a **bird's eye perspective**. It's too bad there's **no multiplayer**.	RollerCoaster Tycoon good simulation, strategy bird view not multiplayer
2b	RollerCoaster Tycoon is a good simulation strategy game. You play with a bird's eye perspective. It's too bad there's **no multiplayer**.	RollerCoaster Tycoon good simulation, strategy not multiplayer
2c	RollerCoaster Tycoon is a good simulation strategy game. You play with a bird's eye perspective. It's too bad there's **no multiplayer**.	RollerCoaster Tycoon good simulation, strategy side view not multiplayer

	BLEU	METEOR	ROUGE-1 (recall)	ROUGE-2 (recall)	ROUGE-L (F1)	BERTScore			BLEURT
						precision	recall	F1	
2a	0.0492	0.3094	0.7778	0.3750	0.4242	0.2796	0.6727	0.4484	0.4821
2b	0.0492	0.3757	0.8571	0.5000	0.3871	0.1561	0.6866	0.3673	0.4202
2c	0.0492	0.3094	0.6667	0.3750	0.3636	0.2530	0.6480	0.4220	0.4338

Figure 8.3: Metric score changes for 2 utterances with different slot mention paraphrases, when the slot is deleted ("~~bird view~~") or substituted in the pseudo-utterance ("bird view" → "side view"). Both examples thus show 3 versions of the pseudo-reference, with the extra slot mention highlighted in red, and the incorrect slot mention highlighted in blue in the utterances with perturbed pseudo-references. Any non-verbatim slot mentions are in boldface. Metric scores for the pairs with a perturbation are highlighted in green when they reduce substantially ($\geq 10\%$), yellow when they reduce slightly ($< 10\%$), and red when they stay the same or increase.

references, can be seen in Figure 8.3. The first perturbation we test the metrics with is deletion. A metric's score remaining unchanged, or increasing, after deleting a slot in the pseudo-reference, implies that the metric is not aware of the slot being correctly mentioned in the utterance. Looking at rows 1b and 2b, we see that that's the case with most of the overlap metrics, except for ROUGE-L, in this example, and even BLEURT fails in row 1b. Second, by substituting the slot value with a different one, we check if the metrics differentiate between a correct and an incorrect slot mention. If the score does not drop, it suggests that the metric regards even the correct slot value paraphrase as incorrect. As we can see in case of 1c, all metrics seem to find the phrase "top down view" equally relevant to "bird view" as to "side view", or, surprisingly, even more to "side view" in case of both neural metrics. The scores in 2c show a different story, as the slot mention "bird's eye perspective" has the word "bird" in common with the pseudo-reference in 2a, but not in 2c. This is enough for most of the metrics to catch the semantic difference, nevertheless BLEU, METEOR and ROUGE-2 still fail. We conclude that neural metrics, especially BERTScore, are more robust in recognizing various equivalent slot mentions. While the example in Figure 8.3 gives the impression that even ROUGE-L performs similarly well, that appears to be due to the particular order of slot mentions in these two utterances, because ROUGE-L failed in virtually all the other examples that we examined where BERTScore and/or BLEURT correctly recognized the paraphrases.

8.2.2.3 Pseudo-Reference vs. Raw MR Format

Although a pseudo-reference looks more similar to an utterance than an MR does, it is not much more than a list of content words after all, just like an MR is, but without special separator symbols and with Boolean slots formulated in a different

Perturbation	BLEU	METEOR	ROUGE-1 (recall)	ROUGE-2 (recall)	ROUGE-L (F1)	BERTScore			BLEURT
						precision	recall	F1	
None	0.1117	0.1069	0.5645	0.2673	0.3369	0.2570	0.1796	0.2154	0.3795
Average of 5 runs									
Insertion	0.0805	0.0943	0.4405	0.2086	0.3049	0.2602	0.0944	0.1704	0.3423
Substitution	0.0865	0.0787	0.4198	0.1930	0.2684	0.2071	0.1222	0.1624	0.3261
Deletion	0.1047	0.0918	0.5591	0.2670	0.2964	0.1648	0.1857	0.1720	0.3437
Bool negation	0.1116	0.1069	0.5631	0.2673	0.3369	0.2519	0.1744	0.2102	0.3791
Slot shuffle (≈)	0.1117	0.1069	0.5645	0.2670	0.2944	0.2473	0.1713	0.2065	0.3694
Ref. shuffle (within DA)	0.0096	0.0116	0.0942	0.0224	0.0612	0.0426	-0.0299	0.0054	0.1821
Ref. shuffle	0.0047	0.0072	0.0699	0.0110	0.0452	0.0093	-0.0585	-0.0254	0.1695
Relative change									
Insertion	-27.97%	-11.75%	-21.97%	-21.98%	-9.50%	1.23%	-47.46%	-20.88%	-9.80%
Substitution	-22.58%	-26.40%	-25.64%	-27.78%	-20.33%	-19.40%	-31.96%	-24.61%	-14.08%
Deletion	-6.27%	-14.13%	-0.95%	-0.13%	-12.02%	-35.88%	3.37%	-20.14%	-9.44%
Bool negation	-0.09%	0.00%	-0.24%	0.00%	0.01%	-1.99%	-2.92%	-2.40%	-0.09%
Slot shuffle (≈)	0.00%	0.00%	0.00%	-0.13%	-12.63%	-3.78%	-4.61%	-4.11%	-2.67%
Ref. shuffle (within DA)	-91.37%	-89.17%	-83.31%	-91.61%	-81.83%	-83.41%	-116.67%	-97.51%	-52.03%
Ref. shuffle	-95.79%	-93.26%	-87.62%	-95.89%	-86.60%	-96.39%	-132.57%	-111.79%	-55.34%

Figure 8.4: Scores calculated by various metrics comparing human-written utterances from the ViGGO validation set to their **corresponding MRs** with DA types removed. The upper half of the table shows the absolute values of metric scores averaged over 5 independent runs, while the bottom half shows the relative score change with respect to the no-perturbation baseline (yellow row labeled "None" at the top). Greater negative changes (good, except for slot shuffle) are highlighted in an increasingly darker shade of blue, and non-negative changes (bad, except for slot shuffle) are highlighted in red.

way. Understandably then, the reader may wonder if creating pseudo-references provides any significant benefits in the semantic evaluation of an utterance. We therefore also carried out an experiment in which we calculated all the reference-based metric scores comparing the reference utterances directly with their respective MRs. We try using MRs without the DA type indications as well, considering they are not likely to aid in the semantic evaluation using reference-based metrics.

An overview of the metric scores and their relative changes on perturbed

MRs is shown in Figure 8.4. We show the results using MRs without DA types, as we observed that omitting the DA from the MR has a few benefits over using the full MRs:

- Metric scores are overall higher.
- BLEU score actually decreases for hallucinations instead of increasing.
- A slight increase in most relative ROUGE score differences.
- BERTScore has an almost 2-times lower relative score difference for shuffled slots (4.11% vs. 7.85%), which would ideally be zero. At the same time, however, there is an almost 10% drop in relative difference for fact omissions, and a ca. 20% drop for hallucinations.
- BLEURT exhibits an up to 15% increase in relative score differences, but also an about 2.5-times larger relative difference for shuffled slots (2.67% vs. 1.07%).

We note that all the metrics are, however, able to detect a good amount of semantic errors already by simply comparing utterances with the corresponding MRs.

The pseudo-reference format most similar to an MR is the one that includes slot names, so we compare the results of the MR experiments with those using pseudo-references with slot names (Figure 8.5). The first thing to notice is that, for most of the metrics, the scores are significantly lower than when using pseudo-references with slot names. More importantly though, there are a number of crucial differences in the relative score changes across the board. Table 8.6 lists the differences broken down by metric. We can see that pseudo-references are a clear winner here, with only a few aspects in which they appear to underperform raw MRs. We attribute this to the more sentence-like nature of pseudo-references, in particular the verbalized slot names and reworded Boolean slots. Nevertheless,

Metric	Raw MRs	Pseudo-references with slot names
BLEU	Lower scores overall; significantly less effective at detecting hallucinations (6% vs. 25% relative difference); fails to detect negations of Boolean slots	Fails to detect fact omissions
METEOR	Drastically lower scores overall; ca. 25% smaller relative score difference for fact omissions; fails to detect negations of Boolean slots	Ca. 25% smaller relative score difference for hallucinations
ROUGE-N	Slightly lower scores overall; fails to detect most negations of Boolean slots; ROUGE-2 fails to detect hallucinations	Up to 15% smaller relative score differences for fact omissions
ROUGE-L	Slightly lower scores overall; ca. 12% smaller relative score difference for hallucinations; fails to detect negations of Boolean slots	Ca. 16% smaller relative score difference for fact omissions
BERTScore	Significantly lower recall scores (but bigger relative differences); 10–25% smaller relative score difference for substitutions and hallucinations (F1); ca. 25% bigger relative score difference when slots are shuffled	Lower precision scores (but bigger relative differences); ca. 10% smaller relative score difference for fact omissions
BLEURT	Lower scores overall; up to 25% smaller relative score differences for fact omissions, substitutions and hallucinations; fails to detect negations of Boolean slots	

Table 8.6: Major differences in metric behavior between using raw MRs (without the DA type) and pseudo-references (with slot names).

special symbols present in MRs appear to also drag the metrics' performance down, especially those that are recall-based or recall-heavy, such as $\text{BERTScore}_{\text{recall}}$ and METEOR.[7]

8.2.2.4 Effects of Including Slot Names

Now that we have established the benefits of using a pseudo-reference over an MR, let us have a look at different pseudo-reference formats, starting with including slot names vs. using slot values only. Comparing Figures 8.1 and 8.5 we see that the result of using slot names is overall lower scores, especially those of recall-based metrics. Nevertheless, this is expected because this format adds a lot of extra words to pseudo-references which, in case of the ViGGO dataset, are rarely mentioned in the corresponding utterances.

Looking at the relative score changes, most metrics exhibit a slightly more prominent difference for insertion perturbations when slot names are included, but a slightly less prominent one for deletions. Substitutions and Boolean negations also see a minor decrease when slot names are included. However, including slot names has one notable benefit for the BERTScore metric: for the slot shuffle perturbation, it brings the score difference much closer to zero, specifically from around 9% down to 2–3%, depending on the separator used.

Overall, speaking in terms of utterance errors, including slot names in pseudo-references may be considered mildly beneficial in emphasizing the metric score change caused by fact omissions, which normally only leads to a low to moderate drop in scores (as opposed to the moderate to high drop for substitutions and hallucinations). Although metric scores for utterances with substitutions and hallu-

[7]The ROUGE metric script removes non-alphanumeric characters before calculating the score, hence the relatively small difference in its scores despite being recall-based.

Perturbation	BLEU	METEOR	ROUGE-1 (recall)	ROUGE-2 (recall)	ROUGE-L (F1)	BERTScore			BLEURT
						precision	recall	F1	
None	0.1335	0.3659	0.5989	0.2708	0.3404	0.2107	0.3255	0.2629	0.4458
Average of 5 runs									
Insertion	0.1339	0.3073	0.4774	0.2197	0.3132	0.2112	0.2198	0.2144	0.3984
Substitution	0.1030	0.2708	0.4374	0.1919	0.2697	0.1577	0.2284	0.1908	0.3702
Deletion	0.1005	0.3274	0.5933	0.2637	0.2940	0.1083	0.3027	0.1936	0.3894
Bool negation	0.1334	0.3625	0.5930	0.2681	0.3381	0.2056	0.3173	0.2565	0.4414
Slot shuffle (≈)	0.1333	0.3661	0.5989	0.2706	0.2996	0.2012	0.3182	0.2544	0.4347
Ref. shuffle (within DA)	0.0112	0.0537	0.0978	0.0199	0.0601	-0.0142	-0.0184	-0.0157	0.1884
Ref. shuffle	0.0050	0.0382	0.0699	0.0099	0.0439	-0.0520	-0.0640	-0.0572	0.1736
Relative change									
Insertion	0.31%	-16.01%	-20.29%	-18.87%	-7.99%	0.23%	-32.47%	-18.46%	-10.64%
Substitution	-22.83%	-25.98%	-26.96%	-29.15%	-20.78%	-25.14%	-29.84%	-27.42%	-16.97%
Deletion	-24.69%	-10.53%	-0.94%	-2.63%	-13.64%	-48.62%	-7.00%	-26.34%	-12.65%
Bool negation	-0.07%	-0.93%	-0.99%	-0.98%	-0.67%	-2.41%	-2.53%	-2.45%	-0.99%
Slot shuffle (≈)	-0.13%	0.05%	0.00%	-0.07%	-11.98%	-4.52%	-2.25%	-3.24%	-2.49%
Ref. shuffle (within DA)	-91.63%	-85.32%	-83.66%	-92.64%	-82.35%	-106.72%	-105.65%	-105.96%	-57.74%
Ref. shuffle	-96.24%	-89.55%	-88.33%	-96.35%	-87.10%	-124.70%	-119.67%	-121.77%	-61.06%

Figure 8.5: Scores calculated by various metrics comparing human-written utterances from the ViGGO validation set to **perturbed pseudo-references** created from the corresponding MRs (using **space** as the separator and **including slot names**). The upper half of the table shows metric scores averaged over 5 independent runs, while the bottom half shows the relative score change with respect to the no-perturbation baseline (yellow row labeled "None" at the top). Greater negative changes (good, except for slot shuffle) are highlighted in an increasingly darker shade of blue, and non-negative changes (bad, except for slot shuffle) are highlighted in red.

cinations show a lower difference when slot names are included, this can be deemed less consequential, since these relative changes are in general substantially greater than for the other errors.

Among all the metrics, it is BERTScore that benefits the most from slot names in the pseudo-utterance. Its F1 version is minimally affected on substitutions and hallucinations, but the 18% boost for fact omissions is significant. Finally, the more than 3-fold difference reduction for shuffled slots is an important improvement, making BERTScore behave in a more desired way, considering the order of slot mentions in the utterance is completely independent of the slot order in the pseudo-reference. We believe these gains in performance come from the use of contextual embeddings in the BERTScore metric, which allows it to make use of the additional information in the form of slot names despite them not being mentioned in the utterance, which overlap-based metrics cannot take advantage of. Moreover, including slot names presumably creates a more stable context around each slot value, resulting in smaller swings in the contextual embeddings when the slots are shuffled.

8.2.2.5 Effects of Different Slot Separators

So far, we have shown results of experiments with pseudo-references using a space between slots. The only other separator that would arguably be suitable in a pseudo-reference, which is supposed to resemble a sentence, is comma (followed by a space).

Our experiments with comma-separated slots, whether including slot names or not, show no benefits over simply using a space. We observe no meaningful dif-

ference in scores for BLEU, ROUGE[8] and BLEURT. METEOR scores are, however, significantly lower overall with the comma separator, and the relative score difference for hallucinations drops by as much as a half. BERTScore is overall slightly higher but relative score differences are 5–15% smaller with the comma separator, and on Boolean negations up to 30% smaller. This last effect is difficult to explain, but our guess is the use of commas in pseudo-references may be excessive and unnatural from the perspective of BERTScore's language model, and may thus be negatively affecting the matches between contextual token embeddings in the pseudo-reference and the utterance.

All in all, using comma as the slot separator in pseudo-references does not seem to be beneficial to the semantic error detection. In fact, it makes some relative score changes less prominent in case of METEOR and BERTScore.

8.2.2.6 Effects of Lowercasing

Converting the utterance and the pseudo-reference to lowercase before calculating metric scores only affects neural metrics. Overlap-based metrics usually automatically lowercase the texts, as that is the only way for them to recognize that the same word capitalized at the beginning of a sentence in one text and in lowercase in the middle of a sentence in the other text are actually a match. A downside of lowercasing is, however, that the metrics cannot differentiate between common words that are a part of a title and those that are not. Neural metrics using contextual token embeddings have a great advantage in this situation.

For both BERTScore and BLEURT the scores become overall lower when data is lowercased. BERTScore's relative F1 score difference for fact omissions is

[8]ROUGE metrics are presumably not affected because of the punctuation being removed by the script.

smaller when lowercased, which is the scenario with the lowest relative difference among the three primary perturbations. The difference is, however, higher for substitutions and hallucinations. With slot names included, the trends are similar, but shuffling slots causes the relative change to double in magnitude (yet it still remains lower than without using slot names). No substantial difference can be seen for BLEURT. Relative score changes tend to merely become slightly smaller for the fact omissions and substitutions, and slightly bigger for hallucinations, when lowercased.

In summary, lowercasing utterances and pseudo-references only has a notable effect on BERTScore, which seems to be slightly thrown off by it, presumably because the text case makes a difference in the contextual embeddings (e.g., "Jack" vs. "jack", or "meteor" vs. "METEOR"). We therefore conclude that it is better for neural metrics to keep the texts' original case.

8.2.2.7 Effects of Importance Weighting in BERTScore Calculation

The BERTScore metric can optionally have importance weighting of tokens enabled, with the goal of preferring matches of salient words over matches of stop-words and other common words. It is calculated using inverse document frequency (IDF) on all references, in our case all pseudo-references of the validation set. The IDF score of a token w is calculated as follows:

$$\mathrm{IDF}(w) = -\log \frac{1}{N} \sum_{i=1}^{N} \mathbb{1}[w \in r_i] \ ,$$

where $\{r_i\}_{i=1}^{N}$ is the set of all N pseudo-references and $\mathbb{1}[\cdot]$ is the indicator function. For details on how the IDF weights are incorporated into the BERTScore calculation,

we refer the reader to Zhang et al. (2020a).

Considering our pseudo-references do not contain stop-words and most of their content is made of salient words, IDF weighting might not quite work as desired in our scenario. We evaluate its effects nonetheless, using all 4 pseudo-reference formats described earlier. The following paragraph summarizes our findings.

In general, with IDF weighting enabled, the precision component of BERTScore significantly drops while recall goes up, but the effect on the F1 scores is relatively small. However, the relative F1 score change increases for fact omissions, substitutions, as well as hallucinations by up to 20%, whereas it decreases for Boolean negations by up to 20%. The gains from IDF weighting are less significant when slot names are included in pseudo-references though. In fact, in the case of hallucinations, the relative change is smaller with IDF weighting, both when using comma and space as the separator. We speculate that slot names actually play a somewhat important role in determining the semantic accuracy using BERTScore because it is them that would be expected to be downweighted the most via IDF, but doing so actually has a negative impact on the relative score differences caused by perturbations.

In conclusion, IDF weighting, surprisingly, has a desirable effect on BERTScore in our scenario under most circumstances. Unfortunately, it cannot be taken advantage of during single-input inference (as opposed to running inference on a whole test set) because of the lack of references to calculate IDF weights on.

8.2.2.8 Effects of Model Size

The two neural metrics in our experiments, BERTScore and BLEURT, are both powered by a pretrained language model. BERTScore works with a variety of

transformer-based models, ranging from BERT (Devlin et al., 2019) and its many variants to XLNet (Yang et al., 2019) to encoder-decoder models like T5 (Raffel et al., 2020). At the time of writing, the best-performing model – recommended by the authors for results that correlate best with human ratings[9] – is a DeBERTa model (He et al., 2020) fine-tuned on the MNLI corpus (Williams et al., 2018) for the natural language inference task. BLEURT, on the other hand, uses a RemBERT model (Chung et al., 2020) fine-tuned on both synthetic and human-annotated WMT data (see Sellam et al. (2020) for more details). At the time of writing, the BLEURT-20 model checkpoint is recommended by the authors for the best results.[10] In our final set of experiments, we evaluate the performance of BERTScore and BLEURT on our semantic accuracy task, using the recommended models of varying sizes. The results are summarized in Figure 8.6.

BERTScore. We compare the recommended DeBERTa model (deberta-xlarge-mnli, ca. 750M parameters) with its smaller version (deberta-large-mnli, ca. 400M parameters). The bigger model gives overall lower absolute scores, but the relative score changes are greater for all perturbations (up to 20% for deletions, and up to 60% for Boolean negations). The slot shuffle perturbation, however, also results in a greater relative score change (by up to 25%), which is not desired. Considering the gains in relative differences are in a similar range for all the perturbations (except for Boolean negations), we would not expect the bigger model to necessarily perform better than the smaller model in differentiating between utterances that are semantically accurate and those that are not. An exception might be recognizing wrong polarity of Boolean slot mentions, where the bigger model might be more

[9] https://github.com/Tiiiger/bert_score
[10] https://github.com/google-research/bleurt

Perturbation	BERTScore (deberta-xlarge-mnli)			BERTScore (deberta-large-mnli)			BLEURT-20		
	Precision	Recall	F1	Precision	Recall	F1	D12	D6	D3
None	0.1643	0.4480	0.2898	0.2016	0.5161	0.3344	0.4663	0.4657	0.4683
Average of 5 runs									
Insertion	0.1707	0.3303	0.2451	0.2017	0.3801	0.2819	0.4249	0.4195	0.4259
Substitution	0.1130	0.3168	0.2056	0.1425	0.3663	0.2408	0.3830	0.3777	0.3930
Deletion	0.0559	0.3974	0.2020	0.0940	0.4752	0.2477	0.3985	0.3981	0.3992
Bool negation	0.1592	0.4352	0.2815	0.1957	0.5029	0.3255	0.4607	0.4583	0.4666
Slot shuffle (≈)	0.1350	0.4147	0.2587	0.1733	0.4821	0.3039	0.4510	0.4512	0.4548
Ref. shuffle	-0.0698	0.0022	-0.0353	-0.0569	0.0143	-0.0227	0.1723	0.1646	0.1766
Relative change									
Insertion	3.90%	-26.28%	-15.42%	0.04%	-26.36%	-15.69%	-8.88%	-9.91%	-9.06%
Substitution	-31.20%	-29.29%	-29.05%	-29.33%	-29.03%	-27.99%	-17.86%	-18.89%	-16.08%
Deletion	-66.00%	-11.30%	-30.30%	-53.37%	-7.92%	-25.93%	-14.54%	-14.51%	-14.75%
Bool negation	-3.09%	-2.87%	-2.88%	-2.95%	-2.56%	-2.66%	-1.20%	-1.59%	-0.36%
Slot shuffle (≈)	-17.82%	-7.42%	-10.73%	-14.03%	-6.59%	-9.13%	-3.29%	-3.12%	-2.88%
Ref. shuffle	-142.51%	-99.50%	-112.18%	-128.24%	-97.23%	-106.80%	-63.05%	-64.66%	-62.28%

Figure 8.6: Scores calculated by **neural** metrics comparing human-written utterances from the ViGGO validation set to **perturbed pseudo-references** created from the corresponding MRs (using **space** as the separator and **no slot names**). The upper half of the table shows metric scores averaged over 5 independent runs, while the bottom half shows the relative score change with respect to the no-perturbation baseline (yellow row labeled "None" at the top). Greater negative changes (good, except for slot shuffle) are highlighted in an increasingly darker shade of blue, and non-negative changes (bad, except for slot shuffle) are highlighted in red.

accurate, given the 60% relative change increase. However, we note that we only observed such a high increase when using the comma separator, while with the space separator it was just below 10%.

BLEURT. We compare the three distilled models of BLEURT-20 provided by the authors: D12 (ca. 167M parameters), D6 (ca. 45M parameters), and D3 (ca. 30M parameters). They run 3–20 times faster, while correlating 10–25% less with human ratings, than the full model. Among the three of them, the two bigger ones perform similarly, while the smallest one underperforms in two perturbations. In fact, D6 achieves slightly greater relative score differences than D12 across all perturbations except for slot shuffle, where a smaller difference is actually desired. We observe the same trends regardless of the separator used in the pseudo-references. D3's relative score change for substitutions is lower, but it is the Boolean negations where it lags the most behind the two bigger models. All in all, the D6 model appears to be the best choice among the three distilled model variants for our purposes of semantic accuracy scoring.

