Controlling Hallucination while Generating Text from Structured Data

A thesis presented for the degree of
MPhil Machine Learning and Machine Intelligence

Tisha Anders
Downing College

Department of Engineering
University of Cambridge

Acknowledgements

First and foremost, I would like to express my deepest gratitude to my supervisor Prof. Bill Byrne, who advised and supported me for countless hours, attending to me promptly whenever I was running into problems. Our discussions provided me with completely new insights, beyond the realm of this thesis. He was the best possible supervisor one could have asked for.
Further, I would like to thank my college tutor Dr. Joseph Webster, for providing me with administrative, as well as moral support, when an injury forced me to take a break from my thesis.
I also would like to thank my PhD supervisors Andy and Alex for a long inspiring discussion, with lots of feedback and inspiration; and especially Alex for helping me out instantly with practical issues.
Finally, I would like to thank Levin, Michael, Fredi, and Oliver for patiently providing human judgements on my system output; Chelsea, Kris, Laurence and Ajulik for their magic hacking when I accidentally removed half of my thesis code.

Declaration of originality

I, Tisha Anders of Downing College, being a candidate for the MPhil in Machine Learning and Machine Intelligence, hereby declare that this report and the work described in it are my own work, unaided except as may be specified below, and that the report does not contain material that has already been used to any substantial extent for a comparable purpose.

Tisha Anders
Thursday 16th September, 2021

External software:
All my code has been written in PyCharm, using Python and Shell. to enable a comparison with their project, I have used the pre-processing scripts for datasets WebNLG and ViGGO, by Harkous, Groves, and Saffari, 2020, available at `https://github.com/amazon-research/datatuner/tree/main/paper`. Plots have been made with Matplotlib and Seaborn (`https://seaborn.pydata.org/index.html`). Model implementations with pre-trained checkpoints, as well as training & evaluation tools were retrieved from the HuggingFace Library (`https://huggingface.co/`).

Word count:
14979

Abstract

Data-to-text generation means the formulation of fluent text, given a set of information encoded in an abstract meaning representation (knowledge graph, table, etc.).
One fundamental challenge in data-to-text generation is that the set of information contained in the text must exactly match the set of information covered by the information source. Existing data-to-text systems however tend to omit, falsify, or hallucinate facts in their generated text. This pathology is additionally fuelled by low semantic overlap among most available data-to-text corpora.
Common approaches to encourage semantic accuracy involve endowing the language model with additional structure, meant to prevent the model from generating unfaithfully (i.e. diverge from the facts) in the first place. This work however aims to distribute the task of faithful generation across several system components, placing less responsibility on the generator. Our approach can thus be partitioned into three stages - dataset preparation, generation of candidate texts, and candidate filtering.
We propose an end-to-end data-to-text generation system, re-using the generator-classifier approach of the DataTuner (Harkous, Groves, and Saffari, 2020). In addition however, we pay special attention to the dataset preparation, actively increasing faithfulness among the examples, and training on mixed corpora. We find the semantic cleaning and the mixed-corpus training to be a viable means of improving generator quality.
In the next step, we train a *T5 langugage generator* on our now de-noised joint data-to-text corpus. We generate five candidate sentences for each information source. Finally, we let a *RoBERTa semantic fidelity classifier* pick the system output among the candidates.

We achieve higher pre-filtering METEOR scores than Harkous *et al.*, and increase METEOR by 4.72% through filtering with RoBERTa, compared to 1.19% in Harkous *et al.*. This may be an unfair comparison, due to METEOR picking up on more overlap in our de-noised datasets. Nevertheless, the performance increase measured from employing de-noised datasets as well as mixed-corpus training ultimately demonstrates an enhancement to the DataTuner. The effect of qualitative differences between their generator (GPT-2) and ours (T5) are difficult to assess.
In human reviews of T5's output, we rarely spot hallucination, but numerous omissions, the opposite phenomenon of hallucination. We attribute this observation to a scale of locquacity among language models, i.e. they are either too 'talkative' or too 'quiet'.

Contents

1 Introduction

Ever-growing databases, holding unwieldy amounts of data, theoretically grant us access to an abundance of knowledge, which we however rarely retrieve due to the practical inaccessibility of the storage format. Mechanisms which automatically transform the inaccessible data structures into fluent text could thus remove the need for a laborious routine of information retrieval, information restructuring and write-up. On the other hand, tasks that involve smaller, more manageable datasets, may come with an aspect of repetitiveness when translating into text, for example the routinely presentation of a small dataset in weather forecasts, Premier League rankings, or medical reports. Chatbot systems must be able to incorporate information retrieved from a knowledge base into the conversation. Yet the automation of the above tasks remains non-trivial, and simply plugging values into pre-made templates is not sufficient.
We shall refer the general task described above as data-to-text generation.

With the considerable advances in Natural Language Processing in general, and *Natural Language Generation*, in particular since the invention of the Transformer (Vaswani et al., 2017), neural text production has taken a serious step towards commercial applicability (R. Dale, 2019, ch. 8), such that the formulation of knowledge base content into text is now a potentially achievable skill.

In any conditional generation[1] setting, staying faithful to the input has proved a challenge even with modern-day neural models. In particular, models tend to *hallucinate* persistently, i.e. include information in the generated text that did not appear in the data. This of course questions the real-life applicability of such neural data-to-text systems, as a lack of content accuracy makes them redundant.
Looking at an example, a table was first linearised into slot-value pairs, from which a sentence was generated, containing hallucinations:

data: {(*name*, Sir Francis Drake); (*born*, 1540); (*occupation*, naval officer)}

1. **text:** "Sir Francis Drake, born in 1540, was an English naval officer."

Although it is true that Drake was English, the model needs to content itself

[1] Conditional generation refers to having generating text after seeing an input (e.g. a source sentence to be translated, a paragraph to be summarised, a dataset to be put into words, etc.)

with the given data. Of course, models mostly hallucinate false information, such as saying

2. **text:** "Sir Francis Drake, born in 1540, was a naval officer and tradesman."
3. **text:** "Sir Francis Drake, a naval officer, born in 1540, was the first man to sail around the world."

Mdels thus add small to big portions of semantic content that is not supported by the data.
Other semantic error types of course exist, such as *repetitions*, *value errors* and *omissions*, whose meanings are evident from the terminology.

This thesis aims to facilitate semantically faithful data-to-text generation, in particular with regards to hallucinations. The problem of controlling hallucinations can be approached from various angles, such as dataset pre-processing, refining the model structure, pre-training and many more. We will take a *sieving approach*, generating a set of candidate sentences first, and then selecting the most faithful candidate according to a semantic fidelity classifier network. This approach was first used by Harkous, Groves, and Saffari, 2020 in their DataTuner, and will be described in more detail in Subsection 3.1.
Initially, it was planned to use DataTuner as a baseline system, due to its solid results and elegant approach. The DataTuner separates the task of faithful data-to-text generation into *candidate text generation* and *semantic fidelity classification*, instantiating a separate model for each of the two subtasks. The code is available on Amazon Research's Github[2]. However, after extensive efforts to run the the DataTuner, it was confirmed with the authors (Harkous, Groves, and Saffari, 2020) that relevant files were missing from the Github page.
Instead of performing changes on the DataTuner baseline system, it was thus decided to build a system with the same generator-classifier architecture, but some crucial differences:

- We use Google's T5 for text generation, whereas the DataTuner uses OpenAI's GPT-2
- We apply additional cleaning to the datasets used by the DataTuner

[2]`https://github.com/amazon-research/datatuner`

- We explore different training techniques, such as
    - conjoining disparate data-to-text corpora into one mixed corpus (for both generator and classifier)
    - applying a hard-engineered procedure to increase faithfulness among the data-to-text examples before training
    - linearising the abstract meaning representation of the data in a simplified manner
    - varying the number of labels in the semantic fidelity classifier
    - testing different model sizes of T5 and RoBERTa

In Section 2, the background of our work will be laid out, introducing basic language modelling terminology and concepts, work done on faithfulness in data-to-text generation, and the tools we use to measure performance, including their limitations.
In Section 3, we provide practical details of our approach, concerning model configuration, cleaning & linearisation methods, training dataset assembly, implementation issues and how we want to assess our generator's quality. Next, in Section 4, we present in logical order the set of experiments done on text generation, semantic fidelity classification, and the combined system. In Section 5, results to the experiments will be revealed and discussed, evaluating any significant findings. Figure 6 finally draws all findings together, explaining their significance in the current research context, and points out opportunities for further research.

2 Background

2.1 Neural approaches to Language Modelling

The task of *language modelling* describes the the attempt to integrate an understanding of language into a model. More formally - by looking at the context of a piece of text, a language model assigns a probability to the string, based on statistical/handcrafted rules, or a learnt latent-variable representation of the language. Language models frequently serve as a fundamental building block in any NLP system, in the pursuit of a basic language endowment.

Previously, NLP tasks were portioned into subtasks, for which a simpler system could cover the restricted problem space sufficiently. For instance language generation, which plays a substantial role in verbalising structured data, was previously segmented into content determination, structuring/grouping of information, and surface realisation (Reiter and R. Dale, 1997), where the subtasks relied on manually engineered system behaviour. The general approach has now shifted towards *end-to-end neural approaches*, incorporating each of the previously mentioned steps. For the remainder of this subsection, basic concepts of those end-to-end neural systems will be described.