8.3 Slot Aligner-Based Semantic Accuracy Metric

We look separately at the heuristic slot aligner as a method for semantic accuracy evaluation, as proposed earlier in Section 8.1.2, where we described four different approaches to measuring semantic accuracy. While using a reference-based metric together with pseudo-references (as described and evaluated in detail in Section 8.2) is a domain-agnostic method that can be used out of the box, the slot aligner approach leans on heuristic rules and a semantic dictionary. Although this affects its scalability, having been developed for semantic accuracy evaluation and

Perturbation	Average of 5 runs	Relative change
Insertion	21.04%	1116.30%
Substitution	24.80%	1333.76%
Deletion	1.47%	-15.14%
Bool negation	11.00%	535.95%
Slot shuffle (≈)	1.73%	0.00%
Ref. shuffle	92.00%	5217.69%

Figure 8.7: Slot error rate (SER) calculated by our heuristic slot aligner comparing human-written utterances from the ViGGO validation set to perturbed MRs. The middle column shows SER (as the proportion of erroneous slot mentions out of all slots) averaged over 5 independent runs, with 0% being the best and 100% the worst possible value. The right column shows the relative SER change with respect to the no-perturbation baseline (i.e., using the original MRs) whose SER is 1.73%. Negative changes are highlighted in blue, and non-negative changes in red.

reranking, the slot aligner is expected to work very effectively. Since it operates directly on MRs, we test it here on perturbed MRs in the same way we tested reference-based metrics on perturbed pseudo-references in the previous section.

8.3.1 Evaluation

A quick glance at the results in Figure 8.7 confirms that the slot aligner's ability to identify fact omissions, substitutions and Boolean negations is excellent, but as expected, it misses all hallucination errors (which correspond to the deletion perturbation).[11] The substitution perturbation changes one slot in each of the 714 examples in the ViGGO validation set, which corresponds to approximately 24.71% of all 2,889 slots across all MRs in the set. This matches almost perfectly with the SER for this perturbation (24.8%). The insertion perturbation adds 714 slots that are not mentioned in the utterances, which corresponds to approximately 19.82%

[11] Although the slot aligner is able to detect certain duplicate slot mentions, it is not designed to recognize hallucinations in general.

of the new total of 3,603 slots. This value slightly deviates from 21.04%, which we explain below. As far as Boolean negations are concerned, 309 examples contain at least one Boolean slot and have thus one negated. This amounts to approximately 10.7% of all slots, which is close enough to the 11% of erroneous slots reported for this perturbation by the slot aligner. Shuffling slots in the MRs does not affect the SER.

We note that according to the slot aligner there are 50 errors in the ViGGO validation set ($\sim$ 1.73% of all slots), the vast majority of which are in the RATING slot. This slot can be realized in a great number of different ways, also depending on the DA type of the MR, not all of which are successfully understood by the slot aligner. These are, however, phrases and formulations that are rather uncommon in the dataset and hence are rarely reproduced by fine-tuned NLG models. As we showed earlier in a human evaluation of the slot aligner, its accuracy is near perfect on model outputs. Nevertheless, analyzing the performance of the slot aligner on perturbed data of the validation set means that the 50 incorrect errors will be counted along with the errors introduced through perturbations, and thus inflating the error percentages reported by the slot aligner, as we saw in the previous paragraph. There are the two main reasons why the SER reported for substitutions (24.8% vs. 24.71%) is not significantly higher, as opposed to that of insertions (21.04% vs. 19.82%). In certain cases, a substitution is technically a deletion, such as when it replaces a list slot's value with a value that is a subset of the original list (e.g., "adventure, indie, platformer" $\rightarrow$ "adventure, platformer"). Moreover, when the substitution perturbation affects a slot that the aligner considered erroneous to begin with, it will not increase the error count from the slot aligner's perspective.

It is clear from the results and discussion above that the heuristic slot

aligner is not a perfect method for measuring semantic accuracy, but a very effective one nonetheless in detecting fact omissions, substitutions and Boolean negations.

8.4 Discussion

In this chapter, we described our proposed approach to referenceless evaluation of utterances, to be used in a stochastic inference method like Batch-MCTS, to discriminate between adequate and inadequate generated utterance candidates. The metric combines two components: one focusing on the fluency of the utterance, and the other on its semantic accuracy. We intend for our referenceless metric to be effective, but at the same time sufficiently easy to use so as to encourage its adoption in other domains and tasks, where applicable. Hence, for fluency, we opt for the perplexity calculated by a large pretrained language model, as it offers both scalability and high competence at determining general language fluency. For semantic accuracy, we proposed and evaluated a method that automatically creates pseudo-references from input MRs and utilizes existing reference-based metrics to assess an utterance's semantic quality. In addition to this method, we integrate the slot aligner-based semantic scoring too into our Batch-MCTS inference, which is highly accurate but not as scalable, representing one end of the spectrum of approaches we discussed in Section 8.1.2. The performance of the other two approaches would be expected to fall somewhere in between, just like their scalability does.

In Figure 8.8, we provide a summary of all the metrics we considered for the pseudo-reference-based evaluation of semantic accuracy. It gives a compact overview of how effective each of them was in our experiments at recognizing various semantic discrepancies in utterances, while not being thrown off by different formulations that

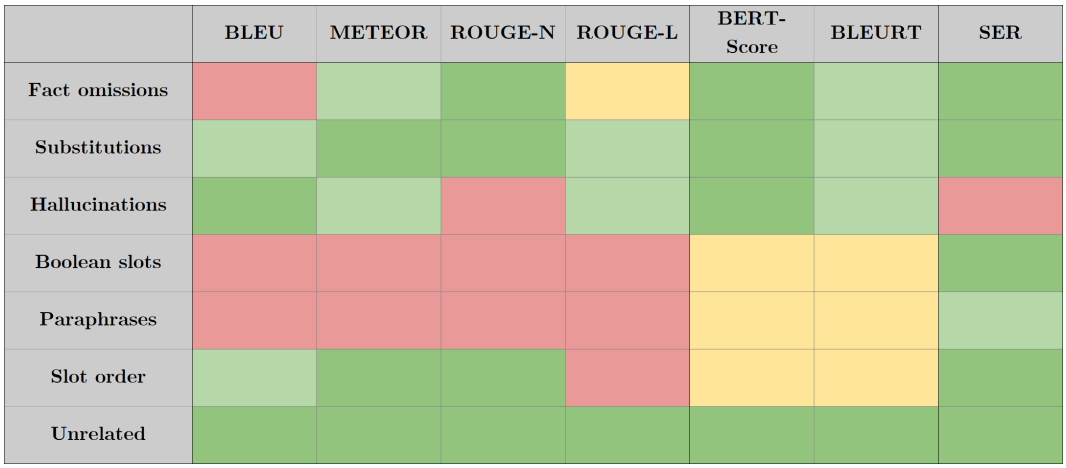


Figure 8.8: An overview of the overall performance of N-gram overlap and neural metrics in pseudo-reference-based semantic evaluation. Dark and light green shades highlight good and acceptable behavior, respectively, yellow shade denotes a not very clear or inconsistent distinction, while red means the metric fails altogether. For comparison, although not a pseudo-reference-based metric, we include SER, the slot error rate metric computed by our slot aligner.

are semantically equivalent to the input information. The two neural metrics among them, BERTScore and BLEURT, are clearly superior to N-gram overlap-based metrics, all of which fail to recognize most errors in Boolean slot mentions and tend to consider valid paraphrases as semantic errors. That being said, the neural metrics do not excel in these two departments either, as we showed in Section 8.2.2.1, but they frequently demonstrate a superior understanding of semantically equivalent expressions and can more reliably distinguish the polarity of a Boolean slot mention, as exemplified in Section 8.2.2.2. We attribute this ability to their contextual embeddings trained on large amounts of text, however, we speculate that it is these embeddings that are also responsible for making these metrics sensitive to how slots are ordered in the pseudo-reference. As a result, there is a chance that these metrics will score an utterance with an error higher than a similar semantically correct utterance that mentions the slots in a different order. Nevertheless, we found in Section 8.2.2.4 that in BERTScore the negative effect of this can be mitigated by

including slot names in the pseudo-references.

From among the overlap-based metrics, METEOR emerges as the most robust one in our setting. Unlike any of the other metrics in this category, it is reasonably reliable in identifying fact omissions, substitutions and hallucinations. This can partially be attributed to the fact that METEOR calculates an F1 score, rather than precision (such as BLEU) or recall (such as ROUGE-N). METEOR's other strength is the strategy of backing off to word stems and synonyms during word matching, but it turns out not to be robust enough in our experiments with paraphrases nonetheless. ROUGE-L, while also an F1-based metric, suffers from the strong dependency on word order which is largely irrelevant when comparing utterances against pseudo-references.

Generally speaking, the main weakness of the recall-oriented metrics is the inability to detect hallucinations, whereas precision-oriented metrics tend to favor shorter utterances, even if it is at the expense of matching all the facts in the reference.[12] Since F1-based metrics, including BERTScore, calculate a harmonic mean of precision and recall, these negative effects are not as prominent in them. Yet, given a fixed-length pseudo-reference and two semantically equivalent utterances, an F1-based metric is still more likely to penalize the more verbose one among them, since they will have a similar recall, but the precision will be lower in the verbose one due to a smaller proportion of its words having a match in the reference.

As an alternative approach to semantic accuracy evaluation, in Section 8.3 we evaluated the slot error rate (SER) calculated by our heuristic slot aligner on the same perturbations as the pseudo-reference-based metrics. As expected, it performed very well on fact omissions, substitutions, as well as Boolean slot negations,

[12]Hence the brevity penalty in BLEU.

confirming its suitability for use as a metric guiding the Batch-MCTS inference to semantically accurate candidates. Compared to the pseudo-reference-based metrics, the SER metric has just one major weakness and that is hallucination detection, which the slot aligner is not designed to handle. On the other hand, this metric is significantly less scalable and requires a certain amount of manual work, and possibly domain expertise, in order to extend it to a new domain.

Finally, there is one more important distinction between the SER metric and the pseudo-reference-based metrics. While the former calculates a score that directly corresponds to the semantic accuracy of an utterance, the latter produces a score that is only meaningful relative to that of a similar utterance. The pseudo-reference approach is highly dependent on the utterance and pseudo-reference being compared: different utterances of different lengths, and evaluated against different pseudo-references, are expected to yield substantially different scores, but that does not mean that the one with a lower score is less semantically accurate than the other. Therefore, even with the best reference-based metric, it is only effective when ranking a pool of similar utterances generated from the same MR, whereas comparing the scores of two unrelated utterances would be meaningless.

Next, we will describe and evaluate experiments with different variants of the automatic referenceless metric integrated into our Batch-MCTS inference proposed in Chapter 7, and see what effect they have on the tree search performance. Going forward, we will focus on the two neural metrics, BERTScore and BLEURT, which emerged as the most robust ones in the pseudo-reference experiments we conducted in this chapter. In addition to that, we will also test Batch-MCTS with the slot aligner's SER, which represents a different approach to calculating semantic accuracy, using the MR itself instead of pseudo-references.

Chapter 9

Batch-MCTS Inference Evaluation

In Chapter 7, we proposed and described a Monte-Carlo Tree Search-based inference method for NLG, as an alternative to the standard beam search and sampling methods that either ignore the output diversity or the semantic accuracy. With Batch-MCTS[1] we intend to reduce this gap and make generated utterances more diverse without a negative impact on their coherence and semantic accuracy. MCTS, however, needs a means of evaluating utterance candidates so that it knows if it is searching in the right direction or not. For this purpose we developed an automatic referenceless metric, described and evaluated in Chapter 8, that ranks utterances based on their quality. It consists of two components, one being a general-purpose language model (evaluating the fluency), and the other a standard neural NLG metric that compares utterances with a pseudo-reference created from the corresponding MR (estimating the semantic accuracy). An alternative metric we experiment with in the semantic accuracy component is the slot error rate calculated by the slot aligner from Chapter 3.

In this chapter, we put all of the above together and evaluate the perfor-

[1]Our Batch-MCTS implementation is available at: https://github.com/jjuraska/mcts-nlg

mance of Batch-MCTS equipped with the proposed referenceless metric, comparing it with multiple other inference methods. We start by giving an overview of our experiments and evaluation criteria, and we then present the results along with their analysis and discussion.

9.1 Experimental Setup

9.1.1 MCTS Parameters

Our proposed Batch-MCTS inference method is highly configurable, and so we investigate the effects of various parameters taking on different values. The two most important parameters are the batch size and the number of iterations. We experiment with batch sizes between 4 and 128, which corresponds to 4–128 parallel playouts – each of which generates an utterance candidate – per iteration. Unlike in typical MCTS applications, the benefits of running Batch-MCTS for many iterations are rather limited. There are two main reasons for this: (1) Batch-MCTS uses heavy playouts, i.e., highly informed simulations that lead to generally good candidates (utterances), as opposed to random ones; and (2) the batch modification means that each iteration explores a number of candidates equal to the batch size instead of just one. We therefore run our experiments with 1, 2 and 4 iterations.

Among the more internal MCTS parameters, there are different choices that can be made for the sampling and pruning strategy, the tree policy, as well as the reward aggregation. For sampling in playouts, i.e., utterance decoding, we settled on nucleus sampling with $p = 0.8$. As a form of pruning we only expand the search tree with nodes corresponding to the subset of tokens sampled from at each decoder step (see the description of the expansion phase in Section 7.2.2). We

keep the default tree policy, UCT, described in Section 7.2.1), with the exploration coefficient set to 1.0. As for the node reward aggregation, we experimented with averaging over utterance scores in a node's subtree and with keeping the maximum among the scores, but we observed no significant difference in the overall Batch-MCTS performance. In all the experiments whose results we report, we use the averaging mode.

9.1.2 MCTS State Evaluation Metric

The utterance evaluation metric to guide Batch-MCTS that we proposed in Chapter 8 has two components. For the fluency component we use the average token perplexity calculated by the GPT-2 pretrained language model (Radford et al., 2019). In comparison with two other language models we considered for this component, XLNet (Yang et al., 2019) and BART (Lewis et al., 2020), GPT-2 was predicting by far the best perplexity values. In general, being a decoder-only autoregressive language model, GPT-2 is best suited for the task of evaluating the fluency of a text.

The second component of the metric measures the semantic accuracy of an utterance by comparing it to a pseudo-reference composed from the corresponding MR. Our experiments in Chapter 8 showed the use of BERTScore and BLEURT for the comparison to be the most effective among other reference-based metrics we considered. To calculate BERTScore, we choose the deberta-large-mnli model because of its smaller size, and for BLEURT we use the D6 model variant. We disable IDF weighting in BERTScore, but we test the effects of using it with and without baseline rescaling. In addition to pseudo-reference-based evaluation, we also experiment with the slot aligner-based semantic evaluation in this component.

Since perplexity needs to be minimized, but BERTScore and BLEURT maximized, we invert the perplexity score when combining it with the semantic accuracy component into our referenceless utterance evaluation metric:

$$\text{score}(\hat{y}|x) = \frac{\text{SA}(\hat{y}|x)}{\text{PPL}(\hat{y})} \ ,$$

where $\text{SA}(\hat{y}|x)$ is the semantic accuracy score, calculated by BERTScore or BLEURT, for an utterance $\hat{y}$ generated from an input MR x. The objective of Batch-MCTS thus is to maximize this score in the tree search. When using SER for the semantic accuracy component, the metric calculation changes to the following:

$$\text{score}(\hat{y}|x) = \frac{1}{\text{PPL}(\hat{y}) \cdot \big(1 + \text{ERR}(\hat{y}|x)\big)} \ ,$$

where $\text{ERR}(\hat{y}|x)$ is the number of slot errors determined by the slot aligner for an utterance $\hat{y}$ generated from an MR x.

9.1.3 Data-to-Text NLG Model

Using an Nvidia RTX 2070 GPU with 8 GB of memory limits the size of models that we can effectively run the inference on, since we need to have three neural models loaded at the same time: one for the PPL calculation, another one for BERTScore or BLEURT, and of course the NLG model itself. All of our experiments with Batch-MCTS are thus performed with one concrete NLG model – a fine-tuned T5-small model which is sufficiently small to fit into the GPU memory along with the other two models and which we previously used for experiments with our SeA-GuiDe method in Chapter 6. For details about the model's parameters and the

	Test set references			Greedy search outputs		
	Vocab	**Vocab (delex.)**	**Unique (delex.)**	**Vocab**	**Vocab (delex.)**	**Unique (delex.)**
ViGGO	1,292	951	428	590	206	39
E2E	987	955	232	116	64	2

Table 9.1: Vocabulary size and unique word statistics for the reference utterances of the ViGGO and E2E test sets, along with the same statistics for the outputs of a fine-tuned T5-small model on the test sets using greedy decoding. Columns denoted "delex." show the statistics for a delexicalized test set and output set, i.e., with utterances having categorical slot mentions (e.g., game/restaurant names, release year, food type, etc.) replaced with a special placeholder (different for each slot type).

training regime we therefore refer the reader back to Section 6.2.2.

9.1.4 Datasets

We train the NLG model and evaluate its performance with different inference methods on two datasets, ViGGO and E2E, both introduced in Chapter 2. Despite ViGGO being a substantially smaller than E2E, models trained on this dataset typically use a larger vocabulary in their outputs, partly due to the 9 DA types used (as opposed to just 1 DA type in E2E). That being said, the vocabulary in these datasets is rather limited, considering each of them focuses on a single domain (video games and restaurants, respectively). Yet models tend to learn to use only a fraction of it confidently in their outputs, resulting in a smaller vocabulary still.

The vocabulary differences are illustrated in Figure 9.1. We note that the ViGGO test set has 1,083 reference utterances, and E2E has 4,693, however, both datasets contain multiple references per MR. There are 359 and 630 unique MRs in the ViGGO and E2E test set, respectively, and the set of model outputs only contains one utterance per unique MR. Therefore, the comparison of these statistics

is not entirely fair between a test set references and the corresponding model outputs. Nevertheless, it still demonstrates that the vocabulary is substantially less diverse in the model outputs. Consider, for example, the ViGGO dataset, for which the model output set is exactly a third of the size of the test set references, yet the delexicalized vocabulary across the model outputs is only about 22%, and the number of unique words only 9%, of their test set counterparts.

Both of these datasets have a reasonably small test set, allowing us to run experiments with many different MCTS configurations, and to execute multiple runs for each configuration. This is particular important because Batch-MCTS relies on sampling. For all the experiments that involve sampling, the results we report are averaged over 3 independent runs.

9.2 Evaluation

In this section, we provide details on the baselines for our experiments, and we give an overview of the automatic metrics we use and the human evaluation we perform to assess the performance of our proposed inference method compared to the baselines.

9.2.1 Baselines

In order to evaluate how Batch-MCTS performs, we compare it with multiple other inference algorithms, some of which we already mentioned earlier as standard methods for inference in NLG. These will serve as baselines in terms of overall quality, semantic accuracy and diversity of utterances that an NLG model can produce. All of the methods described below perform the utterance decoding in

a left-to-right fashion.

Greedy search. The simplest inference method is greedy search, which has the decoder select the most probable token at each time step of generating an utterance until it produces the end-of-sequence token, at which point it stops. More formally, the i-the token in the utterance being generated is chosen as follows:

$$w_i = \underset{w \in \mathcal{V}}{\operatorname{argmax}}\, p_\theta(w|w_{<i}, x) \ ,$$

where $p_\theta(w|w_{<i}, x)$ is the probability of the i-th token given input x and all tokens $w_{<i}$ generated so far in the utterance by an NLG model with parameters θ, and $\mathcal{V}$ is the model's vocabulary.

Beam search. A more robust extension of greedy search is beam search, which does a better job at identifying high-probability tokens hidden behind low-probability ones, possibly leading thus to an utterance with a higher overall probability. It achieves this by simultaneously keeping track of B most probable *beam hypotheses* (i.e., partial utterances in our case), instead of choosing the token with the highest probability at each time step. The hypotheses are updated at each time step with the most probable single-token extensions across all current hypotheses. Beam search ultimately produces a set of B utterances, which allows for further reranking according to different criteria, such as semantic accuracy, if desired.

Diverse beam search. While beam search can find better solutions than greedy search, the top B candidates it generates often differ only marginally from each other (especially with a lower B), and that usually towards the end of the utterance. To

overcome this shortcoming, Vijayakumar et al. (2018) proposed a modified version of beam search that promotes diversity among the beam hypotheses. They do this by partitioning the beam hypotheses into G equally-sized subsets, where $G \leq B$, and enforcing dissimilarity between them. The amount of desired diversity among the candidate utterances can be controlled by the *diversity strength* parameter λ, which modifies the weight of the dissimilarity term in the objective function. The dissimilarity between groups can be calculated in various ways, but by default it penalizes tokens selected for hypotheses in other groups at the same time step, which the authors claim to perform better than several other more complex functions. For more details, we refer the reader to Vijayakumar et al. (2018). Diverse beam search can, in theory, lead to an even more probable utterance being discovered than the standard beam search would find, but we are primarily interested in this method for its diversity-promoting properties.

Nucleus sampling. Sampling-based decoding chooses the i-th token according to the conditional probability distribution across the vocabulary at time step i:

$$w_i \sim p_\theta(w|w_{<i}, x) \; ; \; \forall w \in \mathcal{V} \; .$$

Unrestricted sampling can, however, quickly lead to disfluency and incoherence. To minimize the chance of sampling an ill-fitted word in the given context, Holtzman et al. (2019) introduced nucleus sampling. This method filters the smallest possible subset of tokens whose cumulative probability exceeds certain probability p (hence sometimes referred to as top-p sampling), provided as a parameter. The token is then sampled from this limited set $\mathcal{V}_{\text{top-}p}$ of the most probable tokens with the

probability mass redistributed among them. This results in a dynamic resizing of the token set to sample from at each time step depending on the current context. Nucleus sampling thus improves over top-K sampling, a different approach suggested by Fan et al. (2018), which samples from a fixed set of the K most probable tokens at each time step. Similar to beam search, with nucleus sampling we can also produce a pool of candidates (independently sampled) to be reranked based on other criteria.

9.2.2 Automatic Metrics

For the purposes of automatic evaluation of generated utterances, we use several of the previously mentioned metrics: BLEU, METEOR, BLEURT, BERTScore and language model perplexity (PPL). To calculate corpus-BLEU and METEOR we use the E2E NLG Challenge evaluation script[2]. For the two reference-based neural metrics, BERTScore and BLEURT, we use their respective Python packages, `bert_score` and `bleurt`. In contrast to using these metrics for state evaluation in Batch-MCTS, here we opt for the larger versions of their models, namely deberta-xlarge-mnli for BERTScore and D12 for BLEURT. We calculate BERTScore without IDF weighting and baseline rescaling.

While the first four metrics serve mainly for evaluating the semantics and faithfulness to the inputs (indirectly by comparing the generated utterances to human-written references), PPL measures the overall language fluency of the utterances irrespective of the inputs. To calculate this metric we utilize the implementation of the GPT-2 language model in the Transformers library (Wolf et al., 2020). We use the medium-sized variant of the model (gpt2-medium). Note that PPL increases as the model's confidence about the tokens it generates decreases,

[2] https://github.com/tuetschek/e2e-metrics

so in general, we want PPL to be as low as possible. However, in a setting where the model output diversity is of interest, higher PPL (but still reasonably low) may merely imply the use of rarer words and formulations, which may make an utterance all the more interesting.

In addition to the standard metrics above, we also report the slot error rate (SER) calculated by our heuristic slot aligner. This is calculated on a corpus level as the proportion of erroneous slots out of all slots in the test set. This is therefore a metric that we want the generated utterances to minimize, ideally achieving 0%.

9.2.3 Diversity Metrics

Our primary objective in this chapter is to investigate to what degree Batch-MCTS is capable of increasing the diversity of model outputs without it being at the expense of their fluency and semantic accuracy. Since we have established a set of automatic metrics that we will use in our experiments for evaluating fluency and semantic accuracy, what remains now is for us to introduce metrics to gauge the diversity. We measure the diversity of the language used in model outputs by calculating different corpus statistic across all of the utterances generated by a model for a test set. The metrics we report are the following:

- **Vocabulary size.** The number of distinct words across all utterances.
- **Bigram vocabulary size.** The number of distinct bigrams across all utterances.
- **Distinct-N.** Introduced in Li et al. (2016a), this metric calculates the ratio of distinct N-grams to the total number of N-grams across all utterances. We report it for N $\in \{1, 2\}$.
- **Unique-N.** The number of N-grams occurring only once across all utterances.

We report this metric for N $\in \{1, 2\}$.

- **Unique utterance templates.** This metric requires utterances to be delexicalized, i.e., have slot mentions replaced with a placeholder (see below for details). It then calculates what proportion of these utterance "templates" are unique among all the outputs. This metric, introduced in Oraby et al. (2019), reveals how varied the general structure of utterances is in the model outputs.
- **Average utterance length.** The average number of words in the utterances. Although this is not a diversity metric per se, we include it here, as it characterizes a corpus in a similar way vocabulary size does.

For the first four metrics, we calculate the scores on delexicalized utterances as well. A delexicalized utterance has certain slot mentions replaced with a special placeholder word, which is different for each slot type, such as `_|developer|_` for a mention of the DEVELOPER slot in the ViGGO dataset. In general, we delexicalize categorical, numeric and list slots (e.g., NAME, RELEASE_YEAR or PLATFORMS), but leave Boolean and scalar slots (e.g., HAS_MULTIPLAYER or RATING) untouched because their mentions often cannot be identified in the utterance by simply matching the slot value.

9.2.4 Human Evaluation Criteria

We use automatic metrics to evaluate vast amounts of outputs across a great number of experiments in order to study general trends across different Batch-MCTS configurations and to identify the most promising ones among them. After narrowing it down to a few candidates for which we would like to judge the quality of the model outputs more accurately and to compare them with the baselines, we

carry out a human evaluation. This evaluation consists in manually annotating generated utterances according to a set of criteria corresponding to different types of errors that can occur in an utterance.

We tuned the criteria across 3 trial annotations, each performed on model outputs (generated using Batch-MCTS with 3 different configurations) for the whole ViGGO test set, i.e., $1,077$ examples in total.[3] We eventually settled on 8 error types across 4 categories: DA errors, slot errors, semantic errors and syntactic errors, described in Table 9.2. These categories cover all the errors we encountered in the trial annotations, which there were a few hundred of. The model outputs used in the trials are not part of the human evaluation in Section 9.4.7.

9.3 Standard Inference Method Experiments

Before diving into experiments with Batch-MCTS, we provide a comprehensive overview of how the baseline inference methods described in Section 9.2.1 compare to each other on the ViGGO dataset. We split the results into two tables, one with the automatic semantic accuracy and fluency metrics (Figure 9.1), and the other showing the diversity metrics (Figure 9.2). Figure 9.1 includes the vocabulary size too as a representative of the diversity metrics, in order to allow the reader to view the relationships between semantic accuracy, fluency and diversity all in one place. As we will later see in Figure 9.2, all the diversity metrics are strongly correlated with the vocabulary size.

[3] There are 359 unique MRs in the ViGGO test set.

Error type		Description
DA error		Utterance formulated in a way inconsistent with the corresponding DA type definition/expectations in the context of the dataset. This is typically a domain-specific error, and may require a strong familiarity with the dataset to be recognized.
Slot	Duplicate mention	Undesirable repeated mention of the same slot in the utterance (not an error if the DA type requires/allows it for the given slot).
	Incorrect mention	Slot mention with a different value than indicated in the input MR. This includes, among others, negated mentions of Boolean slots and partial mentions of list slots (i.e., when not all the elements listed in the value are mentioned).
	Omission	Missing slot mention.
Semantic	Hallucination	Information in the utterance that cannot be inferred from the input MR, even if it is technically true/factual and not contradicted by the input.
	Incoherence	Illogical, inconsistent, or unclear expression (e.g., "I enjoy playing adventure games, but I don't like them." or "It is a first-person shooter and is played from a bird's eye view."). An incoherence can reach across sentences.
Syntactic	Fluency / grammar	Grammatical error, missing/incorrect article, wrong preposition, unexpected word, unfinished sentence, gibberish, etc.
	Punctuation / capitalization	Incorrect/unexpected capitalization of a letter or a word (e.g., name of a game in lowercase), or missing/incorrect essential punctuation (e.g., a period used at the end of a direct question). We are lenient with commas and hyphens, and typically do not consider them to be errors, unless, say, missing from the name of a game (i.e., "M-rated" and "M rated" would both be considered correct).

Table 9.2: Classification and description of the final set of error types we distinguish in the human evaluation of generated utterances.

9.3.1 Semantics vs. Diversity

Assuming that the vocabulary size faithfully represents the overall diversity of the model outputs, we observe a very clear inverse relationship between diversity and semantic accuracy across the four inference methods. As Figure 9.1 shows, using beam search produces outputs whose combined vocabulary is around 560 words, i.e., the lowest among all the methods, whereas its BLUE, METEOR, BERTScore and BLEURT scores are the highest, often with a significant lead over the other methods. The fluency, as measured by the PPL metric, follows generally the same trend as the reference-based automatic metrics (only with PPL it is the lower the better). Finally, the slot error rate (SER) is highly dependent on whether the semantic reranking of generated candidates was used after decoding or not. Looking at the results without reranking, we see that SER increases hand-in-hand with the diversity. Reranking typically forces down the SER close to zero, but considering the slot aligner focuses on detecting fact omissions and substitutions, model outputs with SER of 0% can still be incoherent or contain hallucinations. Over the following few paragraphs, we describe in more detail the effects of the individual inference methods.

Greedy search. Greedy search, being the most naive decoding method, will serve more as a baseline to compare the other methods with. Notably, the diversity is not the lowest using greedy search, as one might have expected, however, SER is in the higher range.

Beam search. Beam search leads to significant improvements in all metrics except for the vocabulary size. This is presumably a consequence of beam search optimizing for the overall utterance probability even more aggressively than greedy

Inference method	Reranking	BLEU	METEOR	BERT-Score	BLEURT	PPL ⬇ [43.49]	SER ⬇ [2.24%]	Vocab [1292]
External baselines								
Slug2Slug	yes	0.519	0.388	0.872	0.681	35.23	2.53%	**589**
DataTuner (no FC)	no	0.534	0.391	**0.877**	0.688	**33.43**	2.99%	587
DataTuner	yes	**0.536**	**0.394**	**0.877**	**0.690**	33.73	**1.68%**	**589**
Internal baselines								
GS	no	0.519	0.387	0.869	0.676	37.70	1.65%	590
BS (B=10)	no	0.540	0.392	0.877	0.689	36.81	0.95%	562
	yes	**0.541**	**0.393**	**0.878**	**0.690**	36.92	0.24%	561
BS (B=20)	no	0.538	0.391	0.877	0.689	36.95	0.95%	561
	yes	0.539	0.392	0.877	**0.690**	36.92	0.17%	560
DBS (B=10, λ=1.0)	no	0.526	0.388	0.871	0.677	36.66	1.36%	580
	yes	0.524	0.389	0.871	0.677	36.73	0.07%	581
DBS (B=10, λ=5.0)	no	0.522	0.387	0.871	0.677	37.45	1.58%	584
	yes	0.521	0.389	0.871	0.677	37.50	0.12%	585
DBS (B=20, λ=1.0)	no	0.526	0.388	0.871	0.677	**36.53**	1.29%	580
	yes	0.524	0.388	0.871	0.677	36.68	0.02%	582
DBS (B=20, λ=5.0)	no	0.521	0.387	0.870	0.677	36.91	1.48%	587
	yes	0.518	0.388	0.870	0.677	36.91	**0.00%**	592
NS (p=0.3)	no	0.519	0.387	0.868	0.677	38.47	1.63%	599
	yes	0.520	0.388	0.869	0.678	38.28	0.87%	599
NS (p=0.5)	no	0.510	0.384	0.866	0.671	38.25	2.09%	623
	yes	0.513	0.387	0.867	0.674	38.48	0.15%	624
NS (p=0.8)	no	0.467	0.372	0.851	0.657	41.65	2.72%	**688**
	yes	0.470	0.375	0.854	0.661	42.29	**0.00%**	687
Pearson correlation w/ BLEU		1	0.9822	0.9937	0.9532	-0.9274	-0.2855 (-0.9644)	-0.9845

Figure 9.1: Overview of T5-small's performance with different inference methods on ViGGO. The methods shown are greedy search (GS), beam search (BS), diverse BS (DBS), and nucleus sampling (NS), with different parameters and with semantic reranking on or off (performed by our slot aligner in the internal baselines). For NS, 10 independent candidates were generated when reranking was enabled. The header indicates scores calculated on the test set where applicable. The upper portion of the table also lists three external model baselines for comparison: Slug2Slug (Juraska et al., 2018) and two variants of DataTuner (Harkous et al., 2020), all of which use BS for inference. In the bottom row, the correlation coefficient for SER omitting reranked results is indicated in parentheses. SER and PPL have negative correlation with BLEU because these two metrics are to be minimized.

search, which further diminishes the generation of less common words. While in our experiments we observed a difference in scores between using $B = 5$ and $B = 10$, increasing the beam size further seems to have virtually no benefit at all.[4] As we can see in the row for "BS ($B = 20$)" with reranking enabled in Figure 9.1, it is only the SER metric that may slightly benefit from this, since among 20 candidates the slot aligner has a higher chance of finding a semantically accurate utterance than among just 10.

Diverse beam search. As expected, diverse beam search (DBS) increases the model outputs' diversity over standard beam search, but not dramatically. In fact, the vocabulary size did not even reach that of greedy search, except when using $B = 20$ and $\lambda = 5.0$, which represents the case with the most aggressive diversification. We attribute this to the fact that DBS promotes diversity within the B candidates it generates, but that does not necessarily propagate to corpus-level diversity, as DBS still optimizes for the overall utterance probability within each set of B candidates. Nevertheless, considering the jump in SER from 0.95% (using beam search) to 1.29–1.58% in exchange for the modest increase in diversity, we believe sampling from the B candidates instead of using the top one would likely only lead to more semantic errors. In terms of the other metrics, DBS is in the middle of the pack, comparable to greedy search, although PPL remains on par with that of beam search outputs, if not slightly better. We also experimented with using fewer groups than the beam size (i.e., setting G such that $G < B$), but the increase in overall diversity ended up being even lower, and hence we do not show the results.

[4] We note that this observation is specific to the T5-small model and the ViGGO dataset, and they cannot be generalized. It may well be the case that on a different dataset, using $B = 20$ would still lead to additional gains.

Nucleus sampling. The scores calculated for utterances generated with nucleus sampling vary widely depending on the p-value (see the bottom 6 rows in Figure 9.1). For $p = 0.3$ the results are similar to greedy search, which is not surprising given that nucleus sampling with $p = 0$ is equivalent to greedy search, and 0.3 is a rather small probability mass that probably rarely accommodates more than the single most probable token from the distribution. With the two higher p-values things become more interesting. Increasing the probability mass to 0.5 leads to a modest boost to the diversity (624 vs. 590, i.e., 6% over greedy search), at a minimal cost to the reference-based metrics. The semantic accuracy suffers though, with SER exceeding 2%. Reranking helps here by forcing SER down to 0.15% and even slightly improving most of the other metrics. Finally, with $p = 0.8$, which is the same value we use in the playouts of the Batch-MCTS inference, we achieve a highly superior diversity (688 vs. 590, i.e., a 17% increase over greedy search). This, however, comes at the expense of all other metrics, with BLEU dropping by more than 10% of the greedy search value (from 0.519 to 0.467) and PPL jumping up by 4 points. Although, with reranking, the SER drops all the way to zero, these outputs may be riddled with hallucinations and incoherent phrases, given these drastic reductions in metric scores. This we determine through a manual evaluation of these utterances, whose results we discuss in Section 9.4.

9.3.2 Other Diversity Metrics

Although in Section 9.3.1 we evaluated the effects of different inference methods on the overall vocabulary size across all generated utterances, we will now take a look at the remaining diversity metrics described in Section 9.2.3. First of all, we would like to bring attention to the fact that all the other diversity metrics

Inference method	Re-ranking	Vocab size [1292 / 951]		Bigram vocab [6879 / 5014]		Distinct-1 [4.25%]	Distinct-2 [23.56%]	Unique-1 [412 / 428]		Unique-2 [3101]	Unique templ. [99.91%]	Avg. length [23.85]
		Lex	Delex	Lex	Delex	Delex	Delex	Lex	Delex	Delex		
External baselines												
Slug2Slug	yes	**589**	**226**	**1870**	847	**3.58%**	14.23%	134	46	290	92.43%	**20.89**
DataTuner (no FC)	no	587	216	1845	832	3.56%	14.56%	**142**	**47**	302	96.10%	20.38
DataTuner	yes	**589**	218	1863	**850**	3.53%	**14.60%**	140	45	**306**	**96.38%**	20.68
Internal baselines												
GS	no	590	206	1864	844	3.19%	13.84%	140	39	322	96.28%	21.33
BS (B=10)	no	562	181	1735	705	3.08%	12.76%	135	33	234	94.61%	19.85
	yes	561	181	1739	709	3.07%	12.78%	135	34	239	94.61%	19.91
BS (B=20)	no	561	178	1728	696	3.05%	12.70%	134	31	227	94.80%	19.70
	yes	560	177	1731	699	3.02%	12.71%	133	30	229	94.80%	19.77
DBS (B=10, λ=1.0)	no	580	194	1835	810	3.09%	13.69%	134	32	294	96.19%	20.85
	yes	581	195	1856	825	3.09%	13.87%	136	33	305	96.19%	20.95
DBS (B=10, λ=5.0)	no	584	199	1851	823	3.16%	13.85%	136	33	294	96.19%	20.91
	yes	585	199	1880	841	3.13%	13.98%	133	30	302	96.19%	21.11
DBS (B=20, λ=1.0)	no	580	193	1832	808	3.07%	13.65%	135	31	292	96.19%	20.85
	yes	582	195	1860	828	3.09%	13.90%	135	32	305	96.19%	20.95
DBS (B=20, λ=5.0)	no	587	201	1859	828	3.18%	13.87%	138	36	301	96.10%	20.97
	yes	592	206	1902	860	3.22%	14.23%	138	36	322	96.10%	21.19
NS (p=0.3)	no	599	217	1923	902	3.39%	14.89%	150	51	369	96.47%	21.23
	yes	599	218	1923	899	3.38%	14.81%	150	53	367	96.47%	21.27
NS (p=0.5)	no	623	239	2070	1039	3.72%	17.16%	152	48	441	97.96%	21.14
	yes	624	240	2059	1030	3.71%	16.88%	158	53	441	97.96%	21.30
NS (p=0.8)	no	**688**	**311**	**2478**	**1430**	**4.72%**	**22.97%**	187	**89**	**733**	**99.35%**	21.64
	yes	687	309	2466	1411	4.69%	22.66%	**188**	**89**	725	**99.35%**	**21.67**
Pearson correlation w/ vocab size (lex)		1	0.9980	0.9972	0.9984	0.9805	0.9907	0.9702	0.9561	0.9954	0.9662	0.7500

Figure 9.2: Overview of T5-small's output diversity using greedy search (GS), beam search (BS), diverse BS (DBS), and nucleus sampling (NS) inference on the ViGGO dataset. The header indicates scores calculated on the test set. For further clarifications we refer the reader to the description of Figure 9.1.

are very strongly correlated with the vocabulary size (with the Pearson correlation coefficient $r \in [0.9561, 0.9984]$), except for the average utterance length, which, with $r = 0.75$, is strongly correlated (see the bottom row in Figure 9.2). As discussed earlier, the average length is not a diversity metric per se, so its lower correlation with the vocabulary size is expected.

The vocabulary size of delexicalized utterances correlates almost perfectly with the regular vocabulary size, but it offers a few useful insights. First, by comparing the sizes of the two vocabulary versions we can see that about 400 words actually correspond to names of games, developers, and other categorical video-game-related attributes, which are not present in the delexicalized version.[5] This metric thus illustrates an increase in diversity better than the regular vocabulary, since the above-mentioned set of 400 words remains constant across all experiments. It is the rest of the vocabulary that is variable and indicates thus the real diversity of the language used in the model outputs. In case of greedy search, this amounts to 206 words (see row "GS" in Figure 9.2). On the other hand, using nucleus sampling with $p = 0.8$, the delexicalized vocabulary size is 311. This corresponds to a 51% increase in diversity, as opposed to the 17% that we reported in the previous section using the regular vocabulary, and paints a more accurate picture of the difference in their language diversity.

The number of unique utterance templates is overall very high among the model outputs, but it still increases from 96.28% when using greedy search to 99.35% with nucleus sampling, and drops to 94.61% with beam search. This is further evidence for nucleus sampling encouraging more diverse utterance formulations, and

[5] Conversely, the delexicalized vocabulary has words the standard version does not – the special slot mention placeholders for the 9 slots that we delexicalize.

beam search making them more repetitive.

9.3.3 Summary

The observations across standard inference metrics in this section provide evidence for our conjecture that it is hard to achieve higher diversity among generated utterances without negatively affecting their semantic accuracy. To quantify the strength of the inverse relationship between these two aspects of generated utterances, we calculated the Pearson correlation coefficient between the vocabulary size and the BLEU metric (see the bottom row in Figure 9.1). With $r = -0.9845$, there is a very strong negative correlation, implying that while one of them increases, the other one decreases. All the other metrics strongly correlate with BLEU, including the fluency metric (PPL). The reference-based metrics all correlate very strongly with each other, with METEOR and BLEURT having the lowest correlation among all the pairs ($r = 0.9366$). BLEURT, in general, correlates the least with the other three metrics.

Although we provided the test set statistics in the table headers in Figures 9.1 and 9.2, in the case of most diversity metrics they are not directly comparable with the model output scores because there are 3 times more utterances in the test set than in any model output set (which has a single utterance per MR). Despite that, we can tell that the diversity of the human-written references is significantly greater than that of the generated utterances, even when using nucleus sampling with $p = 0.8$. The delexicalized vocabulary of the references is more than 3-times bigger than that of the nucleus sampling outputs (951 vs. 311), yet we could not expect nucleus sampling to triple the vocabulary size of its outputs by having it generate 3 independent utterances per input MR. In order to achieve that, those

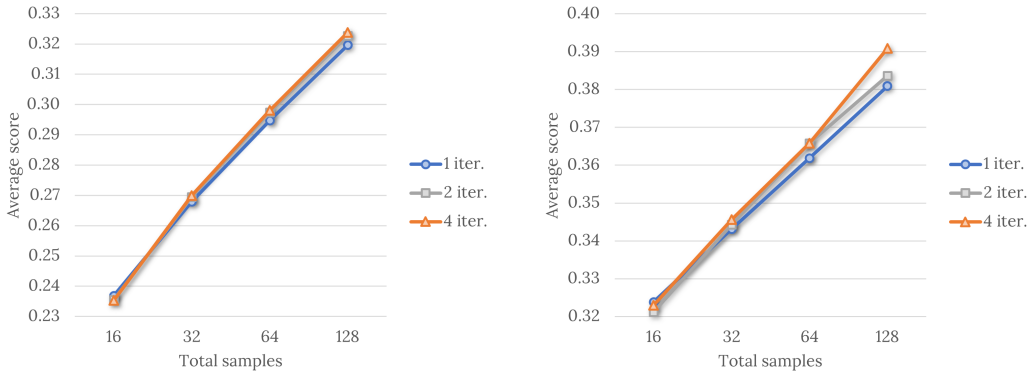
3 candidate utterances would have to use non-intersecting vocabularies, which is virtually impossible since they are supposed to be paraphrases of each other. In addition to the vocabulary size difference, the delexicalized references also contain disproportionately more unique words and bigrams (see columns Unique-1:delex and Unique-2:delex in Figure 9.2).

We also come to a conclusion that, especially for a small dataset like ViGGO, comparing diversity metric scores calculated on *delexicalized* utterances gives us a better idea about variations in the language used in utterances produced with different decoding techniques. Due to the very strong correlations observed among the metrics, going forward, for a better readability, we will only be reporting a small subset of them when evaluating model outputs.

9.4 Batch-MCTS Experiments

In the previous section, we compared the baseline inference methods with each other and determined which metrics are the most informative. Throughout this section we will thus be contrasting the Batch-MCTS results only with the baselines most relevant to the experiment at hand. We continue presenting our analyses on the ViGGO dataset, and we later discuss any differences we observe using the E2E dataset.

Throughout this section, we will be using the $I_{\text{MCTS}} \times B_{\text{MCTS}}$ notation to refer to Batch-MCTS configurations with I_{MCTS} iterations and batch size B_{MCTS}, for example, 4×32 for a configuration with 4 iterations and batch size 32.



(a) Average score of the top 10 candidates.

(b) Average score of the best candidate.

Figure 9.3: Average scores of the top 10 candidates vs. average scores of the best candidate only, using different numbers of MCTS iterations but fixed total numbers of samples. Batch size is implied by the number of iterations and samples, e.g., $(4 \text{ iter.}, 128 \text{ samples}) \implies 32$. The evaluation metric used was PPL+BERTScore. The scores are multiplied by 10, for a better readability, considering the reciprocal of PPL results in a small value. The scores are averaged over 3 independent runs of the Batch-MCTS inference for each configuration.

9.4.1 MCTS Metric Optimization

In order to verify that the Batch-MCTS algorithm works as expected, we look into how the average score of the internal utterance evaluation metric, which MCTS optimizes for, changes with an increasing batch size and number of iterations. Figure 9.3 plots the average scores for the PPL+BERTscore variant of the metric, which is calculated by multiplying BERTScore of an utterance candidate $\hat{y}$, given input x, with the reciprocal of the utterance's PPL (since PPL is being minimized):

$$score_{\text{PPL+BS}}(\hat{y}|x) = \frac{\text{BERTScore}(\hat{y}, x)}{\text{PPL}_{\text{GPT-2}}(\hat{y})} \ .$$

They are calculated across all generated utterances for the ViGGO test set. The two plots show the scores averaged over the top 10 candidates MCTS found for each input, and averaged over just the best candidates.

As we can see in Figure 9.3a, the average scores steadily increase as the

total number of samples increases, but there is only a marginal difference between 1 and 4 iterations (with batch size 128 and 32, respectively). Looking at the plot with the average best candidate scores (Figure 9.3b), we observe a similar overall incline, yet the incline is the steeper the more iterations MCTS uses. This becomes most apparent at 128 samples, at which point the lines appear to be divergent, suggesting that the benefit of a greater number of iterations will continue increasing with the number of samples. Since the configurations 1×128 and 4×32 end up sampling the same number of candidates during the inference, the latter identifying candidates with a higher score is a proof that the MCTS algorithm is effective at optimizing for the given metric.[6] On the other hand, the differences are rather small. This is a consequence of the MCTS playouts being *heavy*, i.e., the candidate utterance generation is highly informed (by the NLG model's learned probabilities), leading to very strong candidates right from the first iteration.