2.1.1 Pre-Trained Language Models

With the removal of task structure in end-to-end neural approaches, the system is faced with the responsibility of finding its own path through the subtasks, resulting in a significantly more challenge. The *pre-training of language models* has been found to equip NLP systems with a general understanding of language, which prepares them for handling this responsibility. In our context, the language model takes on the form of a large neural network with millions of parameters. More will be said about the nature of the neural networks in Subsection 2.1.4.

Irrespective of the task to be completed, the language models are trained on large[3], unspecific text corpora in an unsupervised fashion, for instance by being asked to predict the identity of words masked in the text (Radford and Narasimhan, 2018):

Given a set of tokens $\mathcal{U}$ with $u_i \in \mathcal{U}$, and e.g. a symmetrical context window

$$\mathcal{C}_j = \{u_{j-k}, ..., u_{j-1}, u_{j+1}, ..., u_{j+k}\}$$

[3]The number of training examples being in the billions easily

of size $2k$, estimate the probability of token u_j, using a neural network's parameters θ:

$$p(u_j|\mathcal{C}_j;\theta) \tag{1}$$

Parameters θ are trained with complex iterative optimisation methods, such as stochastic gradient descent, to maximise an objective function, e.g.

$$\mathcal{L}(\theta) = \sum_j \log p(u_j|\mathcal{C}_j;\theta) \tag{2}$$

The ability to predict words from the context is believed to induce the typically gained general understanding of language and its structure.
Other forms of pre-training exist, such as training an *embedding*, a vectorial representation of single words (Mikolov et al., 2013) or sentences (Q. Le and Mikolov, 2014), which however we will not consider here.

Pre-training from scratch any of this thesis' language models would require extensive computational resources for parallelised training, as well as weeks of waiting time. It is thus common practice to download pre-trained models. We download our pre-trained language models from `https://huggingface.co/transformers/pretrained_models.html`.
Another view on pre-training is that it initialises a model's parameters so that they can be quickly adapted to specific task. The adaption to a a certain task is called *fine-tuning* and will be described in the coming sub-subsection.

2.1.2 Fine-tuning

Having pre-trained a model, the model can be adapted to a downstream task, by fine-tuning it on a smaller[4], task-specific dataset in a supervised fashion. NLG-specific task examples include text summarisation, image captioning, question answering, data-to-text generation, story writing, and arguably machine translation.

In practice, fine-tuning involves training a model from a weight checkpoint stored during pre-training.

[4]ranging from very few (e.g. 5) to few (e.g. 200) examples (Z. Chen, Eavani, W. Chen, et al., 2020)

2.1.3 Transformers & Attention

Traditionally, sequence models for language processing relied on variants of Recurrent[5] Neural Networks (RNNs) (e.g. Sutskever, Vinyals, and Q. V. Le, 2014). The processing of a sequence $\mathbf{x} = (x_1, x_2, ..., x_T)$ in RNN-related models evolves chronologically, with each token x_t marking one computation time step $t \in \{1, ..., T\}$. From token x_t and the previous hidden state h_{t-1}, a hidden state h_t is computed:

$$h_t = \mathcal{F}(x_t, h_{t-1})$$

Due to the derivation of h_t from h_{t-1}, intra-sequential dependencies can only be established between tokens within certain proximity[6], since the signal from earlier hidden states and thus inputs weakens over time.

With the recent advent of the Transformer (Vaswani et al., 2017), the field of Natural Language Processing (NLP) has been revolutionised, abandoning the sequential nature found in RNN architectures. Instead, the Attention mechanism underlying the Transformer views the sequence as a whole, allowing dependencies to be established position-independently. Essentially, when trying to encode the i^{th} word w_i in a sequence, Attention gives a weighting over each word $w_j \in \{w_j\}_{j=1}^{N}$, determining its relevance in encoding w_i.

While Attention is by far not the Transformer's only relevant mechanism, it yet marks the fundamental difference between RNN models and the Transformer. Therefore, we will give a quick mathematical overview of Attention in Transformers below.
Words are commonly represented as points in high-dimensional space (*embedding vectors*) in modern-day language models (e.g. Pennington, Socher, and C. Manning, 2014). To retain a notion of location in the input elements, a *positional encoding* term is added to every embedding vector $\mathbf{x}_i$ to mark its position in the sequence:

$$\mathbf{x}_i \longleftarrow \mathbf{x}_i + PE[i]$$

To enable attention over a given input sequence $\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_N)$ of embed-

[5]We treat RNNs as a collective term for vanilla RNNs, LSTMs, Gated RNNs, etc.
[6]One-directional proximity for unidirectional (standard) RNNs, and general proximity for bidirectional RNNs

ding vectors, we first compute for each vector $\mathbf{x}_i$ a *query* vector $\mathbf{q}_i \in \mathbb{R}^{d_k}$, a *key* vector $\mathbf{k}_i \in \mathbb{R}^{d_k}$ and a *value* vector $\mathbf{v}_i \in \mathbb{R}^{d_v}$, using three previously trained matrices:

$$\boldsymbol{\beta}_i = W^{\beta}\mathbf{x}_i, \quad \text{for } \beta \in \{q, k, v\}$$

Then, to encode word w_i, we regard its embedding vector $\mathbf{x}_i$ as the query, and dot it with each word's key vector:

$$[\mathbf{q}_i\mathbf{K}^{\top}]_j = \mathbf{q}_i \cdot \mathbf{k}_j, \text{ for } \quad \mathbf{q}_i, \mathbf{k}_j \in \mathbb{R}^{d_k}, \mathbf{K} \in \mathbb{R}^{N \times d_k}$$

In Vaswani *et al.*, the resulting vectors are normalised by $\sqrt{d_k}$ due to the subsequent application of $softmax$, which has vanishing gradients for arguments with large absolute values. The importance of word w_j in encoding w_i is thus given by

$$\alpha_{ij} = softmax(\frac{\mathbf{q}_i\mathbf{K}^{\top}}{\sqrt{d_k}})_j \quad \in \mathbb{R},$$

so that the encoding of w_i is a weighted average of each word's value vector:

$$\begin{aligned} c_i &= \sum_{j=1}^{N} \alpha_{ij}\mathbf{v}_j \\ &= \sum_{j=1}^{N} softmax(\frac{\mathbf{q}_i\mathbf{K}^{\top}}{\sqrt{d_k}})_j\mathbf{v}_j \quad \in \mathbb{R}^{d_v} \end{aligned} \tag{3}$$

for word w_i. Figure 1 shows a practical example of the above concept, from an application in Machine Translation.

The structure that makes most extensive use of Attention is the Transformer (Vaswani et al., 2017), which is a basic building block of the more complex models we use.
The Transformer enables an auto-regressive[7] encoder-decoder structure. The *encoder* maps input sequence $\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_N)$ to a continuous sequential representation $\mathbf{Z} = (\mathbf{z}_1, \mathbf{z}_2, ..., \mathbf{z}_N)$, which the *decoder* takes as an input to generate an output sequence $\mathbf{Y} = (\mathbf{y}_1, \mathbf{y}_2, ..., \mathbf{y}_N)$. Encoder and decoder both consist of L identical layers, respectively. Every *encoder*-layer consists of two sublayers

[7]i.e. previously generated tokens are considered part of the model input

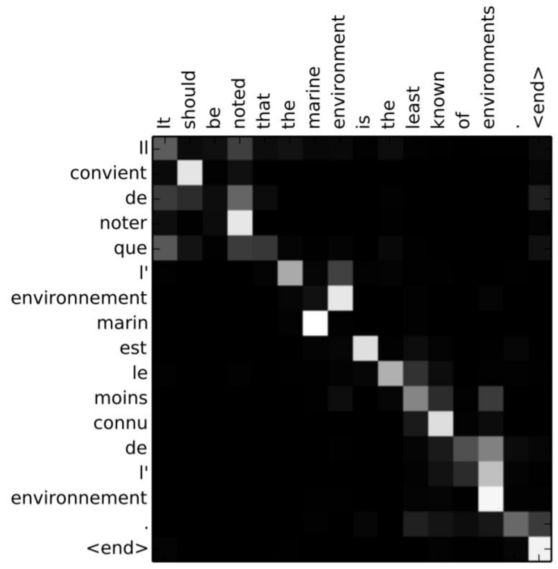


Figure 1: **Attention weights a_{ij} in translating English to French.**
Figure taken from Bahdanau, Cho, and Bengio, 2014. High a_{ij} correspond to brighter cells.
Most English words can just be literally translated from English to French. Thus little attention needs to be paid to source words $w_{\neq i}$ resulting in one highly dominant cell in most rows. Diagonal cells tend to be brighter since English and French have similar word order. Satisfyingly, increased attention is visible beyond the diagonal when there exist grammatical relationships between words (e.g. relative pronoun *que* grammatically relates to subject *It*, and it indeed pays attention to *It* as well, beyond its literal translation *that*).

(multi-head self-attention[8], position-wise feed-forward neural network) wrapped by a residual connection and layer normalisation each. The m *decoder*-layers work like the encoder-layers, with the addition of a multi-head attention over encoder-output $\mathbf{Z}$. Multiple heads allow the transformer to consider different representation subspaces (at different positions).