While this ability of Batch-MCTS to optimize for the internal evaluation metric is promising, we cannot say with certainty if it translates to overall better utterances. To determine that, we will evaluate the generated utterances using external metrics. But before jumping into that, we would like to point out one more thing in this analysis. Although the plot lines in Figure 9.3b, and possibly also in Figure 9.3a, are divergent, at sample size 16 the 1-iteration configuration almost always performs best (we observed this with other metric variations as well). Our interpretation of this is that, with only 4 or 8 samples in the first iteration (in the 4×4 and 2×8 configuration, respectively), there is a more limited variation among the utterance beginnings that the MCTS gets to expand on and further

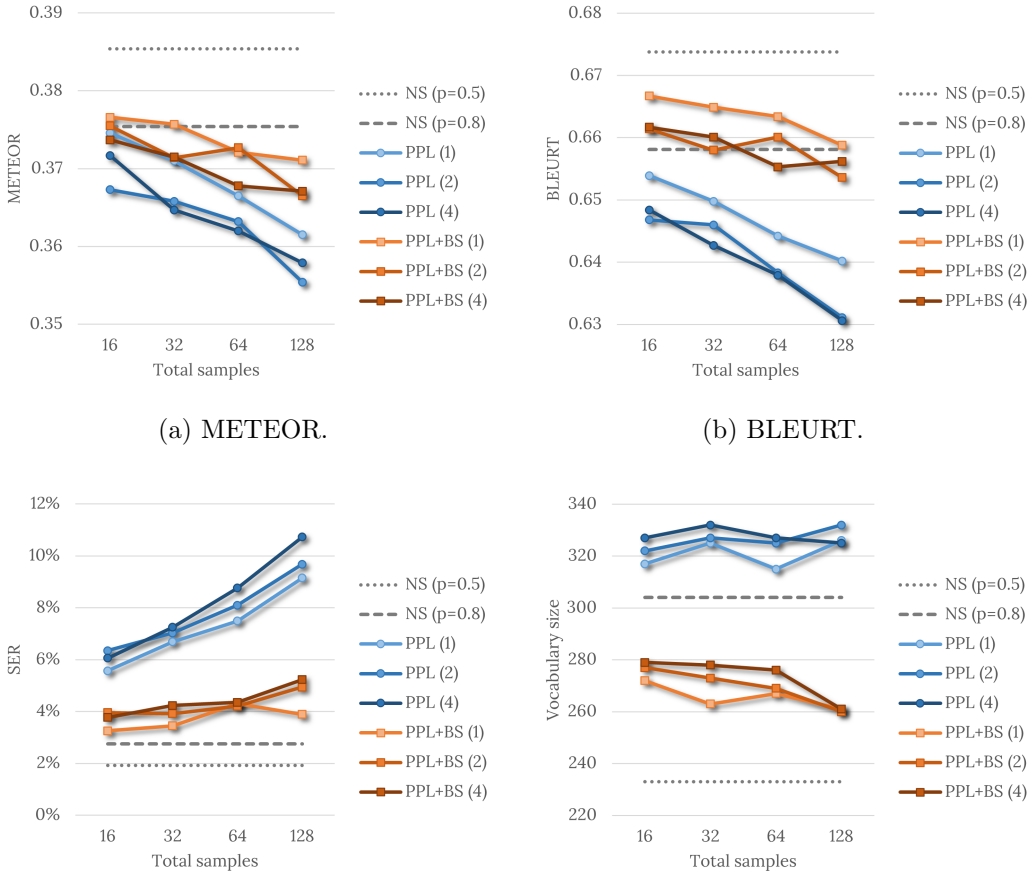
[6] Remember that MCTS with just a single iteration is equivalent to nucleus sampling with reranking by the metric score.

explore in the subsequent iterations. By the time the sample size and, with it, the batch size, doubles, the higher-iteration configurations do not suffer from this initial disadvantage anymore. This highlights the importance of the batch size in Batch-MCTS. In general, the higher the batch size the better, but after a certain threshold the benefit of a higher number of iterations will likely take over that of increasing the batch size. The threshold will vary from dataset to dataset.

9.4.2 PPL With BERTScore

Let us now look at how different metrics score the final utterances generated with Batch-MCTS, and how these model outputs fare against the baselines. Using PPL to guide MCTS, as part of the internal evaluation metric, we expect the utterance candidates to be relatively diverse, and therefore choose nucleus sampling (NS) as the most relevant baseline inference method to compare the results with. Figure 9.4 shows the results for the following metrics: METEOR, BLEURT, SER, and the vocabulary size of delexicalized utterances. In the plots, we also include the scores for Batch-MCTS guided by PPL only, so that we could gauge the effect of the BERTScore component of the MCTS metric.

First, let us focus on the differences between individual Batch-MCTS configurations. In the plots, all configurations with the same number of iterations are shown as a separate series. For the PPL+BERTScore configurations, we will be looking at the three orange series, corresponding to 1, 2 and 4 iterations with varying batch sizes. What we immediately see is that with an increasing number of iterations, the scores become worse across all metrics except for the vocabulary size, where it improves (see Figures 9.4a, 9.4b, 9.4c vs. Figure 9.4d). This suggests that the MCTS algorithm indeed promotes diversity, however, not without taking a toll



(a) METEOR.

(b) BLEURT.

(c) Slot error rate (SER ↓).

(d) Vocabulary size (delexicalized).

Figure 9.4: Metric scores for utterances generated with Batch-MCTS using two variants of the state evaluation metric: **PPL+BERTScore** (PPL+BS for short) and **PPL** only. The number of iterations is indicated in parentheses after the name of the metric, and the batch size is implied by the number of iterations and samples. The baselines here are nucleus sampling (NS) with $p \in \{0.5, 0.8\}$.

on the utterances' semantic accuracy. We conjecture this effect may be a consequence of the metric used to guide the MCTS. Nevertheless, the scores gradually become worse here also as the number of samples increases. This, surprisingly, is the case with the vocabulary size as well.

Moving on to comparing the Batch-MCTS scores with those achieved with NS, we note that the scores across all PPL+BERTScore configurations are closest to NS with $p = 0.8$, i.e., the most diversity-promoting one. While in the METEOR and BLEURT scores there are relatively small differences, SER is where NS significantly outperforms Batch-MCTS (see Figure 9.4c). To make things even worse for Batch-MCTS, its outputs' vocabulary size remains 8–14% smaller (Figure 9.4d). On the other hand, among the metrics not shown, we should mention that PPL scores range from 24 (with sample size 128) to 30 (with sample size 16) for Batch-MCTS outputs, whereas for NS with $p = 0.8$ PPL is 41.05. This is a significant difference, but not at all surprising given the fact that PPL is a part of the metric Batch-MCTS optimizes for.

Next, let us evaluate the contribution of the BERTScore component in the MCTS metric by comparing the PPL+BERTScore variant with the PPL-only variant. Since BERTScore represents the metric's semantic accuracy component, we are most interested in how it affects the SER scores. Comparing the three orange series with the three blue series in Figure 9.4c, we see that BERTScore is responsible for a drastic reduction in SER, specifically from about 5.5–11% (when using PPL only) down to 3.2–5.7%, which amounts to an almost 50% reduction of semantic errors. That being said, the resulting SER range is still too high. Besides SER, the BERTScore component significantly improves the METEOR and BLEURT scores as well (see Figures 9.4a and 9.4b). It does, however, drag the vocabulary size down

from a level that was about 10% higher than that of the NS outputs. Considering the overall poor scores of the PPL-only variant though, we suspect that its use of the larger vocabulary is actually counterproductive and incoherent.

Finally, we observe a strong impact of the BERTScore component on the length of the generated utterances, encouraging them to be shorter. The average utterance length dropped from 25–26.5 to 22.5–23.5 words, which is, nevertheless, closer to the average utterance length in the test set (23.85 words). This effect is presumably a direct consequence of comparing utterance candidates with pseudo-references, which only contain content words, and thus any extra words in the utterance will be considered hallucinations by BERTScore. Therefore, the longer the utterance, the lower the BERTScore, as it is only looking for content words in the utterance.

9.4.3 PPL With BLEURT

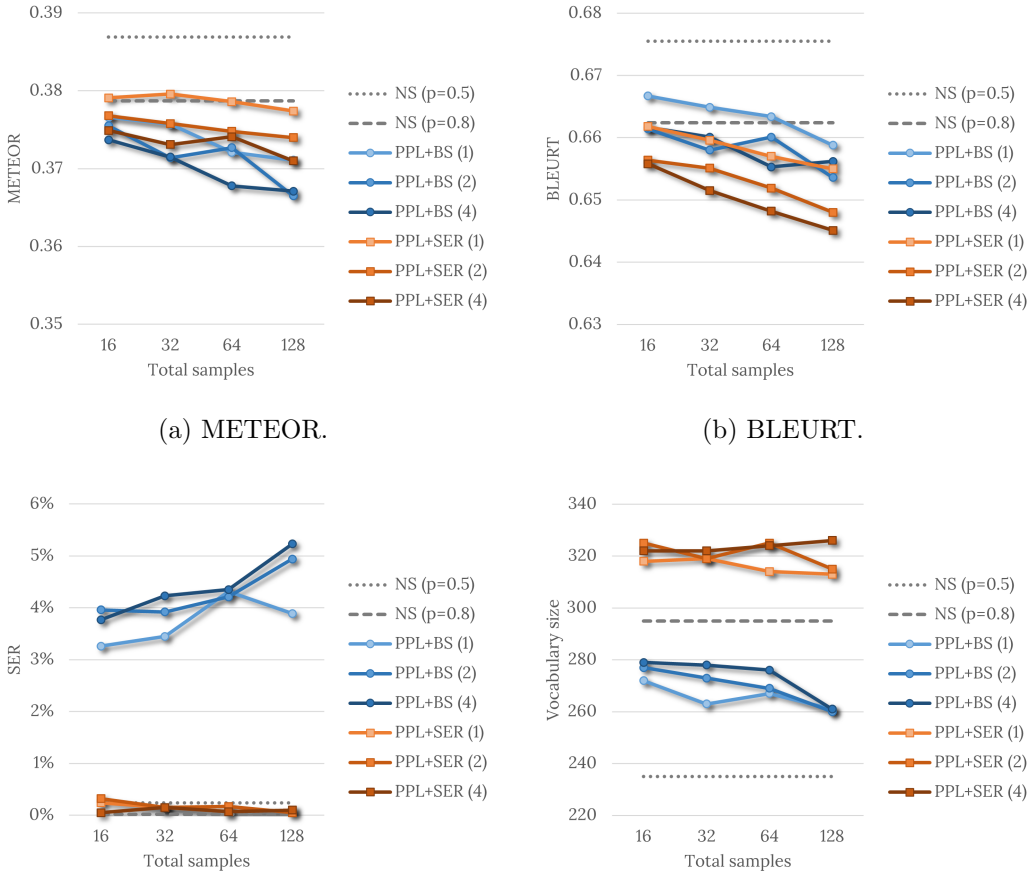
Swapping BERTScore for BLEURT in the role of semantic accuracy in Batch-MCTS has a negative impact on its performance across all metrics, including SER, which ranges from 4–7%, depending on the configuration. The vocabulary size remains slightly above the levels of NS with $p = 0.8$, but there is no solace in that considering the overall weaker performance than the NS baseline. We speculate that this may be a consequence of the smaller range of values BLEURT uses in practice when scoring utterances. Increasing the weight of the BLEURT component in the metric could possibly address this, nevertheless we leave the component weighting for future work.

9.4.4 PPL With SER

Since the PPL+BERTScore metric variant does not successfully guide the MCTS to semantically accurate utterances, we experiment with substituting the slot aligner for the BERTScore component, and see if the SER metric offers better direction in semantic accuracy. To ensure a fair comparison with the baselines, we choose their counterparts with semantic reranking, i.e., those that also take advantage of the slot aligner. Below, we analyze what changes in metric scores swapping BERTScore for SER brings. Figure 9.5 summarizes the results across the same four metrics as Figure 9.4.

Before discussing the Batch-MCTS scores, we note that the NS baselines ($p = 0.5$ and $p = 0.8$) with reranking have overall slightly higher scores across all metrics than previously without reranking, but most importantly, their SER drops to 0.24% and 0.02%, respectively. With the slot aligner component, it is now a more even game for Batch-MCTS. Its METEOR scores are only slightly below those of NS ($p = 0.8$), and SER hovers just above zero (see Figures 9.5a and 9.5c). In BLEURT, however, Batch-MCTS lags behind the NS baselines (Figure 9.5b). At the same time, probably the most interesting observation is that the SER component of the MCTS state evaluation metric helps preserve the vocabulary size, which stays in a similar range to when using only PPL to guide the MCTS (compare Figure 9.5d with 9.4d). As a result, the vocabulary is 6–11% bigger than that of the NS baseline.

Overall, using the slot aligner in place of BERTScore as the semantic accuracy component of the MCTS metric leads to dramatic changes across all metrics. Besides reducing SER from the 3–5% range, the vocabulary is up to 60 words richer. These changes are accompanied by a moderate increase in METEOR and a moderate



(a) METEOR.

(b) BLEURT.

(c) Slot error rate (SER ↓).

(d) Vocabulary size (delexicalized).

Figure 9.5: Metric scores for utterances generated with Batch-MCTS using the **PPL+SER** state evaluation metric variant, along with **PPL+BERTScore** (PPL+BS for short) for comparison. The number of iterations is indicated in parentheses after the name of the metric, and the batch size is implied by the number of iterations and samples. The baselines here are nucleus sampling (NS) with $p \in \{0.5, 0.8\}$ and with slot aligner-based reranking enabled.

decrease in BLEURT, which makes it rather difficult to come to a conclusion about coherence, and even the true semantic accuracy (including hallucinations). The average length of utterances generated with this variant of Batch-MCTS ranges from 24.98 to 27.43 words, which is significantly greater than the 22.5–23.5 words with the PPL+BERTScore metric variant, suggesting that hallucinations may indeed be what keeps the vocabulary size inflated.

One final observation we have from the plots in Figure 9.5 is that the vocabulary size and SER remain virtually flat across different sample sizes instead of becoming worse, as was the case with the PPL+BERTScore variant. BLEURT's rate of decline remains approximately the same, while METEOR becomes decidedly flatter. Ideally, the metric scores would be increasing with the number of samples, not decreasing.

9.4.5 Adding Model's Own PPL

In the previous two experiments, we saw that using BERTScore as the semantic accuracy component in the MCTS metric is relatively effective, although not as much as SER. It appears, however, that it is PPL that pulls Batch-MCTS in a wrong direction – to utterances that do not compare well with the references and that decrease in semantic accuracy, yet do not make up for it with a higher diversity. We can see this clearly in Figure 9.4, where most of the metric scores for all the PPL configurations (blue) become sharply worse as the number of samples, as well as the number of MCTS iterations, increases. Perhaps, that would have been acceptable if the vocabulary size increased correspondingly, but that is not the case (Figure 9.4d).

The PPL metric must have a rather different idea of what a good utterance

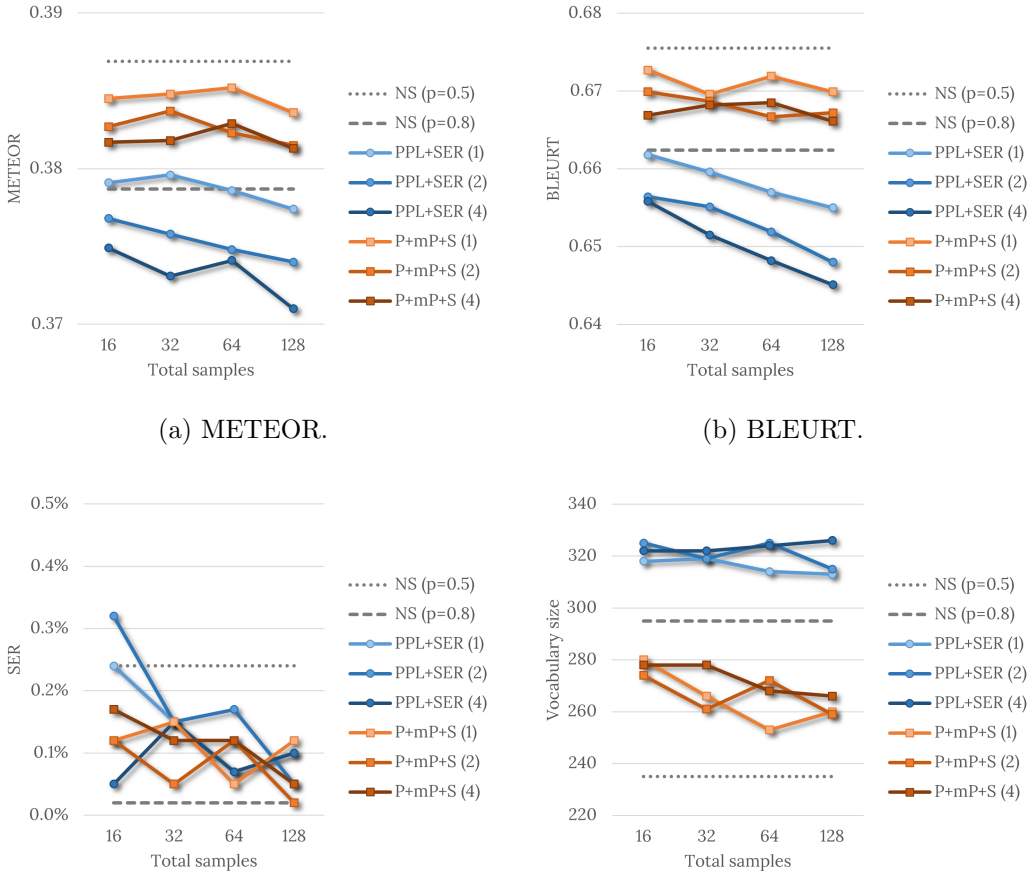
looks like in our particular domain. Additional evidence for this lies in the fact that, guided by PPL only, Batch-MCTS generates utterances that score on average in the 17–22 range in PPL (which the MCTS optimized for), whereas the average PPL across the human-authored utterances in the test set is 43.49. The GPT-2 model behind the PPL metric may deem the references significantly less probable, but, having been written and revised by humans, they are unlikely to be less fluent. There might still be use for PPL though, as long as it is "tamed" by an additional metric.

We therefore experiment with adding the model's own perplexity (mPPL) as a third component in the MCTS state evaluation metric. The scoring function thus becomes:

$$score_{\text{P+mP+S}}(\hat{y}|x) = \frac{1}{\text{PPL}_{\text{GPT-2}}(\hat{y}) \cdot \text{mPPL}(\hat{y}) \cdot \big(1 + \text{ERR}(\hat{y}|x)\big)} \,.$$

In theory, this should make Batch-MCTS more aware of candidate utterances that are fluent and natural in the given domain, while retaining a certain amount of the general diversity-promoting property from the PPL component.

Let us examine how this MCTS metric variant affects the balance of semantic accuracy and diversity in the generated utterances. Figure 9.6 compares the new results with those using PPL+SER in order to demonstrate the effect of adding the new mPPL component to the metric. Both METEOR and BLEURT this time remain relatively flat across different total numbers of samples, while SER generally decreases with more samples explored. These are all desirable trends, however, the vocabulary size gradually declines (Figure 9.6d). Perhaps more relevant is the fact that the vocabulary is 15–20% smaller than without the mPPL component. In



(a) METEOR.

(b) BLEURT.

(c) Slot error rate (SER ↓).

(d) Vocabulary size (delexicalized).

Figure 9.6: Metric scores for utterances generated with Batch-MCTS using the **PPL+mPPL+SER** (P+mP+S for short) state evaluation metric variant, along with **PPL+SER** for comparison. The number of iterations is indicated in parentheses after the name of the metric, and the batch size is implied by the number of iterations and samples. The baselines here are nucleus sampling (NS) with $p \in \{0.5, 0.8\}$ and with slot aligner-based reranking enabled.

all the other metrics though, the current variant significantly outperforms the one without mPPL (Figures 9.6a, 9.6b and 9.6c).

Looking at the scores with respect to the baselines, the results are, in fact, promising. According to METEOR and BLEURT, the model outputs are closer in overall quality to those generated by NS with $p = 0.5$, yet in SER and vocabulary size they are closer to NS with $p = 0.8$, which is the better one according to these two metrics. This, combined with the fact that the vocabulary size without using the mPPL component was higher than that of NS with $p = 0.8$, suggests that, using the current state evaluation metric variant, we could find a Batch-MCTS configuration that could produce outputs with a diversity on par with NS ($p = 0.8$) but better semantic accuracy.[7] This could presumably be achieved by giving the mPPL component a slightly lower weight, in order to allow for more diversity. However, instead of optimizing for this with the goal of matching the vocabulary size of the NS ($p = 0.8$) baseline, we perform a manual evaluation of the generated utterances, as this will likely offer additional insights into how they compare in different semantic and syntactic aspects that cannot be captured by automatic metrics. The results and discussion of the manual evaluation will follow in Section 9.4.7.

9.4.6 Replacing GPT-2 PPL With Model's Own PPL

In our final set of experiments, we study the effect of removing the GPT-2 PPL component from the MCTS state evaluation metric and letting the model's own PPL (mPPL) take full control of the fluency aspect of the metric. This will, naturally, limit the diversity because mPPL will score the same utterances the highest

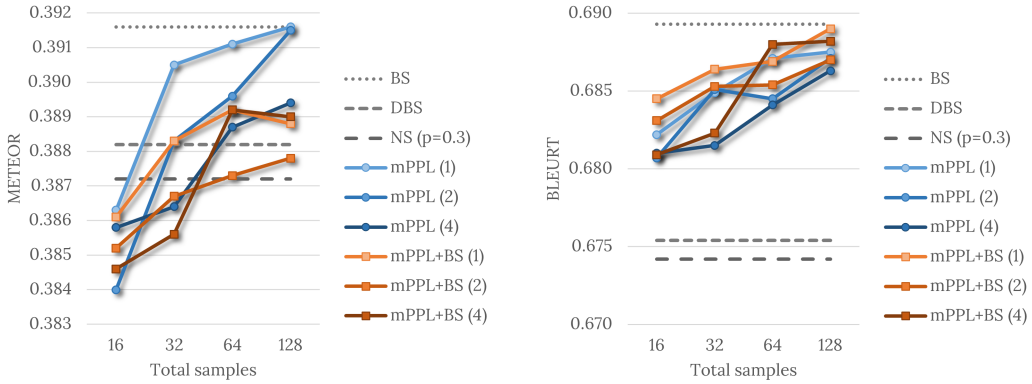
[7] Although SER is as low as 0.02% for NS with $p = 0.8$, as we pointed out previously, the SER metric has its limitations and various semantic errors, such as hallucinations and incoherence, will slip through its fingers. Therefore, we consider a combination of multiple metrics, e.g., METEOR, BLEURT and SER here, a better indicator of the utterances' overall quality.

as greedy or beam search would. However, together with a semantic accuracy-oriented component, such as BERTScore or SER, it has the potential to guide the sampling-based search to more diverse model outputs than diverse beam search, while maintaining the same semantic accuracy.

Considering the overall shift of metric scores for outputs generated using mPPL instead of GPT-2 PPL, we switch to comparing the results with beam search (BS), diverse beam search (DBS), and nucleus sampling with $p = 0.3$. All three of these are relatively close to each other in terms of metric scores, with BS being the best among them in METEOR, BLEURT, and SER, but with the lowest vocabulary size (see Figure 9.7). Nucleus sampling is on the opposite end of the spectrum, with the biggest vocabulary but the lowest performance in the remaining metrics. DBS is in between, closer to the NS performance in all metrics but vocabulary size. We report the results on the mPPL+BERTScore metric variant and later briefly discuss the differences when SER is used in BERTScore's place.