2.1.4 The Transformer architecture in use

Attention and the emergent Transformer architecture, as discussed in Subsection 2.1.3, allow drawing dependencies between any set of sequence elements, regardless of their relative locations. Transformers can be integrated into in more complex models, such as the models we will be using for text classification & generation.

An early well-known large-scale application of Transformers is the BERT model (Devlin et al., 2018), whose base-version consists of $L = 12$ Transformer Encoder blocks and $H = 12$ heads. A refined general setup for working with BERT is given in Yinhan Liu et al., 2019), i.e. RoBERTa ("Robust optimisation BERT approach"). RoBERTa has the same transformer-based architecture as BERT, but differs in pre-training techniques and data preprocessing. Although RoBERTa technically only describes a way of using the BERT model, we will continue to refer to RoBERTa as model.

As opposed to BERT, RoBERTa is trained on 160GB of data, amounting to 10 times the original BERT corpus. Furthermore, Liu *et al.* note that their batch sizes up to $2'000 - 8'000$, in comparison to BERT's 256, reduce perplexity w.r.t. unseen examples, and can potentially increase optimisation speed, and even improve performance on downstream task.

During masked language model pre-training, Devlin *et al.* keep the selection of masked tokens static (determined during pre-processing), whereas Liu *et al.* mask tokens dynamically, by applying a new masking to a sequence every time it is fed into the model. Finally, Liu *et al.* have undertaken careful tuning of the Adam optimiser's parameters, as well as the learning rate. Both models are trained on a thematically broad corpus, consisting of English Wikipedia and the BookCorpus (Zhu et al., 2015), while RoBERTa is additionally trained on 63 English news articles (CC-News[9]), Reddit posts (OpenWebText`http://`

[8]*multi-head:* for each word w_i store H different $\mathbf{q}_i, \mathbf{k}_i, \mathbf{v}_i$, i.e. $\{(\mathbf{q}_{i_h}, \mathbf{k}_{i_h}, \mathbf{v}_{i_h})\}_{h=1}^{H}$ ("H-headed attention").
Each head linearly transforms $\mathbf{q}_i, \mathbf{k}_i, \mathbf{v}_i$ its own way: $\boldsymbol{\beta}_{i_h} = W_{i_h}^{\beta} \boldsymbol{\beta}_i$, for $\beta \in \{q, k, v\}$.

[9]`http://web.archive.org/save/http://commoncrawl.org/2016/10/`

web.archive.org/save/http://Skylion007.github.io/OpenWebTextCorpus.)
RoBERTa in our case takes as input a sequence

$$[CLS]\ x_1, x_2, ..., x_N\ [SEP]\ y_1, y_2, ..., y_M\ [EOS],$$

where $[CLS]$ signifies a classification task involving source sequence $\{x_i\}_{i=1}^N$, and output sequence $\{y_i\}_{i=1}^M$, and a label $l \in \{l_k\}_{k=1}^K$ to be predicted.
The two subsequent tokens represent a separator between source - & output sentence, and an end-of-sentence marker.

From the above example, it is apparent that RoBERTa is well-suited to text generation ($\{x_i\}_{i=1}^N$ given before generation; $\{y_i\}_{i=1}^M$ generated on the basis of $\{x_i\}_{i=1}^N$). HuggingFace[10] however also provides a version with a sequence classification/regression head on top (a linear layer on top of the pooled output), which lets us apply RoBERTa to our multi-label classification problem for sequence pairs (problem set described in 2.3.2).

2.2 Data-to-Text generation

In the first part of this Subsection (2.1), we have introduced some methods for training Language Models, such as pre-training models on general problems and then using the obtained parameter values as clever initialisations for fine-tuning to downstream tasks. We have also studied powerful state-of-the-art model architectures centered around Attention which we will deploy later.
With the sequence-to-sequence methods and models discussed, we shall now turn over to the problem set at hand.

2.2.1 Problem Setup

The recent upsurge in powerful and easily applicable sequence-to-sequence models such as from HuggingFace[11] has opened the opportunity data-to-text generation, where data & text constitute one sequence each.

newsdataset-available.

[10]https://huggingface.co/transformers/model_doc/roberta.html#robertaforsequenceclassification

[11]https://huggingface.co/inference-api?utm_source=Google&utm_medium=Search&utm_campaign=Transformers+10x+Faster&utm_id=12055067954&gclid=CjwKCAjwx8iIBhBwEiwA2quaq8nOC4FWWrIAejitbVm0oCiTbzT3RaZhRUcrFVOupnU6S_KRLryLgBoCKHgQAvD_BwE

Formally, the task is defined as putting into words a collection of structured data **X**, where **X** is initially often non-sequential, but will in our case be linearised to be processed by a sequence-to-sequence model. Examples of shapes **X** can be in include

- *tables*, e.g. linearised to sequences of slot-value pairs
- *knowledge graphs*, e.g. linearised to sequences of knowledge triplets

Due to the inherently different structure of tables vs. knowledge graphs, simply applying sequential models to both data types might seem ill-advised. However, Chen *et al.* have contrasted the performance of sequence vs. graph encoder networks on knowledge-graph data, and found that sequential models score higher on several fluency metrics (BLEU, METEOR, ROUGE) than graph models (W. Chen et al., 2020). Chen *et al.* also find that graph models lead to more semantically faithful generation, however since the generator's focus is on generating fluent text, we picked a sequence encoder. Determining faithfulness will be handled by a semantic fidelity classifier, introduced in 2.3.2.
An important constraint on data-to-text generation is remaining semantically faithful, i.e. generating only content that is supported by the data, but also covering all of the information present in the data.

2.2.2 Linearisation techniques on datasets

When encoding meaning in a structured manner, we rely on concepts of almost spatial nature to help us associate certain pieces of information more closely than others, or relate pieces of information under a certain notion. For instance, the concept of a table *cell*, and the emerging rows/columns, assign a more close relationship to information within the same row/column, or perhaps emphasise diagonal values, etc. . Furthermore, the relationship between a name row/column and actual values is of a classificational nature, whereas relationships between value-cells are purely factual. Similarly, in a knowledge graph, *arcs* stand for relationships between nodes (i.e. predicates linking entities), but the emergent tree also states a notion of proximity/distance between any two pieces of information within the graph (Hamilton et al., 2018).
It is thus obvious that the information content of a structured meaning representation is not exclusively based on the data contained, but also on the structure

in which the data is configured.

Sequential models, such as Transformer-based architectures, can however most easily process sequences, demanding from us a *linearised* version of the spacially structured input data. Typically, a linearised sequence `S` of structured data `D` consists of meaning-bearing tokens and *special tokens*, where the special tokens help emulating a notion of structure. For example, in a knowledge graph setting, we have the node `N: Einstein` connected with two other nodes `N: 1879` and `N: Ulm`, by arcs `A: birthyear` and `A: birthplace`, respectively. We can now use special tokens to encode this graph:

```
(Einstein, birthyear, 1879)
(Einstein, birthplace, Ulm)
```

Above, it is implied that the slot roles in the above triplets are `(node, arc-name, node)` Employing more complex tokens to separate the slots, we can even represent each slot's role in assembling a graph edge:

```
<subject> Einstein <predicate> birthyear <object> 1879
<subject> Einstein <predicate> birthplace <object> Ulm
```

A concatenation of the above edge representations (e.g. with another special token `;`) then symbolises an entire graph. In the linearisation technique above, spatial proximity between nodes is only conveyed very implicitly, since any permutation of a set of triplets within the linearisation encodes the same knowledge graph.
While it is technically possible to fully encode complex information structures within a linearised string, it remains to be seen to what extent the system can to profit from the extra information. Furthermore, a detailed representation of structure in the linearisation might distract the data-to-text system from the actual information content. Depending on their computational resources, researchers are also often limited in sequence length, so that the aim is to keep the input sequences as short as possible. For the above reasons, in practice, linearisation almost necessarily comes with information loss compared to the original structure.

The above linearisation technique is the standard manner in which the WebNLG dataset (Colin et al., 2016) is linearised, although originally WebNLG is given in an `xml`-tree format[12]. The extensive attention mechanism in the Transformer (as described in 2.1.3) allowing multiple representation subspaces, gives rise to questioning the value of using differentiated separator tokens. It remains to be seen whether Transformer-based T5 is able to learn the meanings of the standardised triplet positions itself. In our experiments, we will investigate employing one standard separator token `|`) instead of `<subject>`, `<subject>` & `<subject>`, leaving the role of standardised slots to be learnt by the model. The use of a standard separator might be less distractive, channelling the model's attention to the data content.

There are of course datasets that do not require linearisation, since they do not originate from an information structure. Instead, they have been especially created for data-to-text generation. For instance, the ViGGO dataset (Juraska, Bowden, and Walker, 2019), which we will employ due to its perfect factual reliability, consists of dialogue acts, with each dialogue act being associated with a set of slot-value pairs, e.g.:

```
give_opinion(name [SpellForce 3], rating [poor], genres [strategy, role-playing])
```

In summary, linearisation is a technique whereby the content information from a datastructure are merged into a linearised sequence, in combination with special tokens encoding (parts of the) original structure in which the information were grouped. Linearisation allows us to feed structured data into sequential models, where loss of structural information has to be weighed up against input sequence length, complexity/readability as well as system performance.