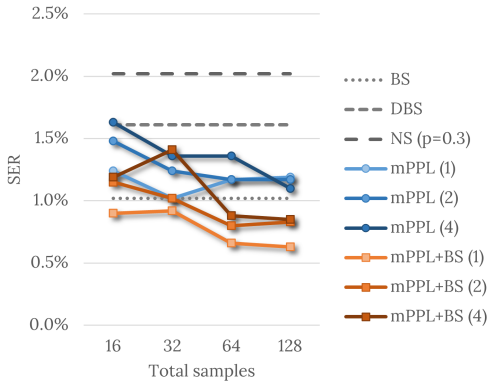
Model PPL. Let us first look at how Batch-MCTS performs when we let the NLG model guide the search algorithm by itself, i.e., using mPPL to compare utterance candidates. As the total number of samples increases, the scores of the model outputs improve steadily in all metrics except for the vocabulary size, which is on a steady decline instead (see the three blue series in the Figure 9.7 plots). The scores vary substantially across different configurations, which makes comparing concrete configurations with individual baselines more practical.[8] With the 4×4 configuration (i.e., corresponding to the data points of "mPPL (4)" at 16 samples), Batch-MCTS achieves NS levels of diversity while significantly outperforming NS in

[8]Considering all the scores are averaged across 3 independent runs, any unusual performance spikes from a "lucky run" for a particular configuration should be at least partially smoothed out.

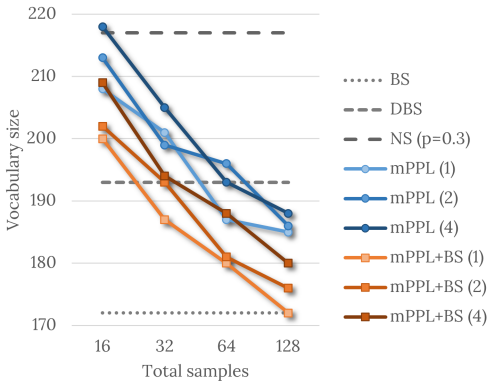


(a) METEOR.

(b) BLEURT.



(c) Slot error rate (SER ↓).



(d) Vocabulary size (delexicalized).

Figure 9.7: Metric scores for utterances generated with Batch-MCTS using two variants of the state evaluation metric: **mPPL+BERTScore** (mPPL+BS for short) and **mPPL** only. The number of iterations is indicated in parentheses after the name of the metric, and the batch size is implied by the number of iterations and samples. The baselines here are beam search (BS) with $B = 20$, diverse BS (DBS) with $B = 20$ and $\lambda = 1.0$, and nucleus sampling (NS) with $p = 0.3$.

all other metrics (including BLEU, BERTScore and PPL, which are not plotted), except for METEOR where it is 0.0014 below. Comparing with DBS, configurations 2×16 and 2×32 (i.e., "mPPL (2)" at 32 and 64 samples) result in a richer vocabulary and superior performance in the other three metrics at the same time. As for the BS baseline, the 1×128 and 2×64 configurations get very close to the BS performance across all metrics, yet with an 8% greater language diversity. These results, including BLEU, BERTScore and PPL scores, are summarized in Figure 9.8.

Model PPL + BERTScore. After adding the BERTScore component, we observe very similar general trends, but slightly shifted (see the three orange series in Figure 9.7 plots). In case of BLEURT and SER, this variant performs overall better, while the model outputs are scored lower by METEOR. The diversity is also negatively affected, dropping by 5–10%. What is most notable about this variant is that in most of the configurations Batch-MCTS achieves significantly higher diversity with a lower SER than BS. The best example would probably be the 4×16 configuration with the vocabulary size of 188 (in contrast to BS's 172) SER of 0.88% (vs. 1.02%). While BLEURT is only marginally lower for the Batch-MCTS outputs in this configuration, there is a bigger difference in METEOR. We therefore cannot conclude with a high confidence, just based on these automatic metrics, which of these two is more semantically faithful to the inputs.

Model PPL + SER. With access to a tool like our slot aligner, the performance of Batch-MCTS further improves. In contrast to the mPPL+BERTScore variant, mPPL+SER consistently achieves zero SER under all configurations, which we stress is not the case with all baselines equipped with slot aligner-based reranking. In fact,

Inference method	Parameters	BLEU	METEOR	BERT Score	BLEURT	PPL ⬇	SER ⬇	Vocab size (delex)
BS	B=20	**0.541**	**0.392**	**0.876**	**0.689**	37.23	1.02%	172
MCTS	mPPL, 1x128	0.533	**0.392**	0.875	0.688	**37.09**	1.19%	185
	mPPL, 2x64	0.537	**0.392**	0.875	0.687	37.40	1.17%	**186**
	mP+BS, 4x16	0.525	0.389	0.874	0.688	37.72	**0.88%**	**188**
DBS	B=20, λ=1.0	0.527	0.388	0.871	0.675	**36.55**	1.61%	193
MCTS	mPPL, 1x32	0.530	**0.391**	**0.875**	**0.685**	37.84	**1.02%**	**201**
	mPPL, 2x16	0.525	0.388	0.874	**0.685**	37.31	1.24%	199
	mPPL, 2x32	**0.535**	0.390	0.874	**0.685**	37.36	1.17%	196
NS	p=0.3	0.519	0.387	0.865	0.674	39.14	2.02%	217
MCTS	mPPL, 4x4	**0.524**	0.386	**0.872**	**0.681**	**38.03**	**1.63%**	**218**

Figure 9.8: Overview of the best Batch-MCTS configurations with the **mPPL** and **mPPL+BERTScore** (mP+BS for short) state evaluation metric. The baselines, shown in light gray rows, are beam search (BS), diverse BS (DBS), and nucleus sampling (NS).

only DBS reaches 0%, whereas BS drops to 0.22% and NS ($p = 0.3$) stays above 1%. In addition to eliminating slot errors, mPPL+SER in general slightly improves on the mPPL-only variant (see the blue series in Figure 9.7) in all aspects, with certain configurations even surpassing BS in the METEOR metric. Meanwhile, the scores of the reranking-enhanced baselines do not shift much compared to their values in Figure 9.7, except for the SER metric, as described above. This further increases the edge the various Batch-MCTS configurations have over their respective baselines described in detail in the previous paragraphs. The best configurations using mPP+SER are summarized in Figure 9.9.

Summary In Sections 9.4.2 through 9.4.5, we evaluated the performance of Batch-MCTS using GPT-2 PPL as the primary fluency component, but what we discovered is that this metric actually guides the algorithm away from good solutions, leading to overall mixed results depending on the semantic accuracy component. Here we therefore experimented with replacing the PPL metric with the NLG model's own PPL. This inevitably led to a lower diversity in model outputs, but a significantly

Inference method	Parameters	BLEU	METEOR	BERT Score	BLEURT	PPL ⬇	SER ⬇	Vocab size (delex)
BS	B=20	**0.541**	0.393	**0.877**	**0.690**	**37.17**	0.22%	172
MCTS	mP+S, 1x128	0.539	**0.394**	**0.877**	0.689	37.30	**0.00%**	182
	mP+S, 2x64	0.538	0.393	**0.877**	0.689	37.24	**0.00%**	**184**
DBS	B=20, λ=1.0	0.523	0.388	0.870	0.674	**36.70**	**0.00%**	198
MCTS	mP+S, 1x16	0.530	**0.390**	**0.874**	**0.685**	38.76	**0.00%**	207
	mP+S, 2x16	**0.533**	**0.390**	**0.874**	**0.685**	37.86	**0.00%**	201
	mP+S, 4x8	0.531	0.389	**0.874**	0.684	38.09	**0.00%**	**208**
NS	p=0.3	0.519	**0.389**	0.866	0.676	39.01	1.07%	215
MCTS	mP+S, 2x8	**0.520**	0.388	**0.872**	**0.681**	**38.65**	**0.00%**	215
MCTS	mP+S, 4x4	0.517	0.386	**0.872**	**0.681**	39.02	**0.00%**	**218**

Figure 9.9: Overview of the best Batch-MCTS configurations with the **mPPL+SER** (mP+S for short) state evaluation metric. The baselines, shown in light gray rows, are beam search (BS), diverse BS (DBS), and nucleus sampling (NS), all with slot aligner-based reranking.

higher quality overall. As we show in Figures 9.8 and 9.9, the Batch-MCTS inference method, in different configurations, became thus superior to three standard methods that optimize more for semantic accuracy than for diversity. Batch-MCTS generates significantly more diverse utterances than beam search, while maintaining the same performance across the metrics, and possibly with a higher semantic accuracy. At the same time, Batch-MCTS can match or even improve on the diversity achieved by nucleus sampling with $p = 0.3$, while radically improving the semantic accuracy and, with it, achieving a slightly higher scores in other metrics as well. Finally, diverse beam search gets outperformed by our method across the board, whether it is diversity, semantic accuracy, or BLEU scores.

9.4.7 Human Evaluation

So far throughout this section, we have performed a thorough analysis of the performance of Batch-MCTS compared to other inference methods, but only using automatic evaluation metrics. As we know, they are not the most reliable method to evaluate NLG outputs, especially when the score differences between the

outputs of two systems – or, in our case, two inference methods – are relatively small (see Section 1.4 for a more detailed discussion). Moreover, our slot aligner cannot detect hallucinations and, due to it operating on word/phrase level, recognizing incoherence and contradictions are also beyond its scope. Hence, we do not expect the SER scores that we observed in our experiments to be telling the whole story about semantic accuracy in the generated utterances. For a more accurate picture, we turn to human evaluation of the model outputs generated using different Batch-MCTS configurations, as well as those generated with standard inference methods, for comparison.

We wrapped up Section 9.4.5 with a conclusion that, based on automatic metric scores, Batch-MCTS outputs generated using the PPL+mPPL+SER state evaluation metric looked promising when compared to the most diversity-promoting baseline, NS ($p = 0.8$). We therefore systematically annotate the model outputs for 8 different types of errors (see Section 9.2.4). For Batch-MCTS, we annotate two configurations, 1×128 and 4×32, so as to determine what effect a higher number of iterations has if the number of samples is fixed. For the 4×32 configuration, we also experiment with sampling from the top 5 and top 10 candidates at the end of the inference, in an attempt to increase the diversity. The results, including automatic metrics for comparison, are summarized in Figure 9.10.

All Batch-MCTS configurations have a relatively even distribution of errors across the four error categories (DA, semantic, etc.). Among slot errors, duplicate and incorrect errors are the most common types. We observed that incorrect mentions typically happen in DAs that mention the RATING slot twice (e.g., *give_opinion* or *verify_attribute*), when the two mentions are inconsistent (e.g., "you disliked" paired with "are you a big fan of"). In this case, when at least one of the men-

Inf. meth. / MCTS metric	Parameters	DA errors ⬇	Slot errors ⬇				Semantic errors ⬇			Syntactic errors ⬇			TOTAL ERR.
			Dupl.	Incor.	Omis.	SUM	Halluc.	Incoh.	SUM	Flu. / gram.	Punc. / capit.	SUM	
Baselines													
DBS + rank.	B=20, λ=1.0	23	5	8	**0**	13	**1**	**14**	**15**	9	2	11	62
NS + rank.	p=0.5	**21**	**3**	**6**	1	**10**	2	19	21	**3**	**1**	**4**	**56**
	p=0.8	34	6	8	**0**	14	7	31	38	21	5	26	112
Batch-MCTS													
PPL + SER	4x32	42	26	19	1	46	9	40	49	15	4	19	156
PPL + mPPL + SER	1x128	29	15	**9**	2	26	5	**12**	**17**	20	4	24	96
	4x32	**20**	8	11	1	**20**	5	18	23	**9**	4	**13**	**76**
	4x32 (top 5)	28	9	13	**0**	22	**3**	23	26	21	**2**	23	99
	4x32 (top 10)	23	**3**	15	3	21	**3**	23	26	15	5	20	90

(a) Errors in utterances (absolute numbers). The total number of errors is indicated in the last column. For reference, the test set has 359 MRs with 1,370 slots in total across all of them.

Inf. meth. / MCTS metric	Parameters	BLEU	METEOR	BERT Score	BLEURT	PPL ⬇ [43.49]	SER ⬇ [2.24%]	Vocab size (delex) [951]	Unique templ. [99.91%]	Avg. length [23.85]
Baselines										
DBS + rank.	B=20, λ=1.0	**0.523**	**0.388**	**0.870**	**0.674**	**36.70**	**0.00%**	198	96.38%	20.99
NS + rank.	p=0.5	0.518	**0.388**	0.867	**0.674**	38.26	0.22%	232	97.77%	20.88
	p=0.8	0.472	0.376	0.853	0.661	40.97	**0.00%**	**294**	**98.89%**	**21.57**
Batch-MCTS										
PPL + SER	4x32	0.410	0.373	0.827	0.651	**18.81**	0.07%	**337**	98.89%	27.30
PPL + mPPL + SER	1x128	**0.453**	**0.385**	0.850	**0.669**	19.19	0.15%	261	97.77%	25.24
	4x32	0.442	0.383	0.849	0.666	19.09	0.07%	274	98.33%	25.28
	4x32 (top 5)	0.450	0.382	0.845	0.663	21.51	**0.00%**	278	**100.00%**	24.90
	4x32 (top 10)	0.447	0.382	**0.851**	0.663	23.33	0.22%	294	99.16%	**24.38**

(b) Automatic metrics, including a few diversity metrics.

Figure 9.10: Results of the human evaluation comparing the outputs of Batch-MCTS using 5 different configurations and those of 3 standard inference methods: diverse beam search (DBS) and two variants of nucleus sampling (NS), all equipped with semantic reranking. The last two rows of each table correspond to randomly sampling from the top 5 and top 10 Batch-MCTS candidates. The error types are explained in Table 9.2.

tions is found to be faithful to the input MR, the slot aligner considers the slot mentioned correctly. Incoherence dominates the semantic error category, whereas hallucinations are surprisingly rare. The *give_opinion* DA is the primary source of incoherence, as it is the most difficult DA in the ViGGO dataset to realize correctly due to its more open-ended nature, compared to the other DAs, and thus highly varied utterances in the training set. Finally, in the syntactic category, grammatical errors and various instances of disfluency occur substantially more frequently than punctuation and capitalization errors.

Comparing the outputs of different Batch-MCTS configurations (see the last four rows in Figure 9.10a), we can see a significant overall reduction in errors when running MCTS for 4 iterations with batch size 32 ($\sim$ 76 total errors) instead of a single iteration with batch size 128 ($\sim$ 96 errors). This suggests that the MCTS algorithm, guided by the PPL+mPPL+SER metric, is beneficial in finding better utterances. The automatic metric scores, however, imply the opposite, i.e., that the 1×128 configuration produces better utterances (Figure 9.10b). That is especially the case with BLEU (0.453 vs. 0.442), while METEOR, BERTScore and BLEURT only show minor differences. When sampling from the top 5 or 10 final candidates, the vocabulary size indeed increases, but so does the number of errors. Sampling from the top 10 seems to be no worse than top 5, yet it increases the vocabulary size more dramatically. We have also annotated one set of model outputs with the PPL+SER metric guiding the MCTS, which led to a significantly higher diversity (vocabulary size 337), but the generated utterances had by far the most errors (156), more than a double of Batch-MCTS with PPL+mPPL+SER and the same 4×32 configuration.

The best configuration of Batch-MCTS to compare with the NS ($p = 0.8$)

baseline is the 4×32 configuration with top-10 candidate sampling. Their vocabulary sizes are matching at 294 words, but the NS outputs have 24% more errors (112 vs. 90). That apparently does not prevent the BLEU score from being 2.5 points higher than that of the Batch-MCTS outputs. Among the reference-based metrics, only METEOR and BLEURT correctly reflect that the Batch-MCTS outputs are superior. For the regular 4×32 configuration, which has the least errors (76), BLEU score is even lower, whereas METEOR and BLEURT scores see a further increase. Noticing the difference in utterance lengths between the Batch-MCTS outputs and the NS baseline, we come to a conclusion that BLEU, being a precision metric, simply prefers shorter utterances. Indeed, the Pearson correlation coefficient between BLEU and the average utterance length across the 8 rows in Figure 9.10b is $r = -0.94$, indicating a very strong negative correlation.

As far as the correlation with the total number of errors is concerned, BLEU has by far the weakest ($r = -0.791$) and BLEURT the strongest correlation ($r = -0.946$), with METEOR in second place ($r = -0.930$) and BERTScore in third ($r = -0.905$).[9] One might expect the number of errors to increase with the average utterance length, yet none of the four error classes correlates strongly with the utterance length individually. There is, however, a non-significant strong correlation between the total number of errors and the utterance length ($r = 0.68$ with p-value of 0.064).

Finally, we observe that the GPT-2 PPL metric is not at all a good indicator of utterance quality in our domain. Its scores fail to capture system-level incoherence or syntactic errors. Compare, for example, the PPL+SER configuration

[9]The correlations are negative because we are maximizing the metric scores, but minimizing the number of errors.

(156 errors and PPL of 18.81) with DBS (62 errors but PPL of 36.7). PPL only has a very weak correlation with incoherence or the number of syntactic errors. Using PPL to guide MCTS in utterance fluency does not appear to be very effective either, considering the number of syntactic errors in the Batch-MCTS outputs is not significantly lower than in the NS ($p = 0.8$) outputs, except for the regular 4×32 configuration (Figure 9.10a). Although incoherence is substantially less prevalent in the Batch-MCTS outputs (17–26 vs. 38), this may also be the effect of the other state evaluation metric components, i.e., mPPL or SER in this case.

9.4.8 Qualitative Analysis

So far, we have been evaluating the model outputs exclusively based on various metrics and semantic criteria. But what do these generated utterances actually look like when using different inference methods?

In Table 9.3, we can compare the outputs of the same model using different inference methods. These utterances were generated for an MR with one of the simpler DAs (*request*) and only two slots, yet we can see that each inference method resulted in a substantially different utterance. While all of them are fluent and faithful to the input MR, the Batch-MCTS output stands out with its more elegantly formulated question ("Is there a game that you found to be soothing to play?") than what the other methods produced (e.g., "What's a soothing game...?" or "Do you know of any soothing games...?"). Furthermore, the second sentence of the utterance contains a nice elaboration "that I can play with my older kids", which sounds very natural in this context.[10] Nevertheless, the Batch-MCTS utterance

[10] Although one might be inclined to consider this phrase a borderline hallucination, since "older kids" is implied by the ESRB slot's "E 10+" value but having kids is not implied by the MR. Nevertheless, we consider this analogical to the HAS_MULTIPLAYER[yes] slot realization "You can play this game multiplayer with your friends.", which we consider valid in the context of ViGGO.

MR	*request* (ESRB [E 10+ (for Everyone 10 and Older)], SPECIFIER [soothing])
Reference #1	Did you play any Everyone 10 and up games that were particularly soothing this year?
Reference #2	What's a soothing rated E10+ game I can play with my brothers?
Reference #3	I'm looking for an E 10+ rated game that is soothing, do you know of any?
GS	What's a soothing E 10+ game you've played lately?
BS ($B = 20$)	What do you think is the most soothing E 10+ game you know of?
DBS ($B = 20$, $\lambda = 1.0$)	Do you know of any soothing games rated E 10+ (for Everyone 10 and Older)?
NS ($p = 0.5$)	Have you ever played a game that was rated E 10+ and was really soothing?
NS ($p = 0.8$)	What's a soothing game rated E 10+?
Batch-MCTS	Is there a game that you found to be soothing to play? I'm looking for something E 10+ that I can play with my older kids?
Batch-MCTS (top 10 sample)	What do you think is the most soothing E 10+ game you've played lately?

Table 9.3: Example utterances generated for the same MR in the ViGGO test set (with the *request* DA type) using different inference methods. Batch-MCTS outputs are shown for the 4×32 configuration using the PPL+mPPL+SER metric variant. The other inference methods are greedy search (GS), beam search (BS), diverse BS (DBS) and nucleus sampling (NS), with parameters indicated in parentheses. For comparison, the 3 human-written reference utterances from the test set are included as well.

has a punctuation error in the form of a question mark at the end of a statement. On the other hand, the utterance sampled from the top 10 candidates generated using Batch-MCTS (see the last row in the table) is more similar to the utterances generated by other inference methods; in fact, this particular utterance is a blend between the beam search and greedy search outputs. Finally, we note that the utterance generated with NS ($p = 0.8$) is particularly terse in this example, which is somewhat surprising and demonstrates that this inference method – using the most lenient sampling approach – produces outputs that are more hit-or-miss than others.

We also show example utterances generated for an MR with a more interesting DA type (*give_opinion*) in Table 9.4. We note that *give_opinion* is more open-ended than the other DAs in ViGGO, allowing for a significantly greater variety of MR realizations, which also makes it the most difficult DA for models to generate correct and fluent utterances for, as we observed in our manual annotation. With inference methods that maximize utterance probability, the utterances typically degenerate into *inform*-style utterances with no personal evaluation or opinion expression (compare the GS, BS, DBS, and even NS ($p = 0.5$) rows in the table with the three reference utterances). The NS ($p = 0.8$) output starts off in a promising way, but soon becomes incoherent ("the first person perspective is the one that I've tried was The Room") and remains incoherent even in the second sentence. Our Batch-MCTS method, on the other hand, despite internally also using nucleus sampling with $p = 0.8$, produces fully coherent and semantically correct *give_opinion* utterances for the same MR. The one sampled from the top 10 candidates (in the last row), although less verbose and more *inform*-like than the top candidate (in the "Batch-MCTS" row), contains a personal evaluation ("one of my favorite games") and qualifies thus as a *give_opinion* DA, similar to the style of reference #3. In

MR	*give_opinion* (NAME [The Room], RATING [excellent], GENRES [indie, point-and-click, puzzle], PLAYER_PERSPECTIVE [first person])
Reference #1	I thought that a first person perspective would be a bad fit for an indie point-and-click puzzle game, but in The Room it instead just added to the unique experience and made it one of the best games I've ever played.
Reference #2	Man, I loved The Room. Such an innovative first person puzzle game. I know the indie scene has been particularly enamored with point-and-click games as of late, but trust me when I say this one stands head and shoulder above the rest.
Reference #3	The Room was absolutely amazing. Indie first person point-and-click puzzle games like that are my favorite.
GS	The Room is an excellent indie point-and-click puzzle game. It's a first person game, and it's a first person game.
BS ($B = 20$)	The Room is an excellent indie first person point-and-click puzzle game.
DBS ($B = 20$, $\lambda = 1.0$)	The Room is an excellent indie point-and-click puzzle game. It's a first person game, and it's a first person game.
NS ($p = 0.5$)	The Room is an excellent first person point-and-click puzzle game.
NS ($p = 0.8$)	I love Indie point-and-click puzzle games, and the first person perspective is the one that I've tried was The Room. It's one of my favorite games of all time, and the indie one I know was just really great.
Batch-MCTS	The Room is one of the best indie point-and-click puzzle games I've ever played. I like indie games that have a first person perspective, and the point-and-click genre is just a huge hit.
Batch-MCTS (top 10 sample)	The Room is one of my favorite games. It's an indie point-and-click puzzle game that you play from a first person perspective.

Table 9.4: Example utterances generated for the same MR in the ViGGO test set (with the *give_opinion* DA type) using different inference methods. Batch-MCTS outputs are shown for the 4×32 configuration using the PPL+mPPL+SER metric variant. The other inference methods are greedy search (GS), beam search (BS), diverse BS (DBS) and nucleus sampling (NS), with parameters indicated in parentheses. For comparison, the 3 human-written reference utterances from the test set are included as well.