2.2.3 Diversions from Input Data

Among the various examples of conditional[13] language generation (e.g. summarisation, question answering, translation, image captioning), an ever-present phenomenon is the factual diversion from the information given in the source when generating text (image-to-text: Rohrbach et al., 2018; text-to-text: Nie et al., 2019). A diversion from the input text is when a fact appearing in the generation is not supported by the input text, or vice-versa. Even state-of-

[12] `https://github.com/WebNLG/WebNLG-Text-to-triples`

[13] Text being generated, *given* an information value (some text or an image). Other types of generation include creative tasks such as storytelling.

the-art models, which generate highly fluent text, suffer from this pathology, rendering them ill-suited for commercial application, since their output can not be trusted in general. The diversion pathology is thus a bottleneck to the astounding progress in conditional generation and its fluency.

In order to tackle diversion, we first require a means of quantisation thereof. Factual errors in generated text are subtle and difficult to measure (more in 2.3.2). Due to their subtle nature, a trade-off is to be made between unravelling the subtleties, and feasibility. With a more refined error analysis, the model determining the error type requires more intricacy, increasing the classification cost.
A model that classifies text according to their semantic faithfulness will be discussed in Sub-subsection 2.3.2.

2.2.4 Existing remedies to diversion

As mentioned in the 2.2.1, unfaithfulness, in particular hallucination, is a persistent problem in conditional generation with neural models. For the data-to-text generation task, one contributing factor, though not the only culprit, is the ubiquitous semantic diversion of the reference sentence (which is given as ground truth) from the structured data in data-to-text corpora. Parikh et al., 2020 note that the target sentence frequently contains information that cannot be directly inferred from the data source (i.e. hallucination).
From a naive perspective, one would thus suggest curing hallucinative generation by training a data-to-text system on an as-clean-as-possible dataset. Parikh et al., 2020 have constructed a dataset cleaning procedure to obtain semantically perfectly aligned data-text pairs. The semantic alignment is ensured a hand-engineered system in combination with human review. However, when training a BERT-to-BERT model on the Books Corpus (Zhu et al., 2015) cleaned with Parikh *et al.*'s method, the best-performing model only generates 78% of its sentences so that all generated meaning is supported by the data (measured through human assessment of information precision). The 22% proportion of sentences containing excess information despite the faithful-dataset approach proves that hallucinative behaviour cannot solely be a consequence of data noise, and that modelling weaknesses must be considered.

A piece of model-structure that aims to directly address the hallucination problem is the *copy-generate gate*, e.g. within an LSTM positional encoder, alongside a powerful pre-trained language model (GPT-2[14]), as described in Z. Chen, Eavani, Yinyin Liu, et al., 2019. The copy-generate gate p_{copy} is a linear transform Φ of LSTM encoder hidden states c_t, state weights s_t and decoder inputs x_t:

$$p_{copy} = sigmoid[\Phi(c_t, s_t, x_t)] \quad \in (0, 1),$$

built into the LSTM decoder. p_{copy} acts as a soft switch within data-to-text context selection, implying two modes, a *copy-from-data* mode and a *generate-with-language-model* mode. The switch between the two modes allows for explicitly learning when to generate novel text (instead of copying from the source), which upon perfection could avoid generating superfluous text, i.e. hallucinations.
While this additional-structure approach suggested by Chen *et al.* is worthy of noting, their dataset construction process significantly simplified the task, as any examples with out-of-vocabulary (OOV) tokens in the reference (compared to the data source) were left out. With the reference tokens being a subset of the source tokens, the copy-generate gate is merely trained on comparing token identities. Therefore, Chen *et al.* are not simulating real-life data-to-text scenarios, where heavy rephrasing of the original data is necessary to create fluency among the generation. Hence why the approach is only evaluated on significantly constricted topic landscapes (WikiBio: Humans, Books & Songs, each separately). Thus the method's fitness for a more general setting in terms of paraphrasing-flexibility & topic landscape is not evident from this paper.

[14]`https://github.com/openai/gpt-2`

Listing 1: **An example of LDC2017T10,** a high-lexical-variability dataset. Pre-processed by Harkous *et al.*, to be fed to RoBERTa semantic fidelity classifier.

```
{
    "data" : "(respond <:ARG0>
                (country <:name> (United States))
                <:ARG1> (develop <:mod> (that))
                <:ARG2> (condemn <:manner> (swift)))",

    "text" : "The United States responded to that development with
        swift condemnation."
}
```

Another example of the additional-structure approach is the DataTuner (Harkous, Groves, and Saffari, 2020), where the additional structure is less discreet, forming one of two building blocks: The data-to-text system consists of a generator and a classifier, where the generator produces e.g. 5 candidate sentences and the classifier selects the one deemed most faithful to the data source.
The DataTuner has been tested on 4 distinct datasets (WebNLG, LDC2017T10, ViGGO, Cleaned E2E) with diverse meaning representations, each of them serving a different purpose:

- *WebNLG:* generalisation to unseen topics in the test set
- *LDC2017T10:* topic variability and high lexical diversity/difficulty[15]
- *ViGGO & Cleaned E2E:* semantic faithfulness

Examples of ViGGO and WebNLG can be found in Subsection 3.2.2 (since we will be using those datasets). Examples of LDC2017T10 and Cleaned E2E can be found in Listings 1 & 2, respectively.
The DataTuner on average generates 85% = (84%+78%+86%+92%)/4 of the sentences faithfully (according to 3 M-Turkers). The average is taken over the faithfulness rate on each dataset listed above. The average of 85% a significantly higher percentage than the copy-gate model (78%), which is even more notable considering the DataTuner's training data difficulty level. Both DataTuner and Chen *et al.*'s copy-gate approach use a GPT-2 language model, so that the comparison between the two approaches is fair, and lets us conlude that the generation-classification approach should be preferred over a pointer-generator

[15]text difficulty is measured with the New Dale-Chall readability score (E. Dale and Chall, 1948)

Listing 2: **An example of Cleaned E2E,** a dataset for perfect semantic fidelity. Pre-processed by Harkous *et al.*, to be fed to RoBERTa semantic fidelity classifier.

```
{
    "data" : "<name> name=[Zizzi]; <area> area=[riverside]
<eatType> eatType=[coffee shop]",

    "text" : "You can find a coffee shop named Zizzi in the
        riverside area"
}
```

network.

2.2.5 Harnessing the power of strong language models

This subsection will briefly introduce and justify our approach in tackling hallucination, with special regards to the additions made compared to the previous literature. While modern-day language models (GPT-2/3, T5, RoBERTa, etc.) are very powerful, and diving into their structure to make them more powerful is difficult, we decided to pay attention to harnessing their full power by constructing training sets in an advantageous way:

Considering that no complete solution to the hallucination pathology has been found, we aim to draw together several existing mechanisms into one model, and combine them with preprocessing/training techniques, in order to maximally encourage faithfulness, and in particular limit hallucination.

Due to its high semantic accuracy and robustness in different data settings, we will adopt the DataTuner's *generator-classifier approach*, effectively distributing the responsibility of ensuring faithfulness across two models. While we hope to construct a generator that generates maximally faithfully, we obtain an additional backup from the classifier, which can detect & rule out unfaithful generations.

Previous work commonly paid little attention to dataset assembly, focussing mainly on model structures. For example, the previously reviewed paper by Z. Chen, Eavani, Yinyin Liu, et al., 2019 states *"[...] we collect datasets from two new domains: Books and Songs by crawling Wikipedia pages. After filtering and cleanup, we end up with 23,651 instances [...]"*, allowing no analysis of preprocessing methods. The DataTuner paper mentions the addition of special tokens, indicating grammatical roles of text-tokens, e.g. in WebNLG:

Albert Einstein||birthyear||1879; Albert Einstein||occupation||physicist

→ <subject> Albert Einstein <predicate> birthyear <object> 1879;

<subject> Albert Einstein <predicate> occupation <object> physicist

In addition to those formal dataset adjustments, we propose a simple yet effective procedure of semantic cleaning, i.e. making noisy datasets more faithful. As Parikh et al., 2020 describe, most data-to-text datasets are highly noisy, essentially preprogramming hallucinative behaviour in any data-to-text system. However, the dataset cleaning method proposed by Parikh et al., 2020 is mainly a benchmark for testing a model's *ability* to narrow its focus to a subset of the given information, creating an unnatural generation task. We have thus identified a need for a less invasive, quick-to-implement method which does not alter the nature of the task, and can thus be included in any data-to-text system training procedure. The cleaning method will be described in Section 3.3.

Given that better language models also tend to result in higher semantic accuracy, we aim to fully exploit the language model factor, without actually altering the model structure.
Through training on a mix of datasets all at once, we will reap two benefits: On the one hand we will be able to provide the decoder with more examples of how to generate fluent text. On the other hand, we shift the focus from perfecting the meaning retrieval for one particular dataset (i.e. way of encoding meaning) towards a structure-agnostic understanding of meaning, a skill that signifies general language understanding, and thus correlates with strong language models.

In this subsection, we have presented the task of data-to-text generation, alongside some basic pre-processing techniques such as the introduction of segmentation tokens to segment the data-to-text examples into meaningful chunks for easier language understanding in the model. We have described the challenge of semantic diversions in generations w.r.t. the data source, in particular with regards to hallucination, linking it to the omnipresence of noise in typical data-to-text datasets, as well as weaknesses in language modelling. In response, we evaluated the approach of additonal model structure to limit unfaithfulness, where structure was added generator-internally (e.g. *copy-gate*), or existed alongside the generator (e.g. a semantic fidelity classifier in the DataTuner). We have finally noted the potential of enhancing the language model through cross-dataset training, as well as manually increasing dataset fidelity; both of which can contribute to increasing semantic accuracy without the construction of a more complex model.

2.3 Measuring Semantic Faithfulness

Having presented methods to avoid semantic diversions in Subsection 2.2.5 from a theoretical perspective, we also require the skill to assess a model's semantic accuracy in practice.
Given a set of structured data, we output a text sequence, whose semantic content should mirror all and only information from the structured data, i.e. the semantic *union* of reference output and generated output equals their *intersection*. While complete semantic overlap is a straightforward concept to humans, it is difficult to measure in practice, as there exists no completely reliable map from text to an abstract meaning representation, where the meaning representation fully and unambiguously captures every aspect of semantic content, and only that. Thus, all meaning-related assessment will be imperfect to some extent.
This section will present some basic tools for assessing semantic fidelity, of various levels of complexity & accuracy. Of course, the option of human judgement always exists, but we will for now focus on system-integrated faithfulness evaluation, the less straightforward assessment method.

2.3.1 Automatic metrics for measuring faithfulness

In contrast to neural models assessing the degree of agreement between generated sentence and reference sentence or data source, the comparison of the former can be hand-engineered. Such hand-engineered evaluation methods result in *automatic metrics*, a few of which we will described below. Hand-engineered methods typically only apply minor transformations (e.g. lowering, potentially stemming) to the metric input, allowing for little flexibility and grace in declaring generations as "faithful".

Traditionally, the overlap between reference and generation is automatically measured through a comparison of n-grams. For instance, BLEU (Bilingual Evaluation Understudy, Papineni et al., 2002) computes a score based on n-gram precision for $n \in \{1, 2, 3, 4\}$:

$$p_n = \frac{\sum_{G_n \in \mathrm{gen}} \mathrm{cnt}_{\mathrm{clip}}(G_n)}{\sum_{G'_n \in \mathrm{gen}} \mathrm{cnt}(G'_n)}, \text{ where } \mathrm{cnt}_{\mathrm{clip}}(G_n) = \begin{cases} \mathrm{cnt}_{\mathrm{gen}}(G_n) & \text{if } \mathrm{cnt}_{\mathrm{gen}}(G_n) < \mathrm{cnt}_{\mathrm{ref}}(G_n) \\ \mathrm{cnt}_{\mathrm{ref}}(G_n) & \text{if } \mathrm{cnt}_{\mathrm{gen}}(G_n) \geq \mathrm{cnt}_{\mathrm{ref}}(G_n) \end{cases},$$

for G_n an n-gram.

The final BLEU-score is then the exponential of the weighted average of n-gram precisions, factored by a brevity penalty term[16] $\beta \in [0, 1]$:

$$\mathrm{BLEU}(\mathrm{gen}, \mathrm{ref}) = \beta \exp\left(\sum_{n=1}^{N} w_n p_n\right)$$

A similar approach, but recall-oriented, is taken by the ROUGE metric (Recall-Oriented Understudy for Gisting Evaluation; Lin, 2004), whose default version considers for every example the fraction of the reference n-grams that are remembered by the system, i.e. appear in the generation.
In comparing n-grams, both BLEU and ROUGE will only recognise semantic overlap if the meaning is expressed with very similar wording. To see why this is problematic, consider the BLEU column in Table 1.

Since purely n-gram based scores are blind to rephrasing, for example using synonyms, or using a stem in a different form, and decrease with changes in word-order, they are less reliable, and yet widely used.

METEOR (Lavie and Agarwal, 2007), a more complex n-gram based metric, also takes into account stemmed versions of words, as well as synonymy.
In particular, METEOR finds a 1-1 mapping ("alignment") between reference words and generated words, where words can be mapped to each other if they agree exactly, their stems agree, or they are synonymous. Stems are extracted using the PorterStemmer[17], syononymy is detected according to affiliation with the same "synset" in WordNet[18]. The METEOR score is then computed as the parameterised harmonic mean of precision $P = \frac{\#\text{mappings}}{\#\text{generation\_words}}$ and recall

[16]compares reference - & generation length, see Papineni et al., 2002 for details
[17]`https://www.nltk.org/_modules/nltk/stem/porter.html`
[18]`https://wordnet.princeton.edu/`

semantic overlap	reference-hypothesis pair	B	M	M − B
unrelated	ref: *My friend had spaghetti for lunch.* hyp: *Internet criminality has risen in the UK.*	0	0	0
vague	ref: *John waited patiently outside the doctor's office.* hyp: *Waiting times in UK surgeries have doubled.*	0	7.14	**7.14**
synonymy	ref: *Max* ***sat a hard exam*** *today.* hyp: *Max* ***took a difficult test*** *today.*	10.68	80.67	**69.99**
word order change	ref: *Tennis is my favourite sport.* hyp: *My favourite sport is tennis.*	45.18	89.2	**44.02**
same stem	ref: *Max* ***is eating*** *spaghetti with tomato sauce.* hyp: *Max* ***eats*** *spaghetti with tomato sauce.*	45.48	85.35	**39.87**
same	ref: *Max eats spaghetti with tomato sauce.* hyp: *Max eats spaghetti with tomato sauce.*	100.0	99.77	-0.23

Table 1: ***Comparison of metric sensitivity in detecting semantic overlap,*** *trialling at different levels of semantic overlap (first column).*
The table shows automatic metric responses for BLEU (B) & METEOR (M). Differences larger than 5 *are highlighted (last column).*
When both sentences are either the same or unrelated, both metrics correctly assign 0% *and* 100% *(up to round-off errors), respectively. In more gradual settings, the verdict is less unanimous:*
*The n-gram focused BLEU-metric does not pick up on vague semantic overlap if the two sentences have no words in common (*0%*). In contrast, METEOR appropriately assigns a small score to examples with even vague semantic overlap (*7.14%*), due to its knowledge of semantic fields ("synsets"). Similarly, BLEU completely misses the shared meaning in synonymy, only assigning points for the actually overlapping unigrams (*10.68%*). METEOR however assigns ca.* 70%*, clearly indicating the semantic overlap present. When two identical sentences only differ in word order, METEOR is barely affected (ca.* 90%*), whereas the number of recognised higher-order n-grams decreases in BLEU, unnecessarily lowering the BLEU-score. BLEU also fails to recognise that "eats" and "is eating" have the same meaning, unlike METEOR, which recognises the common stem "eat" and still assigns approx.* 85%*.*

$R = \frac{\#\text{mappings}}{\#\text{reference\_words}}$:

$$\begin{aligned} \text{METEOR}(\text{gen}, \text{ref}) &= (1 - \text{pen}) \cdot \frac{P \cdot R}{\alpha \cdot +(1-\alpha) \cdot R} \\ &= (1 - \text{pen}) \cdot F_{mean} \end{aligned} \tag{4}$$

A change in word order can be penalised through

$$\text{pen} = \gamma_1 \cdot (\frac{\#\text{chunks}}{\#\text{mappings}})^{\gamma_2}, \tag{5}$$

which measures word order similarity between reference and generation by considering the smallest number of "chunks" into which the generation output can be divided, where a chunk is defined as a string of generated words that are contiguous in the reference.
METEOR correlates with human similarity judgement significantly more strongly than BLEU. Table 1 confirms this empirically. All METEOR scores in Table 1 were computed with default parameter settings $\alpha = 0.9$; $\gamma_1 = 0.5$; $\gamma_2 = 3$).

In line with the strong correspondence between METEOR and human judgements, METEOR captures semantic similarity between reference sentence and generation (i.e. "hypothesis") more independently of the sentence phrasing (choice of words, word order). BLEU's assessment of semantic similarity is however strongly contingent on similarity in the surface realisation of the underlying meaning. Hence, METEOR is the superior sentence-meaning comparison tool, but we will maintain BLEU as well due to its traction among the research community.

Many n-gram based metrics (BLEU, ROUGE, METEOR), which were originally invented for Machine Translation, rely on the reference output being the ideal output. For instance, the German sentence *Die Katze saß auf der Matte* is the optimal translation of the English sentence *The cat sat on the mat*. However, this assumption is rarely true in our context, since in the vast majority of data-to-text datasets, the reference sentence associated with a set of structured data is not faithful to the data, i.e. we have a lack of high-quality datasets. The lack of trustworthiness in data-to-text reference sentences calls for a metric which can compare a generated sentence to the source data directly. In fact, Dhingra *et al.* have created such a metric, especially for measuring the quality of data-to-text systems, instead of borrowing from Machine Translation. which compares the generation to the data source, instead of the potentially unfaithful reference sentence. The suggested metric PARENT (Precision And Recall of Entailed N-grams from the Table, Dhingra et al., 2019) compares the n-grams in the generation to the reference sentence as well as the data source. PARENT is the F-score of what the authors call *entailed precision* and *entailed* recall:

$$PARENT = 2 \cdot \frac{P^{\text{Ent}} R^{\text{Ent}}}{P^{\text{Ent}} + R^{\text{Ent}}},$$

where P^{Ent} & R^{Ent} are the geometric mean of the sub-quantities P_n^{Ent} & R_n^{Ent}, for the particular n-gram sizes.
Let $Pr(G_n)$ be the fraction of words in n-gram G_n that occur in the table. Then the entailed precision of a generated sentence ("hypothesis") for n-gram size n is defined as

$$P_n^{\text{Ent}} = \frac{\sum_{G_n \in \text{hyp}} Pr(G_n) + (1 - Pr(G_n))\mathbb{1}(G_n \in \text{ref})}{\sum_{G_n \in \text{hyp}} 1},$$

rewarding n-gram words occurring either in the data source or in the reference. The entailed recall allows a weighting between recall from the table vs. from the reference. It is a geometric average of data-recall $R_n^{\text{Ent}}(d)$ and reference recall $R_n^{\text{Ent}}(r)$, weighted by λ:

$$R_n^{\text{Ent}} = R_n^{\text{Ent}}(r)^{(1-\lambda)} \cdot R_n^{\text{Ent}}(d)^{\lambda}$$

By default, $\lambda = 0.5$.
PARENT is thus an n-gram based metric, tailored to data-to-text generation, by not fully relying on the reference sentence, and taking into account the data source instead.