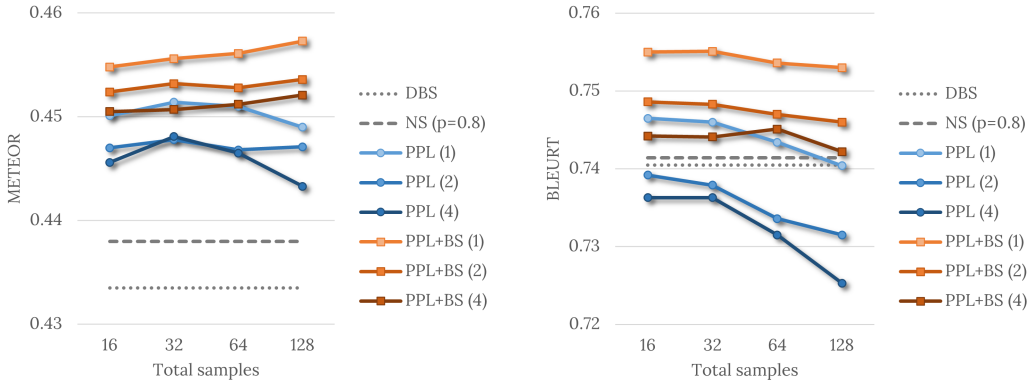
contrast, references #1 and #2 are more verbose and free-form, which, from our observations using smaller pretrained LMs, is beyond the ability of the models fine-tuned on the ViGGO dataset to reproduce in their outputs in a coherent way.

Although these were just two examples from the test set, in our manual annotation results we observed overall better coherence in utterances generated by Batch-MCTS than NS with $p = 0.8$. This is also reflected by the lower number of incoherence errors in Table 9.10a for the Batch-MCTS outputs than there are for the NS outputs. On the other hand, we noticed that, while the diversity of Batch-MCTS outputs is overall comparable to those generated with NS, the utterance variety within certain DAs (e.g., *confirm* or *request_explanation*) is rather low because they all use similar formulations. This, however, suggests that by stimulating diversity of utterances for the same DA type, we may be able to achieve a further increase in overall Batch-MCTS output diversity. In order to promote more variety on the DA level, we could, for example, enforce low similarity among the utterances in the Batch-MCTS candidate pool, when being updated at the end of each iteration. We leave these experiments to future work.

9.4.9 E2E Comparison

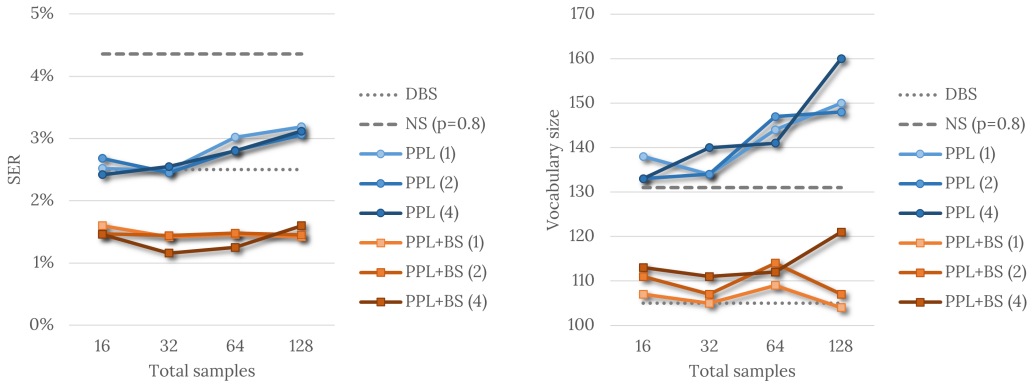
Although we studied the performance of Batch-MCTS in different settings and configurations primarily using a model trained on the ViGGO dataset, we conducted experiments with the E2E dataset as well. We only report the most important differences compared to the ViGGO experiments.

The biggest difference we observe is that using GPT-2 PPL is actually effective on the E2E dataset. Both on its own and combined with BERTScore, it guides the MCTS to diverse utterances that have dramatically higher semantic



(a) METEOR.

(b) BLEURT.



(c) Slot error rate (SER ↓).

(d) Vocabulary size (delexicalized).

Figure 9.11: Metric scores for utterances generated with Batch-MCTS on the E2E dataset using two variants of the internal utterance evaluation metric: **PPL+BERTScore** (PPL+BS for short) and **PPL** only. The number of iterations is indicated in parentheses after the name of the metric, and the batch size is implied by the number of iterations and samples. The baselines here are diverse beam search (DBS) with $B = 20$ and $\lambda = 5.0$, and nucleus sampling (NS) with $p = 0.8$.

accuracy than the baselines. As the plots in Figure 9.11 reveal, using PPL only ($\sim$ the three blue series) produces utterances with a substantially larger vocabulary than even nucleus sampling (NS) with $p = 0.8$, while reducing the SER from 4.36% to the 2.42–3.19% range, depending on the configuration. METEOR remains higher, but BLEURT drops below the scores achieved with NS. Adding the BERTScore component, on the other hand, helps further increase the semantic accuracy: SER drops to 1.16–1.6%, and METEOR and BLEURT both significantly increase over the PPL-only variant's performance (see the three orange series in Figure 9.11). As expected, such large gains in semantic accuracy do not come for free – the vocabulary size dropped by 15–30%. While this is a drastic drop, it is at a level slightly above the diverse beam search (DBS) baseline in terms of diversity, but in a league of its own according to the the other metrics.[11]

Switching from GPT-2 PPL to the model's own PPL (mPPL) drastically reduces the effectiveness of BERTScore in its role of semantic guidance. Considering E2E is a very noisy dataset, with a relatively high proportion of errors in the references, an NLG model trained on E2E inevitably learns to reproduce these errors. As a result, mPPL leads the MCTS to utterances with increasingly more errors. The Boolean slot FAMILYFRIENDLY) is responsible for virtually all of the slot errors, and BERTScore (as well as BLEURT) only has a limited ability to recognize those, as we saw in Chapter 8. Hence, the model outputs generated with the mPPL+BERTScore metric variant exhibit only a less than 10% improvement in SER than using mPPL by itself. Replacing BERTScore with the slot aligner, which is highly accurate in detecting Boolean slot errors, the story is similar to that on the ViGGO dataset, yet

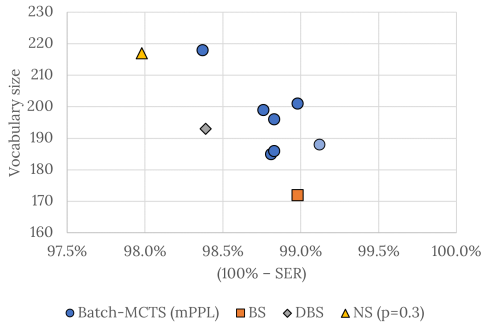
[11] On the E2E dataset, DBS produces significantly more diverse utterances than standard BS. For comparison, DBS with $B = 20$ and $\lambda = 5.0$ results in a delexicalized vocabulary size of 105 words, whereas with BS ($B = 20$) it only reaches 60, with greedy search 68, and with NS ($p = 0.5$) 81 words.

with Batch-MCTS having an even more pronounced edge over the various baselines.

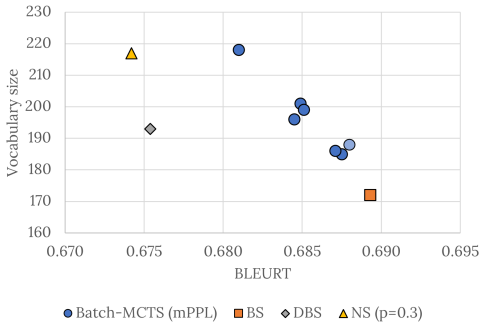
9.5 Discussion

This chapter combined the methods proposed and described in Chapters 7 and 8 into a full-fledged inference method, which we conducted extensive experiments with and thoroughly evaluated by comparing it with multiple standard inference methods across various criteria. Using several different variants of the MCTS state evaluation metric, our experiments revealed interesting, and to a certain degree unexpected, results that we summarize and discuss below.

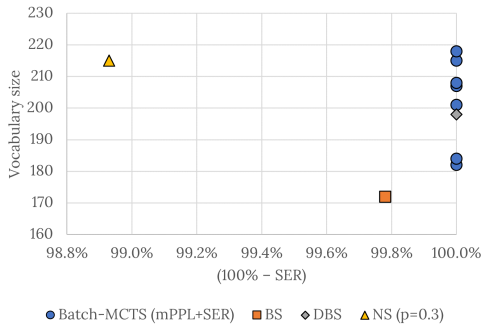
Our first finding is that optimizing strongly for PPL clearly leads the Batch-MCTS inference astray, presumably imposing sentence structures on the generated utterances that are preferred by the GPT-2 model, but in a direction away from the desired outputs in our domain. With PPL being part of the optimization metric in Batch-MCTS, the model outputs exhibit substantially lower PPL than that of the outputs generated using any of the baseline inference methods. Although we consider PPL to be a fluency metric, the difference between, say, 19 ($\sim$ Batch-MCTS outputs) and 37 ($\sim$ beam search), though seemingly large, does not necessarily imply superior fluency in the Batch-MCTS outputs. For comparison, PPL of the human-authored reference utterances in the test set is 43.49. All these values are in a relatively low PPL range. Therefore, the lower PPL in these experiments likely only reflects the fact that the utterance formulations are those preferred by the GPT-2 model, which calculates the PPL scores, based on its training data distribution. That being said, we used the smallest version of the GPT-2 model in the experiments we reported our results on, and perhaps using the larger variants instead would lead



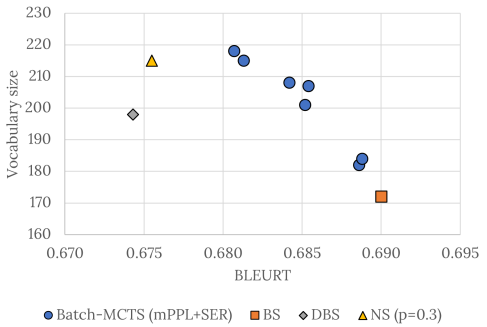
(a) Semantic accuracy vs. diversity in Batch-MCTS outputs using the **mPPL** metric variant. The light blue dot corresponds to an mPPL+BERTScore configuration.



(b) BLEURT vs. diversity in Batch-MCTS outputs using the **mPPL** metric variant. The light blue dot corresponds to an mPPL+BERTScore configuration.



(c) Semantic accuracy vs. diversity in Batch-MCTS outputs using the **mPPL+SER** metric variant.



(d) BLEURT vs. diversity in Batch-MCTS outputs using the **mPPL+SER** metric variant.

Figure 9.12: Semantic accuracy and diversity trade-off in model outputs produced by Batch-MCTS compared to 3 baselines: beam search (BS), diverse BS (DBS) and nucleus sampling (NS) with $p = 0.3$. The blue dots represent various Batch-MCTS configurations, corresponding to those in Figures 9.8 and 9.9.

to a more desired behavior. In fact, we observed a definite improvement across all metrics when we replaced the smallest GPT-2 model with its medium version, which promises a potentially desirable behavior of Batch-MCTS with a sufficiently large language model.

Due to our hardware constraints, we explored a different direction instead. We tried adding the NLG model's PPL as a third component to the metric, in order to impose more control over the generated utterances and tap more into the knowledge of the NLG model learned from the training data, instead of leaving it entirely

to GPT-2. This proved rather effective, and prompted us to experiment with entirely replacing GPT-2 PPL with the model's own PPL, which we had originally not intended to. Using the model's PPL on its own to guide Batch-MCTS, we were able to outperform both the standard and diverse beam search, as well as nucleus sampling with $p = 0.3$, when considering the diversity and the semantic accuracy performance at the same time (see scatter-plot visualizations in Figures 9.12a and 9.12b). With SER included for semantic guidance, Batch-MCTS became even more robust and easily exceeded the performance of the standard methods even when equipped with semantic reranking (see Figures 9.12c and 9.12d).

Through our manual annotation of 8 different types of errors in generated utterances, we found that Batch-MCTS can produce outputs as diverse as NS with $p = 0.8$, yet with a significantly lower overall error rate. Besides that we confirmed that GPT-2 PPL is not a suitable evaluation metric in our domain, as evidenced by its very weak correlation with both syntactic and semantic errors. Even more interestingly, we showed that BLEU behaves in a severely undesirable way, scoring system outputs more according to their length than the errors present in them. BLEURT and METEOR, on the other hand, were very strongly correlated with the actual errors in the annotated outputs, making them likely the best choice for evaluating generated utterances, at least in the domain of the ViGGO dataset.

To conclude these three chapters on Batch-MCTS, we developed a highly configurable and effective inference method, which can optimize for an arbitrary metric or a combination of multiple metrics. Although we achieved a superior performance compared to standard inference methods, there is a number of potential improvements to Batch-MCTS that could lead to further performance and diversity gains. Besides using a larger language model, normalization and weighting of the

state evaluation metric's components could provide better control, while tuning the numerous MCTS parameters (such as the exploration coefficient or the p-value of the internal nucleus sampling) could lead to a better balance between diversity and semantic accuracy.

Chapter 10

Conclusions and Future Work

Dialogue systems require high semantic fidelity of their responses, so as to remain true to the real world or the knowledge for the task at hand, as well as to the context of the conversation. If they provide the user with incorrect or contradicting information, they will quickly lose the user's trust. Task-oriented dialogue systems typically rely on data-to-text natural language generation (NLG), producing responses conditioned on structured input data that specifies the dialogue act type and content to be expressed in the response. While large pretrained generative language models (LMs), such as GPT-2 or T5, perform excellently at generating fluent text, when fine-tuned for such a data-to-text NLG task, even they often fail to produce an utterance that correctly mentions all the information provided in the input and avoids including extraneous information not grounded in the input.

The majority of previous work in neural data-to-text NLG focuses on optimizing model outputs to resemble a limited set of reference texts, which implicitly ensures that their semantic accuracy is relatively low, but it neglects the important aspect of diversity altogether. The aim of conversational AI, after all, is to allow people to interact with different types of technology through language, i.e., in a way

natural to humans, yet the experience will not feel natural unless the language used by the AI exhibits the variety and creativity of human speech.

In this thesis, we showed that the diversity of model outputs in data-to-text NLG tends to be a fraction of that observed in human-written references for the same inputs. Moreover, explicitly optimizing for semantic accuracy typically drives the already rather low diversity further down. We therefore explored ways to make the generated utterances faithful to their inputs, while using richer language, closer to that of humans. We developed new methods that help data-to-text NLG models automatically assess and enforce semantic accuracy, yet allow for higher language diversity in the generated utterances.

In this final chapter, we summarize our findings, including the limitations of our proposed methods, and we discuss possible future directions of the work.

10.1 Conclusions

10.1.1 Overview

In Chapter 1, we defined the task of MR-to-text generation and motivated the need for diversity in a dialogue system's responses, while simultaneously ensuring their highest possible semantic accuracy. We also provided an overview of the state-of-the-art approaches to language generation, reviewed previous work on semantic control and stylistic variation in NLG, and discussed a variety of automatic metrics commonly used for evaluating model outputs in data-to-text NLG and their limitation. In Chapter 2, we introduced the data-to-text datasets we would use for training and testing of our models throughout the thesis, including ViGGO, a new corpus we created to address some of the limitations of the existing datasets. We

described our slot aligner – a heuristic tool we developed for automatically finding a semantic alignment between an utterance and the corresponding input MR – and the various tasks it can be used for, in Chapter 3. In Chapter 4, we went over the different architectures of our models and evaluated them on the E2E and ViGGO datasets as baselines for our subsequent experiments. We then studied the effects of two different stylistic control methods on the diversity and semantic accuracy of the model outputs in Chapter 5. In Chapter 6, we presented and evaluated SeA-GuiDe, a new method for attention-based encoder-decoder models to track the semantic accuracy of a generated utterance during the decoding. In Chapters 7–9, we described Batch-MCTS, a new diversity-promoting inference method based on Monte-Carlo Tree Search, then studied the behavior of different automatic referenceless metrics to guide the MCTS algorithm to fluent and semantically correct utterances, and finally, thoroughly evaluated Batch-MCTS combined with the best three metrics, comparing its with standard inference methods.

In this thesis, we focused on the task of generating an utterance from a structured meaning representation (MR), nevertheless, we expect most of our proposed methods, perhaps with small modifications, to generalize to other data-to-text problems.

10.1.2 Semantic Accuracy

To achieve high semantic accuracy in generated utterances, neural models for data-to-text NLG have invariably been reliant on extrinsic components or methods, which typically require training a separate classifier or changing the NLG model's architecture. While the slot aligner (Chapter 3) we developed early on is such an extrinsic method, it served for multiple other purposes throughout the thesis

than just increasing the semantic accuracy of the generated utterances. The other method we proposed, SEA-GUIDE (Chapter 6), can be used out of the box on an already trained model and in any domain, as it merely enhances the way how the model takes advantage of the knowledge learned from the training data during inference. Here we go over these two methods and how effective each of them is in improving semantic accuracy.

First, the slot aligner uses heuristic rules and a dictionary of semantically equivalent expressions to detect slot mentions in the utterance. It categorizes slots into 5 general types (Boolean, numeric, scalar, categorical, and list), handling each of them in a different way. This part of the slot aligner is domain-agnostic, and merely requires that for each new dataset/domain it is indicated which slots are of which type. To further increase the slot aligner's accuracy in a new domain, its semantic dictionary can be populated with alternative expressions for specific slots and their values. Since the slot aligner identifies the exact slot mention positions, it can be used for training data manipulation or augmentation, such as splitting utterances into individual sentences and associating each sentence with a new MR consisting only of slots mentioned in the sentence. Most importantly though, the slot aligner can be used to calculate the slot error rate (SER) for generated utterances, evaluating thus their semantic accuracy. We used SER as one of our automatic evaluation metrics in virtually all experiments throughout the thesis, revealing that standard off-the-shelf reference-based metrics, such as BLEU or ROUGE, are not very sensitive to slot errors (omissions, substitutions, etc.), unless multiple of them are present in an utterance. They can easily end up rating an utterance with an error higher than one that is faithful to the MR but has less in common with the corresponding references. A human evaluation of the slot aligner, on the other

hand, showed that it is 95–100% accurate in identifying errors the ViGGO, E2E and MultiWOZ datasets.

Another useful application of the slot aligner is in candidate utterance ranking. With any of our models, we can overgenerate utterances (such as by using beam search, nucleus sampling or our Batch-MCTS inference) and then rerank the pool of candidate utterances based on the number of errors detected in them by the slot aligner. Using a transformer-based model trained from scratch on the ViGGO dataset, reranking helped reduce SER almost 3-fold, with ca. 2.5% of slots remaining erroneous, mostly because of out-of-vocabulary issues. With pretrained LMs (T5 and BART) that we fine-tuned on ViGGO, the reranking consistently brings down SER almost to zero on the E2E dataset (from about 3–4% when using beam search), 0.2–0.4% on ViGGO (typically a 4–5-fold reduction from beam search), and 0.4% on MultiWOZ (corresponding to an almost 3-fold reduction). As we mentioned above, standard reference-based metrics are largely insensitive to slot errors, but getting rid of errors through reranking, we still observed a small bump in METEOR, BERTScore and BLEURT scores on ViGGO, and a significant one in BLEU on the E2E dataset.

In contrast to the slot aligner, SEA-GUIDE requires virtually no manual adaptation in a new domain. It works by automatically extracting interpretable information from the attention weights at each time step during decoding, inferring from it which slot in the input MR was mentioned at that step, if any, and marking it in the MR. The slots that remain unmarked at the end of the decoding are considered to have been mentioned incorrectly or not at all. When SEA-GUIDE is combined with beam search, this information can be directly used to rerank the pool of candidate utterances according to their inferred semantic accuracy. By testing

SEA-GUIDE on three datasets (ViGGO, E2E and MultiWOZ), using two different models (T5 and BART) of two different sizes each, we showed that this method is highly domain-transferable and model-independent. The reduction in SER with SEA-GUIDE is on par with the slot aligner on the E2E dataset (i.e., down to almost zero), and almost 2-fold for the smallest models on ViGGO and MultiWOZ both (compared to beam search). These are large improvements in semantic accuracy, not to mention SEA-GUIDE tends to also slightly improve the other automatic metric scores, similar to the slot aligner.

Although developed with data-to-text NLG in mind, with some simple input preprocessing, SEA-GUIDE may be effective in text-to-text generation tasks as well. For instance, if we first determine where the most salient phrases (e.g., noun phrases, adjectives, etc.) are located in the input text, then SEA-GUIDE can be used to track their mentions during the inference the same way it does in data-to-text NLG. Some of the biggest advantages of SEA-GUIDE over other methods for semantic control are that: (1) it is domain- and model-independent (for encoder-decoder architectures), (2) it requires no model modifications, additional training data or manual annotation, and (3) it adds only a relatively small performance overhead over the standard beam search decoding, which it, however, dramatically outperforms in semantic accuracy and slightly in the standard metrics too.

10.1.3 Evaluation Metrics and Diversity

As we saw in our semantic accuracy evaluation, using the slot aligner or SEA-GUIDE to reduce SER in model outputs had no negative impact on the automatic metric scores (BLEU, BERTScore, etc.). However, we observed an ever-present trade-off between the automatic metrics and the *diversity* of utterances

generated by an NLG model when using standard inference methods. For example, beam search typically improves the metric scores and SER over greedy search outputs, but the diversity drops. With nucleus sampling, on the other hand, the diversity shoots up dramatically, but at the expense of the automatic metrics, which fall far below the greedy search levels. SER also tends to increase in this case, unless semantic reranking is used.

It is important to keep in mind that SER is only effective at detecting fact omissions and incorrect slot mentions. Through manual evaluation of model outputs, we determined that hallucinations, incoherence and disfluencies are still present in utterances with zero SER and low automatic metric scores. The automatic metrics are reference-based, however, so maximizing their scores merely corresponds to maximizing the lexical similarity of a generated utterance to a small set of references. Our conclusion therefore is that neither automatic metrics (especially BLEU) nor SER by itself gives an accurate idea of a generated utterance's quality. We tried adding language model perplexity (calculated by GPT-2) to the mix, but this metric turned out to be even more inconsistent, scoring human-written references significantly worse than model outputs with errors. The perplexity of a language model evidently does not directly correspond to the evaluated text's fluency, per se, as is commonly assumed. It is clear that none of the currently available automatic metrics can replace human evaluation in accuracy and reliability, yet they are the only practical way to at least estimate the general utterance quality trend among different model or inference method configurations in experiments. Diversity metrics, although not showing anything about the overall quality of model outputs, are at least consistent in indicating the language diversity across a set of utterances, since they are neither reference-based not language model-dependent. All throughout our

evaluations, we considered the vocabulary size as a representative of the diversity metrics, since most of them were very highly correlated with the vocabulary size.

Considering the above findings, we tried an approach that promotes utterance diversity, while keeping SER as low as possible and maintaining reference-based metric scores in reasonable ranges. We developed Batch-MCTS (Chapter 7), an inference method based on Monte-Carlo Tree Search (MCTS), which utilizes sampling, to increase diversity, along with informed tree search that can optimize for an arbitrary metric (e.g., fluency or semantic accuracy, or their combination). The MCTS algorithm is guided by a referenceless utterance evaluation metric, which we designed and verified the stand-alone performance of in a thorough analysis in Chapter 8. We used LM perplexity for its fluency component, and experimented with different standard metrics comparing utterance candidates with pseudo-references (automatically built from MRs) for the semantic accuracy component. Among the different variants of the referenceless semantic accuracy component we tried, we found the neural metrics BERTScore and BLEURT to be the most robust. Besides these two, we also experimented with SER, which is a referenceless metric by nature. Generally speaking, this metric composed of LM perplexity and BERTScore/BLEURT can readily be used for a referenceless utterance evaluation outside of our Batch-MCTS inference use case.