Below is an example of PARENT's attention to the data source, even if the reference says otherwise:

- **data:** `<subject> Albert Jennings <predicate> birth place <object> New York`
- **reference:**
  `Albert Jennings was born in New York, United States. He died in 1946, after short illness.`

- **generation:** `Albert Jennings was born in New York.`

Although the reference is highly hallucinative, mentioning the US, the death, and a short illness, the generation is awarded a PARENT score of 79.93. When both reference and generation capture the data perfectly, PARENT is 96.41 in this example. The score of 79.93 is still overall positive, indicating good overlap between generation and data.
When flipping the roles of reference and generation (i.e. the generation is now hallucinative), we only obtain 47.99.
This example has thus verified PARENT's attention to the data source (alongside the reference).

2.3.2 Semantic evaluation with neural models

Above, we have seen how automatic metrics can be used to rate semantic accuracy of generated text. Their behaviour is essentially n-gram based, making them transparent on the one hand, but inflexible on the other. We will now turn to using neural models to assess semantic faithfulness, where we hope to trade some transparency for reliability.
In particular, the semantic classifier approach, where linguistic pairs are labelled according to semantic overlap, is a common form of employing neural models for assessing semantic accuracy, as suggested by the existence of the Stanford NLI Corpus (MacCartney and C. D. Manning, 2008), which labels sentence pairs either *entailment*, *contradiction*, or *neutral*. We will adopt this approach. It is also used in the semantic fidelity classifier of the DataTuner, whose architecture we have largely adopted.
One simple low-cost approach would be to distinguish between *accurate* and *inaccurate* generations, given the data. However, this division would tell little about the nature of the error, making it impossible to change the setup/model in an advantageous way to possibly avoid that error. Instead, we could focus on one particular error type, to learn more about the model's rationale:
Since hallucination has been the most perplexing semantic error type, having been investigated extensively in the literature, we will focus our scope on hallucination. Now given a data-text pair, the simplest classification scenario would be *hallucination* vs. *not hallucination*. This division would however force very disparate generation types into the *not hallucination* class (e.g. sentences containing value errors together with fully accurate sentences).

To avoid the most contradictory class mix-up described above, accurate generations would have to be separate from any faulty generations, forcing us to abandon the binary classification setting, in favour of a *accurate-hallucination-other_error* setting.
An even more ontologically sensible division is implied by the Stanford NLI, where sentence pairs are labelled as *contradictory*, *neutral* or *entailment*. It is possible to define several error classes, given the ability to detect which information in the generation can be inferred from the source and vice-versa. For instance:

- a bidirectional contradiction would constitute a *value error*,
- the source sentence being entailed by the generation and the generation being neutral towards the source would constitute a *hallucination*
- an *omission* is equivalent to a hallucination, when source and generation are swapped
- a bidirectional entailment would classify as *accurate*

While the above setup works from basic principles, and thus might be easier to grasp for any classification model, one frequent data-to-text generation error type remains to be covered by this model: *repetition*. Even under the strict examination of entailment, repetition can go unchecked, since it would not change the set of information present.

This brings us to the most extensive classification scenario, suggested in Harkous, Groves, and Saffari, 2020 in the DataTuner, which distinguishes 5 classes (*accurate*, *hallucination*, *omission*, *value error*, *repetition*).
We will build on this framework, experimenting with reduced class-settings to question the necessity of such a number of classes, when we are only interested in the presence of one of them (*hallucination*).

One important advantage of training a model to assess faithfulness is that it can be trained on any form of data, i.e. comparing the data source to fluent text is no problem. This used to pose a hurdle for automatic metrics, who were very inflexible w.r.t. switching between data representation mode (i.e. text vs. linearised data), due to their reliance on n-grams.

In this Subsection, we have seen the two entirely different approaches (neural vs. hand-engineered) that can be taken in evaluating data-to-text output. While automatic metrics are transparent, they can be used for comparability to other systems. The performance of neural models in assessing semantic accuracy is very unpredictable, due to the various design choices the researcher can make.

2.4 Data-to-text corpora

We have discussed how the generation quality of a data-to-text system can be assessed. As opposed to neural models, automatic metrics for data-to-text generation judge a generation's quality by comparing the generation to the data/reference according to a hard-engineered set of rules. Thus in particular for automatic metrics, it is clear that the metrics' meaningfulness depends on the dataset in many ways.

The most obvious dependency arises from the semantic fidelity among the dataset, i.e. how closely the references themselves cover what is mentioned in the data source. Automatic metrics tend to measure overlap between generation and reference, so that their reliability depends directly on the reliability of the references.

The most widely-used graph-to-text corpus is WebNLG, where each example consists of 1-7 subject-predicate-object triplets (see Section 2.2.2), spans 15 topics, of which 10 are represented in the training set. WebNLG is commonly evaluated with BLEU and METEOR (e.g. Harkous, Groves, and Saffari, 2020; W. Chen et al., 2020; Ribeiro et al., 2020), and sometimes with ROUGE. Precision metric BLEU can potentially pick up on hallucinations, since any statement made in the generation that is not supported by the reference will decrease precision. Of course BLEU is thus only effective in measuring fidelity to the data source if the reference sentence mirrors the data well. ROUGE however, the recall counterpart of BLEU, is less meaningful in our research question, since the information recalled in the generation are unaffected by potentially hallucinated extra-information in the generation. We will thus avoid using ROUGE. Taking the oblivion of recall to an extreme, the system could learn to generate a minimum amount of information from the data, so that most facts are omitted. Such outputs would still achieve high precision. In a holistic evaluation of a data-to-text system, recall thus still plays a subordinate role. PARENT is an F1-score, evaluating P^{Ent} and R^{Ent}, so that a recall-containing metric is available. Given that R^{Ent} takes into account the structured data too, R^{Ent}

might even be preferred to ROUGE, which considers just the n-grams of the potentially noisy reference.

The previously mentioned dialogue dataset ViGGO (Juraska, Bowden, and Walker, 2019) is a smaller dataset restricted to questions about gaming user experience. The data hold a dialogue act each, which then applies to 1-8 slot-value pairs (see Section 2.2.2). However, only 9 dialogue acts and 14 slot names are available, limiting semantic diversity among the corpus. In turn though, ViGGO was created with the aim of total semantic faithfulness, making it a worthwile choice still in tackling hallucination.
Data-to-text systems trained on ViGGO are commonly assessed with METEOR too (Juraska, Bowden, and Walker, 2019; Harkous, Groves, and Saffari, 2020), alongside BLEU and PARENT.

For the WikiInfo2Text dataset (S. Chen et al., 2019), the widely-used WikiBio dataset has been merged with 20 self-collected categories from Wikipedia, such as *UK_place*, *Book*, *Automobile*, *Military_conflict* & *French_commune*. Originally, WikiInfo2Text was created to investigate the the inclusion of external knowledge from WikiData in the data source (see example in Listing 3). WikiInfo2Text has not gained much traction yet, as without the exclusion of the external knowledge, sentences can hardly be faithful to the data source. Due to its topic variability, we will use WikiInfo2Text, removing WikiData information.

Listing 3: **A WikiInfo2Text example in raw form.** (Lengthy entries cut with three dots.)
Many data entries seem cryptic (e.g. the field `KB_id_tuples`) and/or irrelevant (e.g. `KB_str_tuples`) to the reference text.

```
{
  "Sentences": [
    "The soundtrack to \" Raiders of the Lost Ark \" was released
        by Columbia Records in June 1981 .",
    "The music was composed and conducted by John Williams , and
        performed by the London Symphony Orchestra ."
  ],
  "name": "Raiders of the Lost Ark",
  "artist": "\" Indiana Jones \"",
  "cover": "Raiders soundtrack.jpg",
  "released": "June 1981",
  ...
  "KB_id_tuples": [
    [ "genre", "Q217199", "P279", "Q2188189" ],
    ...
    [ "next_album", "Q6023297", "P31", "Q482994" ]
  ],
  "KB_str_tuples": [
    [ "genre", "soundtrack", "subclass_of", "musical_work" ],
    ...
    ["instance_of", "album" ]
  ]
}
```

In this Section, we have reviewed basic techniques of modern-day Natural Language Generation, and data-to-text generation in particular, in terms of datasets, data-preprocessing, training and evaluation. We have explored the additional challenges mainly arising from noisy data-to-text attests, creating a need for special automatic metrics and to some extent even measures to limit the omnipresent phenomenon of hallucination. In the next Section, we will turn to describing and justifying the particular approach selected for this thesis.