Our extensive evaluation of Batch-MCTS guided by the different variants of the referenceless metric in Chapter 9 revealed several strengths of this novel method over standard inference methods. We note that there is no single configuration of Batch-MCTS that would beat all the baselines (i.e., achieve diversity of nucleus sampling with $p = 0.8$, while maintaining automatic metric scores on par with beam search outputs). Nevertheless, with different configurations, Batch-MCTS can do

overall significantly better than any configuration of any standard inference method. We showed that, on the ViGGO dataset, Batch-MCTS guided by BERTScore outperforms diverse beam search in terms of diversity and semantic accuracy at the same time, while also offering superior performance in automatic metrics too. It can also match the diversity of nucleus sampling with $p = 0.3$, while significantly increasing semantic accuracy, as well as most of the automatic metric scores. Other configurations then approach beam search in automatic metric scores and exceed it in diversity and semantic accuracy. With SER instead of BERTScore for semantic guidance, Batch-MCTS became even more robust and easily exceeded the performance of the standard methods even when equipped with semantic reranking. To achieve the above performance of Batch-MCTS, we had to replace GPT-2 perplexity with the NLG model's own perplexity as the fluency component – for similar reasons to those that make it a poor evaluation metric of generated utterances in our task, discussed in an earlier paragraph.

In a human evaluation, we found that, when Batch-MCTS outputs match the high diversity of nucleus sampling with $p = 0.8$, the generated utterances contain significantly fewer actual errors overall (across 8 different categories of semantic and syntactic errors), despite the Batch-MCTS outputs scoring 3 points lower in BLEU. This further stresses the importance of human evaluation, especially when the inference employs sampling-based methods, resulting in diverse outputs that are penalized by most standard reference-based evaluation metrics.

On the E2E dataset, using GPT-2 perplexity is actually effective, guiding the MCTS to utterances with up to 20% larger vocabulary than even nucleus sampling with $p = 0.8$, yet reducing SER from 4.36% to the 2.42–3.19% range. With a different configuration, SER drops to 1.16–1.6%, i.e., far below that of diverse beam

search, with 15% higher diversity nonetheless and substantially higher automatic metric scores.

All in all, our experiments showed that Batch-MCTS is a promising method for inference in NLG. It successfully optimizes for a chosen metric, and generates diverse utterances with a higher semantic accuracy than standard inference methods do. It is flexible and highly configurable, and leverages parallel processing, thanks to which it is feasible in real time.

10.1.4 Limitations of Our Work

Although we demonstrated the strengths of the methods we proposed and evaluated in this thesis, here we summarize some of the most important limitations of our work that should be considered before using these methods:

- Our heuristic slot aligner has a limited scalability. As pointed out multiple times throughout the thesis, it is domain-transferable, but it can involve a significant amount of manual work and domain knowledge if the new domain has a large ontology. Adapting the slot aligner to a new domain requires categorizing all possible slots into the 5 classes the slot aligner recognizes, and populating its semantic dictionary with synonyms and paraphrases for possible values, where applicable. While this is feasible for relatively small domains, it would become rather impractical if we tried to do it for a domain with a thousand different slots.

- The SEA-GUIDE decoding method is only compatible with encoder-decoder, and ideally transformer-based, model architectures due to its dependence on the cross-attention mechanism. Furthermore, as discussed in Section 6.4.2,

SEA-GUIDE's effectiveness on Boolean slots is limited, and its design does not allow it to detect hallucinations (which is the case with the slot aligner too). Finally, we observed that sampling during decoding confuses the tracking ability of SEA-GUIDE, which is the main reason for us not having experimented with it in Batch-MCTS.

- The running time of the Batch-MCTS inference increases approximately linearly with the number of iterations, which cannot be parallelized, yet the main benefit of the MCTS algorithm lies in more iterations, as it is at the end of one iteration and the beginning of the next that candidate utterances get evaluated (and tree node rewards get calculated) and MCTS applies the tree policy to decide what parts of the search tree to explore next.

- Speaking of runtime performance, our Batch-MCTS experiments were limited to the smallest pretrained language models, i.e., T5-small for the NLG model, and the smallest GPT-2 variant for the fluency component of the referenceless metric. In our preliminary experiments with GPT-2-medium, however, we saw a definite improvement in Batch-MCTS performance, and the gains could further stack when using Batch-MCTS with a larger and more capable NLG model. For example, as we showed in Section 6.3.2, T5-base is massively more semantically accurate than T5-small without reranking, which would presumably translate to overall better utterance candidates sampled in Batch-MCTS playouts, leaving more room for emphasizing utterance diversity.

- When comparing model outputs in terms of language diversity, we evaluated diversity metrics on corpus level, i.e., on all the model outputs for the entire test set. This may, however, obscure variation, as well as hide repetitiveness,

in word choice and sentence structure within clusters of related inputs, such as inputs of the same DA type. Calculating per-DA diversity across the test set may thus offer a more accurate picture of the language variation in the model outputs. Alternatively, the average diversity within the pool of top K candidates generated by an inference method, across the whole test set, could be another more accurate way of assessing the inference method's diversity-promoting properties.

- All of our experiments were performed on data in English, nevertheless we expect our methods to work in principle with other languages as well. In Batch-MCTS, however, we rely on pretrained language models for measuring text fluency, as well as semantic accuracy in the form of the neural metrics BERTScore and BLEURT. These language models are typically pretrained on English data only, but some of them have multilingual versions, such as M-BERT (Pires et al., 2019), mBART (Liu et al., 2020b) and mT5 (Xue et al., 2021), which could be used for automatic evaluation of utterances in up to 100 other languages. Yet for many low-resource languages, we may not be able to take advantage of a large pretrained model and would have to resort to using standard N-gram overlap metrics instead, which, as we showed, are not as effective as neural metrics. With a few modifications, our slot aligner would also work with other languages and would thus remain a valid alternative for the semantic accuracy component.

- With the rapid progress in deep learning and a wide adoption of large language models, dialogue systems are gradually shifting closer to end-to-end response generation, i.e., bypassing a discrete dialogue manager component.

NLG in these systems is performed directly from the conversation context and an external knowledge base as inputs, delegating the decision making about the content selection and the form of the response entirely to the NLG model itself (Dinan et al., 2018; Gopalakrishnan et al., 2019; Zhao et al., 2020; Li et al., 2022b). These approaches are more scalable than MR-to-text generation, but lack controllability, and the creation of training data for this type of NLG task is very expensive. Our methods for enforcing semantic accuracy are largely incompatible with this approach to NLG, however, we would expect the diversity-promoting Batch-MCTS inference method to be equally effective as in MR-to-text NLG.

- Finally, there is the set of challenges faced in the evaluation of NLG model outputs. Automatic metrics are always a limiting factor due to their usual dependence on human-written utterances, but human evaluation is typically only possible on a small scale. Even then, humans may end up doing an unsatisfactory job, if they are not experts in the domain and/or the task. Hence, we performed the majority of the analyses in this thesis using a variety of automatic metrics, which only reflect the overall quality of utterances approximately, and we only turned to human evaluation in a few very specific cases, typically involving model outputs identified as the best by the automatic metrics. We avoided crowdsourced evaluation, preferring instead two or three expert annotators familiar with the task/domain.

10.2 Future Work

We showed that the Batch-MCTS method we proposed and evaluated across Chapters 7–9 already outperforms individual standard inference methods, such as beam search or nucleus sampling. In this section, we outline several ideas that may further increase its gains in performance.

10.2.1 Modifications to Batch-MCTS

Batch-MCTS is a highly configurable inference method, with a number of different parameters and modules that can be modified. Below, we describe some of the ideas that we did not have an opportunity to implement and test, grouped by different aspects of Batch-MCTS.

MCTS parameters. The MCTS algorithm has multiple parameters that modify the way the tree search is conducted, such as the exploration coefficient or the tree policy. In addition to these standard parameters, Batch-MCTS can use different node selection methods (e.g., only selecting leaf nodes to start playouts from), or alternative node reward aggregation methods (e.g., keeping the maximum score in a subtree instead of averaging). In our preliminary experiments with some of these, we did not observe any significant differences compared to the default settings in terms of the best candidate . Nevertheless, this could change with the batch size, the increasing number of iterations, as well as the model size and the state evaluation metric, so performing a more comprehensive hyperparameter tuning could possibly reveal parameter settings with which Batch-MCTS performs better than with the default ones we used in the experiments in this thesis.

State evaluation metric. When combining the fluency and semantic accuracy scores for the MCTS state evaluation metric, we use the reciprocal of perplexity so as to be maximizing the score and to convert it to the $[0, 1]$ range. However, this results in only a very small part of the $[0, 1]$ range being used. Ideally, the perplexity metric would be normalized before being combined with other scores, such as BERTScore or BLEURT, both of which use most of the $[0, 1]$ range. One possible way to normalize perplexity scores would be by calculating the minimum and maximum values across utterances generated for the training set inputs using nucleus sampling. Normalized scores would allow for the components of the state evaluation metric to be weighted in order to, for example, put more importance on semantic accuracy in the candidate utterances.

Sampling in playouts. The sampling employed during playouts is the main contributing factor to the candidate utterance diversity in Batch-MCTS. In all of our experiments, we used nucleus sampling with a fixed p-value of 0.8. While this does a good job at promoting diversity, we noticed that this value is high enough to cause the decoder to be considering hundreds of tokens at some time steps, among which many may be a poor choice leading to incoherence. There are at least a few methods we could try in order to avoid this situation, while preserving most of the sampling-induced diversity: (1) by combining nucleus sampling with top-K sampling to reduce the chance of introducing incoherence when the token probability distribution has a heavy tail; (2) by reducing the p-value of the nucleus sampling and slightly increasing the sampling temperature, so as to ensure the lower-probability tokens among those within the p probability mass get sampled more frequently; (3) by decaying the p-value or the sampling temperature over the length of the sequence

or, alternatively, over the MCTS iterations.

SEA-GUIDE for semantic guidance. Although we mentioned that sampling negatively affects the ability of SEA-GUIDE to correctly track slot mentions in the generated utterance, the drop in performance happens between $p = 0.5$ and 0.8. This suggest that it could still work reliably as the semantic accuracy component in the MCTS state evaluation metric, as long as the p-value of the nucleus sampling is set to a slightly lower value. SEA-GUIDE showed the biggest gains for the T5-small and BART-base models, so it could be effective as part of Batch-MCTS particularly for these smaller models.

Diversity. On the ViGGO dataset, we had to introduce the NLG model's own perplexity into the state evaluation metric in order for Batch-MCTS to start performing better. The consequence of this was, however, a drastic reduction in utterance diversity, as the model's perplexity started guiding the MCTS more toward the most probable utterances. We hypothesize that the diversity could be better preserved if, instead of the whole-utterance perplexity, we focused on a smaller contiguous window of tokens with the lowest probability in the utterance. This is motivated by the fact that interesting utterances often contain low-probability tokens, but they are typically followed by relatively high-probability tokens. On the other hand, when an utterance contains an incoherence, the token probabilities remain low across a longer subsequence of tokens. The lowest probability of a contiguous subsequence of a fixed number of tokens (e.g., 10) could thus be a useful component for the metric, penalizing candidate utterances with a low-probability segment (likely an incoherence), as opposed to penalizing utterances that are less probable because of

a few uncommon words. Other ways of encouraging greater diversity among the candidate utterances could involve adding a diversity metric as a component in the state evaluation metric, or enforcing diversity within the pool of best candidates Batch-MCTS maintains between iterations.

10.2.2 Reinforcement Learning With Batch-MCTS

Inspired by the success of DeepMind's AlphaGo (Silver et al., 2016, 2017), AlphaZero (Silver et al., 2018) and MuZero (Schrittwieser et al., 2020) systems that consistently achieve superhuman performance at Go, chess and visually complex Atari games, we think the combination of deep reinforcement learning (Arulkumaran et al., 2017; Sutton and Barto, 2018) and MCTS may be an interesting avenue to explore even in NLG. Reinforcement learning (RL) is a learning strategy for sequential decision-making problems in which the reward is not known until an end state is reached. After all, in data-to-text NLG (or any NLG task, for that matter) the model makes a sequence of decisions about which token to use next when generating an utterance, while at the same time, it is impossible to tell from a partial utterance whether the final utterance is going to be good or not. The objective of an RL model is to learn to predict which action (token, in our case) in the given state would ultimately lead to the best outcome (i.e., utterance), even if it means taking an action with a negative immediate reward (such as generating a token with a lower conditional probability).

Here we outline two possible ways of using MCTS to train an RL-based NLG model on top of a standard trained encoder-decoder. Each of them uses the concept of a *value network*, which in conventional RL is a model that predicts the reward that could be accumulated from a given state until an end state is reached.

In other words, it predicts the future reward for the current state, which in case of NLG corresponds to the score of a whole utterance.

In the first approach, we propose training a value network $f_\theta(x, \hat{y}_{<t})$ to produce a single scalar value – the predicted whole-utterance score – given an input x and a prefix $\hat{y}_{<t}$ (i.e, a partial utterance), similar to He et al. (2017). To this end, random prefixes are first generated from the inputs in the training set using a trained NLG model. Subsequently, Batch-MCTS with heavy playouts, as described in 7.2.2, is performed from the state corresponding to each of the prefixes. The score of the best candidate the tree search finds then becomes the value v of the value network:

$$v = f_\theta(x, \hat{y}_{<t}) = \underset{y' \in \mathcal{Y}:\, y'_{<t} = \hat{y}_{<t}}{\operatorname{argmax}} \operatorname{score}\big(y', y(x)\big)\, p(y'|x) \;,$$

where $\mathcal{Y}$ is the space of whole utterances, $\operatorname{score}\big(y', y(x)\big)$ is the generated utterance's score on a $[0, 1]$ scale given the reference utterance $y(x)$ for input x, and $p(y'|x)$ is the conditional probability calculated by the NLG model. In case of a referenceless metric, the term $y(x)$ would be replaced by just the input x. Finally, once the value network is trained, it can be used during inference to replace the decoder's conditional probabilities when making token predictions, or in a linear combination with them.

The second approach, inspired by the AlphaGo Zero system (Silver et al., 2017), differs from the first one in that the value network is trained to produce, along with the future reward prediction v, a vector $\mathbf{p}$ of action probabilities $p_a = P(a|s)$ in the given state s, i.e., $(\mathbf{p}, v) = f_\theta^{\mathrm{A}}(x, \hat{y}_{<t})$. The training process for this value network involves generating an utterance from the input x token by token, while at each time step t, Batch-MCTS would serve as the policy determining the probability

distribution $\boldsymbol{\pi}_t$ of the next token in the current context. Once an end state is reached, the generated utterance y' is evaluated. Then the score $z = \text{score}\big(y', y(x)\big)$ and the search probability distributions $\boldsymbol{\pi}$ are used to update the parameters θ of the value network through gradient descent so as to maximize the similarity of $(\mathbf{p}, v)$ to $(\boldsymbol{\pi}, z)$. Such a trained value network would be used for inference directly, replacing the original NLG model entirely. Since the value network outputs a probability distribution for the next token, the distribution can be sampled from in order to achieve additional diversity.

Bibliography

Shubham Agarwal and Marc Dymetman. 2017. A surprisingly effective out-of-the-box char2char model on the e2e nlg challenge dataset. In *Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue*, pages 158–163.

Shubham Agarwal, Marc Dymetman, and Eric Gaussier. 2018. Char2char generation with reranking for the e2e nlg challenge. In *Proceedings of the 11th International Conference on Natural Language Generation*, pages 451–456.

Kai Arulkumaran, Marc Peter Deisenroth, Miles Brundage, and Anil Anthony Bharath. 2017. Deep reinforcement learning: A brief survey. *IEEE Signal Processing Magazine*, 34(6):26–38.

Dzmitry Bahdanau, Kyung Hyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015*.

Ashutosh Baheti, Alan Ritter, Jiwei Li, and William B Dolan. 2018. Generating more interesting responses in neural conversation models with distributional constraints. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3970–3980.

Anusha Balakrishnan, Jinfeng Rao, Kartikeya Upasani, Michael White, and Rajen

Subba. 2019. Constrained decoding for neural nlg from compositional representations in task-oriented dialogue. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 831–844.

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2013. Abstract meaning representation for sembanking. In *Proceedings of the 7th linguistic annotation workshop and interoperability with discourse*, pages 178–186.

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

Kevin K Bowden, Jiaqi Wu, Wen Cui, Juraj Juraska, Vrindavan Harrison, Brian Schwarzmann, Nick Santer, and Marilyn Walker. 2018. Slugbot: Developing a computational model and framework of a novel dialogue genre. In *Alexa Prize SocialBot Grand Challenge 2 Proceedings*.

Peter F Brown, Vincent J Della Pietra, Stephen A Della Pietra, and Robert L Mercer. 1993. The mathematics of statistical machine translation: Parameter estimation. *Computational linguistics*, 19(2):263–311.

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

Cameron B Browne, Edward Powley, Daniel Whitehouse, Simon M Lucas, Peter I

Cowling, Philipp Rohlfshagen, Stephen Tavener, Diego Perez, Spyridon Samothrakis, and Simon Colton. 2012. A survey of monte carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in games*, 4(1):1–43.

Paweł Budzianowski, Tsung-Hsien Wen, Bo-Hsiang Tseng, Iñigo Casanueva, Stefan Ultes, Osman Ramadan, and Milica Gasic. 2018. 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*, pages 5016–5026.

Michael Buhrmester, Tracy Kwang, and Samuel D Gosling. 2011. Amazon's mechanical turk: A new source of inexpensive, yet high-quality, data? *Perspectives on psychological science*, 6(1):3–5.

Ozan Caglayan, Pranava Swaroop Madhyastha, and Lucia Specia. 2020. Curious case of language generation evaluation metrics: A cautionary tale. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 2322–2328.

Chris Callison-Burch, Miles Osborne, and Philipp Koehn. 2006. Re-evaluating the role of bleu in machine translation research. In *11th conference of the european chapter of the association for computational linguistics*, pages 249–256.

David L Chen and Raymond J Mooney. 2008. Learning to sportscast: a test of grounded language acquisition. In *Proceedings of the 25th international conference on Machine learning*, pages 128–135.

Mia Xu Chen, Orhan Firat, Ankur Bapna, Melvin Johnson, Wolfgang Macherey, George Foster, Llion Jones, Mike Schuster, Noam Shazeer, Niki Parmar, et al.

2018. The best of both worlds: Combining recent advances in neural machine translation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 76–86.

Wenhu Chen, Jianshu Chen, Yu Su, Zhiyu Chen, and William Yang Wang. 2020. Logical natural language generation from open-domain tables. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7929–7942.

Andrew Chisholm, Will Radford, and Ben Hachey. 2017. Learning to generate one-sentence biographies from wikidata. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 633–642.

Kyunghyun Cho, Bart van Merriënboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. In *EMNLP*.

Hyung Won Chung, Thibault Fevry, Henry Tsai, Melvin Johnson, and Sebastian Ruder. 2020. Rethinking embedding coupling in pre-trained language models. In *International Conference on Learning Representations*.

Rémi Coulom. 2006. Efficient selectivity and backup operators in monte-carlo tree search. In *International conference on computers and games*, pages 72–83. Springer.

Heriberto Cuayáhuitl, Nina Dethlefs, Helen Hastie, and Xingkun Liu. 2014. Training a statistical surface realiser from automatic slot labelling. In *Spoken Language Technology Workshop (SLT), 2014 IEEE*, pages 112–117. IEEE.

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime G Carbonell, Quoc Le, and Ruslan Salakhutdinov. 2019. Transformer-xl: Attentive language models beyond a fixed-length context. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2978–2988.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186.

Emily Dinan, Stephen Roller, Kurt Shuster, Angela Fan, Michael Auli, and Jason Weston. 2018. Wizard of wikipedia: Knowledge-powered conversational agents. In *International Conference on Learning Representations*.

Ondřej Dušek, David M Howcroft, and Verena Rieser. 2019. Semantic noise matters for neural natural language generation. In *Proceedings of the 12th International Conference on Natural Language Generation*, pages 421–426.

Ondřej Dušek and Filip Jurčíček. 2016. 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 (Volume 2: Short Papers)*, pages 45–51.

Ondřej Dušek, Jekaterina Novikova, and Verena Rieser. 2017. Referenceless quality estimation for natural language generation. In *1st Workshop on Learning to Generate Natural Language*. ICML.

Ondřej Dušek, Jekaterina Novikova, and Verena Rieser. 2018. Findings of the E2E

NLG challenge. In *Proceedings of the 11th International Conference on Natural Language Generation*, pages 322–328, Tilburg University, The Netherlands. Association for Computational Linguistics.

Ondřej Dušek, Jekaterina Novikova, and Verena Rieser. 2020. Evaluating the state-of-the-art of end-to-end natural language generation: The e2e nlg challenge. *Computer Speech & Language*, 59:123–156.

Chris Dyer, Victor Chahuneau, and Noah A Smith. 2013. A simple, fast, and effective reparameterization of ibm model 2. *NAACL HLT 2013*, pages 644–648.

Mihail Eric, Rahul Goel, Shachi Paul, Abhishek Sethi, Sanchit Agarwal, Shuyang Gao, Adarsh Kumar, Anuj Goyal, Peter Ku, and Dilek Hakkani-Tur. 2020. Multiwoz 2.1: A consolidated multi-domain dialogue dataset with state corrections and state tracking baselines. In *Proceedings of The 12th Language Resources and Evaluation Conference*, pages 422–428.

Angela Fan, Mike Lewis, and Yann Dauphin. 2018. Hierarchical neural story generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 889–898.

Maryam Fazel-Zarandi, Shang-Wen Li, Jin Cao, Jared Casale, Peter Henderson, David Whitney, and Alborz Geramifard. 2017. Learning robust dialog policies in noisy environments. *NIPS 2017 Workshop on Conversational AI.*

Thiago Castro Ferreira, Claire Gardent, Nikolai Ilinykh, Chris van der Lee, Simon Mille, Diego Moussallem, and Anastasia Shimorina. 2020. The 2020 bilingual, bi-directional webnlg+ shared task: Overview and evaluation results (webnlg+

2020). In *Proceedings of the 3rd International Workshop on Natural Language Generation from the Semantic Web (WebNLG+)*, pages 55–76.

Jessica Ficler and Yoav Goldberg. 2017. Controlling linguistic style aspects in neural language generation. In *Proceedings of the Workshop on Stylistic Variation*, pages 94–104.

Markus Freitag, George Foster, David Grangier, Viresh Ratnakar, Qijun Tan, and Wolfgang Macherey. 2021. Experts, errors, and context: A large-scale study of human evaluation for machine translation. *Transactions of the Association for Computational Linguistics*, 9:1460–1474.

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

Milica Gašić, Simon Keizer, Francois Mairesse, Jost Schatzmann, Blaise Thomson, Kai Yu, and Steve Young. 2008. Training and evaluation of the HIS POMDP dialogue system in noise. In *Proceedings of the 9th SIGDIAL Workshop on Discourse and Dialogue*, pages 112–119. Association for Computational Linguistics.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. 2017. Convolutional sequence to sequence learning. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1243–1252.

Douwe Gelling and Trevor Cohn. 2014. Simple extensions and POS tags for a reparameterised IBM model 2. In *Proceedings of the 52nd Annual Meeting of*

the Association for Computational Linguistics (Volume 2: Short Papers), pages 150–154, Baltimore, Maryland. Association for Computational Linguistics.

Marjan Ghazvininejad, Xing Shi, Jay Priyadarshi, and Kevin Knight. 2017. Hafez: an interactive poetry generation system. In *Proceedings of ACL 2017, System Demonstrations*, pages 43–48.

Karthik Gopalakrishnan, Behnam Hedayatnia, Qinlang Chen, Anna Gottardi, Sanjeev Kwatra, Anu Venkatesh, Raefer Gabriel, and Dilek Hakkani-Tür. 2019. Topical-chat: Towards knowledge-grounded open-domain conversations. *Proc. Interspeech 2019*, pages 1891–1895.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor OK Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1631–1640.

Hamza Harkous, Isabel Groves, and Amir Saffari. 2020. 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*, pages 2410–2424.

Vrindavan Harrison, Juraj Juraska, Wen Cui, Lena Reed, Kevin K Bowden, Jiaqi Wu, Brian Schwarzmann, Abteen Ebrahimi, Rishi Rajasekaran, Nikhil Varghese, et al. 2019. Athena: Constructing dialogues dynamically with discourse constraints. In *Alexa Prize SocialBot Grand Challenge 3 Proceedings*.

Di He, Hanqing Lu, Yingce Xia, Tao Qin, Liwei Wang, and Tie-Yan Liu. 2017.

Decoding with value networks for neural machine translation. In *Advances in Neural Information Processing Systems*, pages 178–187.

Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2020. Deberta: Decoding-enhanced bert with disentangled attention. In *International Conference on Learning Representations*.

Matthew Henderson, Blaise Thomson, and Steve Young. 2014. Robust dialog state tracking using delexicalised recurrent neural networks and unsupervised adaptation. In *Spoken Language Technology Workshop (SLT), 2014 IEEE*, pages 360–365. IEEE.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. 2019. The curious case of neural text degeneration. In *International Conference on Learning Representations*.

Ari Holtzman, Jan Buys, Maxwell Forbes, Antoine Bosselut, David Golub, and Yejin Choi. 2018. Learning to write with cooperative discriminators. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1638–1649.

David Howcroft, Crystal Nakatsu, and Michael White. 2013. Enhancing the expression of contrast in the SPaRKy restaurant corpus. In *Proceedings of the 14th European Workshop on Natural Language Generation*, pages 30–39.

David M Howcroft, Anja Belz, Miruna-Adriana Clinciu, Dimitra Gkatzia, Sadid A Hasan, Saad Mahamood, Simon Mille, Emiel Van Miltenburg, Sashank San-

thanam, and Verena Rieser. 2020. 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*, pages 169–182.

Shui Hu, Yang Liu, Anna Gottardi, Behnam Hedayatnia, Anju Khatri, Anjali Chadha, Qinlang Chen, Pankaj Rajan, Ali Binici, Varun Somani, Yao Lu, Prerna Dwivedi, Lucy Hu, Hangjie Shi, Sattvik Sahai, Mihail Eric, Karthik Gopalakrishnan, Seokhwan Kim, Spandana Gella, Alexandros Papangelis, Patrick Lange, Di Jin, Nicole Chartier, Mahdi Namazifar, Aishwarya Padmakumar, Sarik Ghazarian, Shereen Oraby, Anjali Narayan-Chen, Yuheng Du, Lauren Stubell, Savanna Stiff, Kate Bland, Arindam Mandal, Reza Ghanadan, and Dilek Hakkani-Tur. 2021. Further advances in open domain dialog systems in the fourth alexa prize socialbot grand challenge. In *Alexa Prize SocialBot Grand Challenge 4 Proceedings*.

Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Yejin Bang, Andrea Madotto, and Pascale Fung. 2022. Survey of hallucination in natural language generation. *arXiv preprint arXiv:2202.03629*.

Juraj Juraska, Kevin Bowden, Lena Reed, Vrindavan Harrison, Wen Cui, Omkar Patil, Rishi Rajasekaran, Angela Ramirez, Cecilia Li, Eduardo Zamora, Phillip Lee, Jeshwanth Bheemanpally, Rohan Pandey, Adwait Ratnaparkhi, and Marilyn Walker. 2021. Athena 2.0: Contextualized dialogue management for an Alexa Prize SocialBot. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 124–133.

Juraj Juraska, Kevin K Bowden, and Marilyn Walker. 2019. ViGGO: A video game

corpus for data-to-text generation in open-domain conversation. In *Proceedings of the 12th International Conference on Natural Language Generation*.

Juraj Juraska, Panagiotis Karagiannis, Kevin Bowden, and Marilyn Walker. 2018. A deep ensemble model with slot alignment for sequence-to-sequence natural language generation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 152–162.

Juraj Juraska and Marilyn Walker. 2018. Characterizing variation in crowd-sourced data for training neural language generators to produce stylistically varied outputs. In *Proceedings of the 11th International Conference on Natural Language Generation*, pages 441–450.

Juraj Juraska and Marilyn Walker. 2021. Attention is indeed all you need: Semantically attention-guided decoding for data-to-text nlg. In *Proceedings of the 14th International Conference on Natural Language Generation*, pages 416–431.

Jad Kabbara and Jackie Chi Kit Cheung. 2016. Stylistic transfer in natural language generation systems using recurrent neural networks. In *Proceedings of the Workshop on Uphill Battles in Language Processing: Scaling Early Achievements to Robust Methods*, pages 43–47.

Mihir Kale and Abhinav Rastogi. 2020. Text-to-text pre-training for data-to-text tasks. In *Proceedings of the 13th International Conference on Natural Language Generation*, pages 97–102.

Katharina Kann, Sascha Rothe, and Katja Filippova. 2018. Sentence-level fluency

evaluation: References help, but can be spared! In *Proceedings of the 22nd Conference on Computational Natural Language Learning*, pages 313–323.

Chris Kedzie and Kathleen McKeown. 2019. 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*, pages 584–593.

Chris Kedzie and Kathleen McKeown. 2020. Controllable meaning representation to text generation: Linearization and data augmentation strategies. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5160–5185.

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

Chandra Khatri, Behnam Hedayatnia, Anu Venkatesh, Jeff Nunn, Yi Pan, Qing Liu, Han Song, Anna Gottardi, Sanjeev Kwatra, Sanju Pancholi, et al. 2018. Advancing the state of the art in open domain dialog systems through the alexa prize. In *2018 Alexa Prize Proceedings*.

Nikita Kitaev, Lukasz Kaiser, and Anselm Levskaya. 2020. Reformer: The efficient transformer. In *International Conference on Learning Representations*.

Levente Kocsis and Csaba Szepesvári. 2006. Bandit based monte-carlo planning. In *European conference on machine learning*, pages 282–293. Springer.

Ioannis Konstas and Mirella Lapata. 2013. A global model for concept-to-text generation. *J. Artif. Intell. Res.(JAIR)*, 48:305–346.

Gerasimos Lampouras and Andreas Vlachos. 2016. Imitation learning for language generation from unaligned data. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 1101–1112.

Irene Langkilde and Kevin Knight. 1998. Generation that exploits corpus-based statistical knowledge. In *COLING 1998 Volume 1: The 17th International Conference on Computational Linguistics*.

Jey Han Lau, Alexander Clark, and Shalom Lappin. 2017. Grammaticality, acceptability, and probability: A probabilistic view of linguistic knowledge. *Cognitive science*, 41(5):1202–1241.

Hugo Laurençon, Lucile Saulnier, Thomas Wang, Christopher Akiki, Albert Villanova del Moral, Teven Le Scao, Leandro Von Werra, Chenghao Mou, Eduardo González Ponferrada, Huu Nguyen, et al. 2022. The bigscience corpus a 1.6 tb composite multilingual dataset.

Rémi Lebret, David Grangier, and Michael Auli. 2016. Neural text generation from structured data with application to the biography domain. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1203–1213.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

Chris van der Lee, Albert Gatt, Emiel van Miltenburg, Sander Wubben, and Emiel Krahmer. 2019. Best practices for the human evaluation of automatically gener-

ated text. In *Proceedings of the 12th International Conference on Natural Language Generation (INLG'19)*, Tokyo, Japan. Association for Computational Linguistics.

Leo Leppänen, Myriam Munezero, Mark Granroth-Wilding, and Hannu Toivonen. 2017. Data-driven news generation for automated journalism. In *Proceedings of the 10th international conference on natural language generation*, pages 188–197.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and William B Dolan. 2016a. A diversity-promoting objective function for neural conversation models. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 110–119.

Jiwei Li, Will Monroe, and Dan Jurafsky. 2016b. A simple, fast diverse decoding algorithm for neural generation. *arXiv preprint arXiv:1611.08562*.

Jiwei Li, Will Monroe, and Dan Jurafsky. 2017a. Learning to decode for future success. *arXiv preprint arXiv:1701.06549*.

Juncen Li, Robin Jia, He He, and Percy Liang. 2018. Delete, retrieve, generate: A simple approach to sentiment and style transfer. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Wei Li, Wenhao Wu, Moye Chen, Jiachen Liu, Xinyan Xiao, and Hua Wu. 2022a. Faithfulness in natural language generation: A systematic survey of analysis, evaluation and optimization methods. *arXiv preprint arXiv:2203.05227*.

Xiujun Li, Yun-Nung Chen, Lihong Li, Jianfeng Gao, and Asli Celikyilmaz. 2017b. End-to-end task-completion neural dialogue systems. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 733–743.

Yu Li, Baolin Peng, Yelong Shen, Yi Mao, Lars Liden, Zhou Yu, and Jianfeng Gao. 2022b. Knowledge-grounded dialogue generation with a unified knowledge representation. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 206–218, Seattle, United States. Association for Computational Linguistics.

Percy Liang, Michael I Jordan, and Dan Klein. 2009. Learning semantic correspondences with less supervision. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 91–99.

Opher Lieber, Or Sharir, Barak Lenz, and Yoav Shoham. 2021. Jurassic-1: Technical details and evaluation. *White Paper. AI21 Labs*.

Chin-Yew Lin and Eduard Hovy. 2003. Automatic evaluation of summaries using n-gram co-occurrence statistics. In *Proceedings of the 2003 human language technology conference of the North American chapter of the association for computational linguistics*, pages 150–157.

Chin-Yew Lin and Franz Josef Och. 2004. Automatic evaluation of machine trans-

lation quality using longest common subsequence and skip-bigram statistics. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*, page 605. Association for Computational Linguistics.

Chia-Wei Liu, Ryan Joseph Lowe, Iulian Serban, Michael Noseworthy, Laurent Charlin, and Joelle Pineau. 2016. How not to evaluate your dialogue system: An empirical study of unsupervised evaluation metrics for dialogue response generation. In *EMNLP*.

Qi Liu, Matt J Kusner, and Phil Blunsom. 2020a. A survey on contextual embeddings. *arXiv preprint arXiv:2003.07278*.

Yinhan Liu, Jiatao Gu, Naman Goyal, Xian Li, Sergey Edunov, Marjan Ghazvininejad, Mike Lewis, and Luke Zettlemoyer. 2020b. Multilingual denoising pre-training for neural machine translation. *Transactions of the Association for Computational Linguistics*, 8:726–742.

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

François Mairesse, Milica Gašić, Filip Jurčíček, Simon Keizer, Blaise Thomson, Kai Yu, and Steve Young. 2010. Phrase-based statistical language generation using graphical models and active learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1552–1561.

François Mairesse and Marilyn Walker. 2007. Personage: Personality generation for dialogue. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 496–503.

François Mairesse and Steve Young. 2014. Stochastic language generation in dialogue using factored language models. *Computational Linguistics*, 40:763–799.

Joshua Maynez, Shashi Narayan, Bernd Bohnet, and Ryan McDonald. 2020. On faithfulness and factuality in abstractive summarization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1906–1919.

Hongyuan Mei, Mohit Bansal, and Matthew R Walter. 2016. What to talk about and how? selective generation using lstms with coarse-to-fine alignment. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 720–730.

Haitao Mi, Baskaran Sankaran, Zhiguo Wang, and Abe Ittycheriah. 2016. Coverage embedding models for neural machine translation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 955–960.

Nikita Moghe, Siddhartha Arora, Suman Banerjee, and Mitesh M Khapra. 2018. Towards exploiting background knowledge for building conversation systems. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2322–2332.

Soichiro Murakami, Akihiko Watanabe, Akira Miyazawa, Keiichi Goshima, Toshihiko Yanase, Hiroya Takamura, and Yusuke Miyao. 2017. Learning to generate market comments from stock prices. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1374–1384.

Linyong Nan, Dragomir Radev, Rui Zhang, Amrit Rau, Abhinand Sivaprasad, Chiachun Hsieh, Xiangru Tang, Aadit Vyas, Neha Verma, Pranav Krishna, et al. 2021. 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*, pages 432–447.

Neha Nayak, Dilek Hakkani-Tür, Marilyn Walker, and Larry Heck. 2017. To plan or not to plan? discourse planning in slot-value informed sequence to sequence models for language generation. *Proc. Interspeech 2017*, pages 3339–3343.

Feng Nie, Jin-Ge Yao, Jinpeng Wang, Rong Pan, and Chin-Yew Lin. 2019. A simple recipe towards reducing hallucination in neural surface realisation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2673–2679.

Xing Niu and Marine Carpuat. 2017. Discovering stylistic variations in distributional vector space models via lexical paraphrases. In *Proceedings of the Workshop on Stylistic Variation*, pages 20–27.

Jekaterina Novikova, Ondřej Dušek, Amanda Cercas Curry, and Verena Rieser. 2017a. Why we need new evaluation metrics for NLG. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2241–2252.

Jekaterina Novikova, Ondřej Dušek, and Verena Rieser. 2017b. The E2E dataset: New challenges for end-to-end generation. In *Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue*, pages 201–206.

Jekaterina Novikova, Oliver Lemon, and Verena Rieser. 2016. Crowd-sourcing NLG

data: Pictures elicit better data. In *Proceedings of the 9th International Natural Language Generation conference*, pages 265–273.

Shereen Oraby, Vrindavan Harrison, Abteen Ebrahimi, and Marilyn Walker. 2019. Curate and generate: A corpus and method for joint control of semantics and style in neural NLG. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5938–5951, Florence, Italy. Association for Computational Linguistics.

Shereen Oraby, Sheideh Homayon, and Marilyn Walker. 2017. Harvesting creative templates for generating stylistically varied restaurant reviews. In *Proceedings of the Workshop on Stylistic Variation*, pages 28–36. Association for Computational Linguistics.

Shereen Oraby, Lena Reed, Shubhangi Tandon, TS Sharath, Stephanie Lukin, and Marilyn Walker. 2018. Controlling personality-based stylistic variation with neural natural language generators. In *Proceedings of the 19th Annual SIGdial Meeting on Discourse and Dialogue*, pages 180–190.

Daniel S Paiva and Roger Evans. 2004. A framework for stylistically controlled generation. In *Natural Language Generation*, pages 120–129. Springer.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318.

Ankur Parikh, Xuezhi Wang, Sebastian Gehrmann, Manaal Faruqui, Bhuwan Dhingra, Diyi Yang, and Dipanjan Das. 2020. ToTTo: A controlled table-to-text

generation dataset. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1173–1186.

Steffen Pauws, Albert Gatt, Emiel Krahmer, and Ehud Reiter. 2019. Making effective use of healthcare data using data-to-text technology. In *Data Science for Healthcare*, pages 119–145. Springer.

Baolin Peng, Chenguang Zhu, Chunyuan Li, Xiujun Li, Jinchao Li, Michael Zeng, and Jianfeng Gao. 2020. Few-shot natural language generation for task-oriented dialog. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings*, pages 172–182.

Telmo Pires, Eva Schlinger, and Dan Garrette. 2019. How multilingual is multilingual bert? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4996–5001.

Vassilis Plachouras, Charese Smiley, Hiroko Bretz, Ola Taylor, Jochen L Leidner, Dezhao Song, and Frank Schilder. 2016. Interacting with financial data using natural language. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*, pages 1121–1124.

Weizhen Qi, Yu Yan, Yeyun Gong, Dayiheng Liu, Nan Duan, Jiusheng Chen, Ruofei Zhang, and Ming Zhou. 2020. Prophetnet: Predicting future n-gram for sequence-to-sequencepre-training. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 2401–2410.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8).

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21:1–67.

Sudha Rao and Joel Tetreault. 2018. Dear Sir or Madam, may I introduce the GYAFC dataset: Corpus, benchmarks and metrics for formality style transfer. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, volume 1, pages 129–140.

Vikas Raunak, Arul Menezes, and Marcin Junczys-Dowmunt. 2021. The curious case of hallucinations in neural machine translation. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1172–1183.

Lena Reed, Shereen Oraby, and Marilyn Walker. 2018. Can neural generators for dialogue learn sentence planning and discourse structuring? In *Proceedings of the 11th International Conference on Natural Language Generation*, pages 284–295.

Ehud Reiter and Anja Belz. 2009. An investigation into the validity of some metrics for automatically evaluating natural language generation systems. *Computational Linguistics*, 35(4):529–558.

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

Verena Rieser and Oliver Lemon. 2010. Natural language generation as planning

under uncertainty for spoken dialogue systems. In *Empirical methods in natural language generation*, pages 105–120. Springer.

Ananya B Sai, Akash Kumar Mohankumar, and Mitesh M Khapra. 2022. A survey of evaluation metrics used for nlg systems. *ACM Computing Surveys (CSUR)*, 55(2):1–39.

Keisuke Sakaguchi, Matt Post, and Benjamin Van Durme. 2014. Efficient elicitation of annotations for human evaluation of machine translation. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 1–11. Association for Computational Linguistics.

Natalie Schluter. 2017. The limits of automatic summarisation according to rouge. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 41–45.

Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, et al. 2020. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588(7839):604–609.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1073–1083, Vancouver, Canada. Association for Computational Linguistics.

Thibault Sellam, Dipanjan Das, and Ankur Parikh. 2020. Bleurt: Learning robust

metrics for text generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7881–7892.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Controlling politeness in neural machine translation via side constraints. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Lingui stics: Human Language Technologies*, pages 35–40.

Pararth Shah, Dilek Hakkani-Tür, Gokhan Tür, Abhinav Rastogi, Ankur Bapna, Neha Nayak, and Larry Heck. 2018. Building a conversational agent overnight with dialogue self-play. *arXiv preprint arXiv:1801.04871*.

Mandar Sharma, Ajay Gogineni, and Naren Ramakrishnan. 2022. Innovations in neural data-to-text generation. *arXiv e-prints*, pages arXiv–2207.

Sheng Shen, Daniel Fried, Jacob Andreas, and Dan Klein. 2019. Pragmatically informative text generation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4060–4067.

Xiaoyu Shen, Ernie Chang, Hui Su, Cheng Niu, and Dietrich Klakow. 2020. 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*, pages 7155–7165.

Joonbo Shin, Yoonhyung Lee, and Kyomin Jung. 2019. Effective sentence scoring method using bert for speech recognition. In *Asian Conference on Machine Learning*, pages 1081–1093.

David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. 2016. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484.

David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. 2018. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362(6419):1140–1144.

David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. 2017. Mastering the game of go without human knowledge. *Nature*, 550(7676):354.

Aaron Smith, Christian Hardmeier, and Jörg Tiedemann. 2016. Climbing mont bleu: the strange world of reachable high-bleu translations. In *Proceedings of the 19th annual conference of the European association for machine translation*, pages 269–281.

Amanda Stent, Matthew Marge, and Mohit Singhai. 2005. Evaluating evaluation methods for generation in the presence of variation. In *international conference on intelligent text processing and computational linguistics*, pages 341–351. Springer.

Amanda Stent, Rashmi Prasad, and Marilyn Walker. 2004. Trainable sentence planning for complex information presentation in spoken dialog systems. In *Proceedings of the 42nd annual meeting on association for computational linguistics*, page 79. Association for Computational Linguistics.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. *Advances in neural information processing systems*, 27.

Richard S Sutton and Andrew G Barto. 2018. *Reinforcement Learning: An Introduction*, second edition. MIT press.

Romal Thoppilan, Daniel De Freitas, Jamie Hall, Noam Shazeer, Apoorv Kulshreshtha, Heng-Tze Cheng, Alicia Jin, Taylor Bos, Leslie Baker, Yu Du, et al. 2022. Lamda: Language models for dialog applications. *arXiv preprint arXiv:2201.08239*.

Zhaopeng Tu, Zhengdong Lu, Yang Liu, Xiaohua Liu, and Hang Li. 2016. Modeling coverage for neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 76–85.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 6000–6010. Curran Associates Inc.

Ramakrishna Vedantam, C Lawrence Zitnick, and Devi Parikh. 2015. Cider: Consensus-based image description evaluation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4566–4575.

Ashwin Vijayakumar, Michael Cogswell, Ramprasaath Selvaraju, Qing Sun, Stefan Lee, David Crandall, and Dhruv Batra. 2018. Diverse beam search for improved description of complex scenes. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32(1).

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. In *Advances in Neural Information Processing Systems*, pages 2692–2700.

Marilyn Walker, Amanda Stent, François Mairesse, and Rashmi Prasad. 2007. Individual and domain adaptation in sentence planning for dialogue. *Journal of Artificial Intelligence Research*, 30:413–456.

Peng Wang, Junyang Lin, An Yang, Chang Zhou, Yichang Zhang, Jingren Zhou, and Hongxia Yang. 2021. Sketch and refine: Towards faithful and informative table-to-text generation. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 4831–4843.

Qingzhong Wang and Antoni B Chan. 2019. Describing like humans: on diversity in image captioning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4195–4203.

Zongsheng Wang, Yunzhi Bai, Bowen Wu, Zhen Xu, Zhuoran Wang, and Baoxun Wang. 2018. A prospective-performance network to alleviate myopia in beam search for response generation. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 3608–3618.

Tsung-Hsien Wen, Milica Gašić, Dongho Kim, Nikola Mrkšić, Pei hao Su, David Vandyke, and Steve Young. 2015a. Stochastic language generation in dialogue using recurrent neural networks with convolutional sentence reranking. In *In Proceedings of SIGdial. Association for Computational Linguistics.*

Tsung-Hsien Wen, Milica Gašić, Nikola Mrkšić, Lina M. Rojas-Barahona, Pei-Hao Su, David Vandyke, and Steve Young. 2016. Multi-domain neural network language generation for spoken dialogue systems. In *Proceedings of the 2016 Confer-*

ence of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 120–129.

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

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1112–1122.

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

Sam Wiseman, Stuart Shieber, and Alexander Rush. 2018. Learning neural templates for text generation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3174–3187.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, et al. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi,

Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Lukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Gregory S. Corrado, Macduff Hughes, and Jeffrey Dean. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

Linting Xue, Noah Constant, Adam Roberts, Mihir Kale, Rami Al-Rfou, Aditya Siddhant, Aditya Barua, and Colin Raffel. 2021. 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*, pages 483–498.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. *Advances in neural information processing systems*, 32.

Xiaoxue Zang, Abhinav Rastogi, Srinivas Sunkara, Raghav Gupta, Jianguo Zhang, and Jindong Chen. 2020. Multiwoz 2.2: A dialogue dataset with additional annotation corrections and state tracking baselines. In *Proceedings of the 2nd Workshop on Natural Language Processing for Conversational AI*, pages 109–117.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q Weinberger, and Yoav Artzi. 2020a. Bertscore: Evaluating text generation with bert. In *International Conference on Learning Representations*.

Yizhe Zhang, Siqi Sun, Michel Galley, Yen-Chun Chen, Chris Brockett, Xiang Gao,

Jianfeng Gao, Jingjing Liu, and William B Dolan. 2020b. Dialogpt: Large-scale generative pre-training for conversational response generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 270–278.

Xueliang Zhao, Wei Wu, Can Xu, Chongyang Tao, Dongyan Zhao, and Rui Yan. 2020. Knowledge-grounded dialogue generation with pre-trained language models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3377–3390.