3 Approach

This Section aims to clarify the conditions under which the experiments will be carried out, mainly for reference and replicability. It contains practical details of models, datasets and metrics, as well as departures from previously published approaches.

We will begin by describing the system architecture and its components in Subsection 3.1, followed by an elaboration on the changes applied to the raw datasets (Subsection 3.2). In Subsection 3.3, we will present our 'cheap and cheerful' corpus faithfulness boosting procedure, which allows us to manually increase semantic overlap between data and text in often noisy data-to-text corpora. After giving some practical details on training techniques in Subsection 3.4, we will finally turn to evaluation, explaining how we assess our generator's quality w.r.t. semantic accuracy in Subsection 3.5, taking into account automatic metrics, human judgement and a RoBERTa-based semantic fidelity classifier.

3.1 Generator-Classifier architecture

The challenge of faithful data-to-text generation can be tackled in many ways, as described in Background subsection 2.2.4. We have decided for the approach taken by Harkous, Groves, and Saffari, 2020 in their DataTuner, which distributes responsibility to two models:
The *generator* will be concerned with turning a linearised abstract meaning representation of some information (data) into fluent text, whereby we will endeavour to make its generations as faithful as possible to the data source given. The *classifier* is then an independently trained system, which we will employ to judge the generations' faithfulness to the data, by assigning one out of a fixed number of semantic relationship labels.

3.1.1 T5 Generator

In our experiments, the T5 was presented with a string representing the data source in a manner prescribed by the linearisation technique. For example[19]

```
webnlg: <subject> Bhajji <predicate> country <object> India ;
<subject> India <predicate> leader name <object> Mahajan </s>,
```

[19]A complete list of linearisation examples for every dataset will be given in 3.2.2.

where `</s>` is the mandatory T5 end-of-sequence token. The T5 was then asked to generate a sentence, ideally in this case `Bhajji is in India, where the leader is Mahajan.`
The generator was trained on datasets *WebNLG*, *WikiInfo2Text* and *ViGGO* (with ViGGO's repetitions removed during cleaning).
For all experiments, unless stated otherwise, we used the `t5-base` configuration, which marks the second smallest among the released versions. Current options on the HuggingFace model hub[20] are `small`, `base`, `large`, `3b` & `11b` (size in ascending order).
In its attention mechanisms, `t5-base` uses key/query/value vector dimension $d_{k,v} = 64$, the size of $d_{ff} = 3072$ for intermediate feed forward layer in each T5-block, the size of $d_{\text{model}} = 768$ for encoder layers and the pooler layer, 12 hidden layers in the Transformer encoder, and 12 attention heads in each Transformer encoder attention layer. Further technical details can be found in the standard `t5-base` configuration file at `https://huggingface.co/t5-base/tree/main`.
Unless stated otherwise, the T5 was trained for one epoch, with batch size 4, using the Adafactor optimizer (Shazeer and Stern, 2018), with learning rate 0.001[21].
At the cost of speed, we had to enable gradient checkpointing for the one experiment with `t5-large`, to reduce memory uptake.

3.1.2 RoBERTa Semantic Fidelity Classifier

The attentive reader might have noticed that RoBERTa is actually a sequence-to-sequence model, but is used for outputting a class label here. In their repertoire of sequence classification models, HuggingFace have a model `RobertaForSequenceClassification` listed, which is a standard RoBERTa with an additional classification head on top. The RoBERTa will thus use its encoder to process the data source, its decoder process the text given the data, and finally, a classification head will assign a class based on the decoder output.
Thus, one RoBERTa training example from the ViGGO would consist of three parts (see Listing 4).
RoBERTa was trained on *ViGGO* (`viggo` & `viggo-clean`) and WebNLG. The semantic-fidelity-classification versions of ViGGO & WebNLG that we use have

[20]`https://huggingface.co/models`
[21]Find more parameter settings in the code on `https://github.com/tishaAnders/d2t_generator`.

been manally assembled by Harkous, Groves, and Saffari, 2020. For each class label, Harkous *et al.* hand-generated examples through sheer string manipulation.

Listing 4: **A pre-processed ViGGO example to be fed to RoBERTa semantic fidelity classifier.** The `label`-field carries `4`, which in our case stands for *hallucination*. In this case, the text enquires about indie adventure games, when it should only ask why the user likes shooter games.
Fields `label`, `data` & `text` are handled by the classification head, the RoBERTa encoder and the RoBERTa decoder, respectively.

```
{
    "label" : 4,

    "data" : "<request_explanation> request explanation ( <rating>
        rating: [ excellent ], <genres> genres: [ shooter ]> )",

    "text" : "Have you ever tried an indie adventure game, like The
         Vanishing of Ethan Carter for example? What is it about
        shooters that you find so great?"
}
```

The `RobertaForSequenceClassification` model is by default configured with the values suggested by the developers of RoBERTa (Yinhan Liu et al., 2019). `RobertaForSequenceClassification` can be used with either `roberta-base` or `roberta-large`, both of which experimented with. We left the suggested configurations unchanged, due to RoBERTa in fact being an instance of BERT, with carefully selected parameter configurations (see Section 2.1.4).

3.2 Preparing the datasets

Having presented the technical details of the model architecture in the previous subsection, we will now look at the data the models have been trained with, and in particular the modifications made compared to the raw corpora. To prepare our datasets for training and testing, we clear out undesirable elements that occur in a repetitive fashion, either of structural or of substantial nature. We then adjust the dataset to our purpose - restructuring of or picking from the data might be necessary. Additionally, new structure might be added (e.g. special tokens) to facilitate the model's understanding of the data.

Both ViGGO and WebNLG were retrieved from their official sites, `https://`

`nlds.soe.ucsc.edu/viggo` & `https://github.com/ThiagoCF05/webnlg.git` (v1.4; Castro Ferreira et al., 2018, Gardent et al., 2017) respectively, and were then adjusted with Harkous *et al.*'s pre-processing scripts.
The outcome of this pre-processing, as well as further steps taken, will be described below.

3.2.1 Cleaning

The attentive reader might have noticed in Listing 4 the repetition of dialogue acts and slot names in the data source, e.g.

```
<recommend> recommend ( <name> name:  [ The Wolf Among Us ], <genres>
genres:  [ adventure, point-and-click ], <available_on_steam> available
on steam:  [ yes ]>
```

Given that Harkous *et al.* used this version of ViGGO examples to train their DataTuner, we were sceptical of simply removing the repetitions (red), and kept two versions of ViGGO instead (`viggo` & `viggo-clean`). It might have been that RoBERTa learns best from this syntax. However, after close investigations into the origin of the repetitions, they had been introduced by Harkous *et al.*'s standardised pre-processing procedure.
The RoBERTa semantic classifier was trained on both sets, however when ViGGO is mentioned in context of the T5 generator, we assume `viggo-clean`.
The WebNLG dataset, which has been widely used (e.g. in the WebNLG Challenges[22]), had no inconsistencies, apart from minor ones, which would immediately catch the reader's eye.
WikiInfo2Text required intensive cleaning, however not of structural nature, but concerning the content. Due to its general applicability, the WikiInfo2Text semantic cleaning procedure will be laid out in 3.3.

3.2.2 Adjusting the datasets to our purpose

After some rudimentary cleaning described in 3.2.1, the next step is to make the training examples easier to process and learn from for our models.
In our case, this step comprises linearisation, the addition of end-of-sequence tokens, in combination with semantic faithfulness boosting (for the WikiInfo2Text

[22]`https://webnlg-challenge.loria.fr/challenge_2020/`

Listing 5: **A pre-processed WikiInfo2Text example**, without a faithfulness label, as we do not train the semantic classifier on WikiInfo2Text.

```
{
    "data" : "<name> H Is for Homicide && <author> Sue Grafton && <
        series> Alphabet Mysteries && <genre> Mystery && <pages>
        256 pp " first edition " && <preceded_by> G Is for Gumshoe
        && <followed_by> I Is for Innocent && <articletitle> H Is
        for Homicide </s>",

    "text" : "H Is for Homicide is the eighth novel in Sue Grafton'
        s Alphabet series of mystery novels and features Kinsey
        Millhone, a private eye based in Santa Teresa  California."
}
```

Listing 6: **A pre-processed WebNLG example**, labelled as semantically faithful.

```
{
    "label" : 0,

    "data" : "<subject> Adam Holloway <predicate>  alma mater <
        object> Magdalene College, Cambridge ; <subject> Adam
        Holloway <predicate> birth place <object> Kent </s>",

    "text" : "Adam Holloway was born in Kent and attended Magdalene
         College in Cambridge."
}
```

corpus).
The procedure of linearising structured data into a string has been described empirically for our datasets in 2.2.2. For completeness, we will list one representative (cleaned & adjusted to purpose) training example for each corpus included (Listings 5, 6 & 7)

3.3 Dialling semantic fidelity among the datasets

As described in Background Subsection 2.2.4, the semantic content of data and text in data-to-text corpora is often highly divergent, which is believed to aggravate hallucinative behaviour in data-to-text models.
Previous work typically trains their models on different datasets, stating their assessment of the dataset's semantic accuracy, and then evaluate their model output's faithfulness in the context of the dataset's semantic accuracy level. Commonly, two broad classes of data-to-text corpora exist: *web-crawled* (e.g. WebNLG, WikiInfo2Text) vs. *constructed aiming for complete faithfulness* (e.g.

Listing 7: **A pre-processed `viggo-clean` example**, labelled as semantically faithful.

```
{
    "label" : 0,

    "data" : "<give_opinion> ( <name>: [ Undertale ], <rating>: [
        excellent ], <has_multiplayer>: [ yes ]> ) </s>",

    "text" : "I think Undertale was truly excellent. It also has a
        multiplayer-mode."
}
```

ViGGO, ToTTo by Parikh et al., 2020). While the performance on web-crawled vs. on supposedly faithful datasets reveals the model's susceptibility to dataset noise to some extent, the comparison is affected by other factors too, such as lexical complexity differences (measured by Harkous, Groves, and Saffari, 2020), cleanliness and suitability of linearisation technique.
A unique feature of this thesis is that we ruled out all other contributing factors, by isolating data noise as a single parameter: For WebNLG and WikiInfo2Text, we created instances of the same dataset that only differ in their semantic faithfulness level.

As mentioned in 3.2.2, Harkous *et al.* have handcrafted a WebNLG training set for their semantic fidelity classifier, including faithful and unfaithful examples, which are also labelled as such. From this labelled dataset, we retrieved `webnlg-consist` and `webnlg`, where the former considers only examples labelled as semantically accurate, and the latter considers *all* examples.
The same procedure was used for ViGGO, to obtain `viggo-consist` and `viggo`.

With WikiInfo2Text, a corpus Harkous *et al.* do not use, we performed a semantic cleaning procedure to the originally downloaded corpus, to obtain `wikiinfo-clean`, in contrast to original `wikiinfo`. A descriptive flowchart of the cleaning procedure can be found in Figure 2.

Due to WikiInfo2Text being crawled from Wikipedia, meta-data are also present. Thus, any slot-value pair with the slot name containing at least one out of

```
["KB", "image", "website", "homepage", "caption", "coordinates", "postcode"]
```

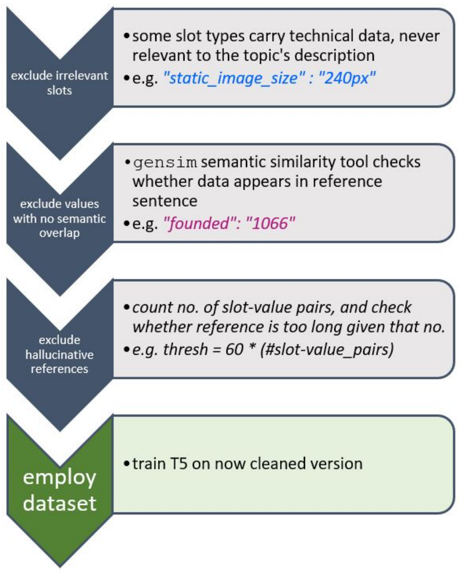

exclude irrelevant slots
•some slot types carry technical data, never relevant to the topic's description
•e.g. "static_image_size" : "240px"
exclude values with no semantic overlap
•gensim semantic similarity tool checks whether data appears in reference sentence
•e.g. "founded": "1066"
exclude hallucinative references
•count no. of slot-value pairs, and check whether reference is too long given that no.
•e.g. thresh = 60 * (#slot-value_pairs)
employ dataset
•train T5 on now cleaned version

Figure 2: **Cheap and cheerful semantic fidelity increasing**, applied to the WikiInfo2Text corpus.

would be removed from the training example. In the same vein, slot-value pairs with values unrelated to the text were removed. To measure relatedness, a semantic similarity checker was given a slot-value pair and the reference sentence, upon which it returned a relationship strength $r \in [0, 1]$. Slot-value pairs were kept if $r > 0$, since longer sentences would automatically result in lower r, so that any $r > 0$ would imply a real semantic overlap. The semantic similarity checker is based on Latent Semantic Indexing (Deerwester et al., 1990), which picks up not only on synonyms, but also on concept-relatedness, by associating terms that occur in similar contexts. The `gensim` instantiation[23] `LsiModel` was used. Beyond targeting specific elements due to their content, we can also infer hallucinative behaviour from looking at macro-factors: Given the number of data points (slot-value pairs or knowledge triplets) n_{data_pt}, a maximum threshold τ for the number n_{text_char} of characters in the text can be determined, so that any example included must fulfil

$$n_{text\_char} \leq \tau \cdot n_{data\_pt}$$

The above constraint aims to rule out hallucinative examples, where the text-

[23]`https://radimrehurek.com/gensim/models/lsimodel.html`

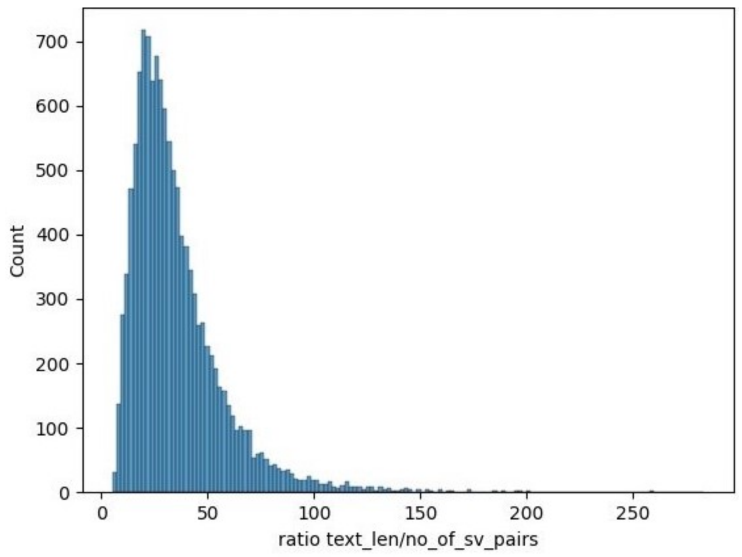


Figure 3: **Empirical distribution of text-to-data length** in the WikiInfo2Text corpus. The heavy positive skew justifies cutting off any examples with ratio larger than $\tau = 60$.

to-data ratio is typically increased due to added text with no grounding in the data.
τ was found empirically, from considering a histogram of text-to-data ratios among the dataset (see Figure 3). Choosing τ implied a trade-off between semantic faithfulness and dataset size.

τ was set to 60, allowing within one example an average number of 60 characters per data point, or approx. 10 words[24]. This left us with 89.04% of the dataset, i.e. 1346 out of 12279 examples were excluded.

This Subsection has described how for ViGGO, WebNLG and WikiInfo2Text, we obtained versions of the datasets that vary in semantic fidelity. For ViGGO and WebNLG, we obtained a consistent and a noisy version by using a subset of Harkous *et al.*'s semantic classification dataset. For WikiInfo2Text, irrelevant slot-value pairs were deleted (based on their slot name or relevance of the slot

[24]The average word length in English is $n_{text\_char} = 4.7$ letters, but including spaces and punctuation we arrive at $n_{text\_char} \approx 6$.

value), and examples with data-to-text ratio above $\tau = 60$ were excluded.
The next Subsection presents practices worth noting for replicating the training process.

3.4 Lab practice: Training techniques

While in Section 3.4.1, performance-relevant techniques will be laid out, the subsequent part deal with implementation issues, such as a bug-fix in the HuggingFace T5 configuration file (3.4.3).

3.4.1 Cross-training & alien datasets

At the end of all experiments, the system's quality will be judged upon the generator's faithfulness and the classifier's accuracy, all tested on WebNLG. The generator will generate from the abstract meaning representations in `webnlg-consist` and the semantic classifier will classify those generations. Therefore, WebNLG is our target corpus we will finally evaluate on.
Yet we will investigate whether both T5 generator and RoBERTa classifier can benefit from seeing examples from *alien datasets* (WikiInfo2Text, ViGGO). Datasets do differ in meaning representation, according to their linearsiation technique. However, the idea is that if the model is clever enough to overcome the structural differences of the dataset-specific meaning representations, it might profit from seeing more examples.
In the following, we will lay out the exact joining procedure for the multiple datasets.

From the datasets `webnlg-consist`, `viggo-consist` & `wikiinfo-consist`, as well as their inconsistent counterparts, all described in 3.3, it is possible to construct *joint datasets*. The joining of two datasets `dsA` and `dsB` is done by first attaching a dataset identifier to each example's linearised data string, i.e. "`dsA: `" or "`dsB: `". The dataset identifier is inspired by task-identifying *prompts* (Li and Liang, 2021), in modern-day NLP tasks often attached at the beginning of the input string (e.g. "`summarize: `" or "`translate-eng-to-fr: `"). Such prompt prefixes allow the language models to learn several tasks at once, while attending to their differences. We therefore treat different datasets as different tasks here. Once every training example from both datasets has a dataset identifier prefix, the datasets will be merged by simply listing all examples from `dsA`

and then from `dsB`, and mixing up the order afterwards.
We will join up to three datasets in this manner, never including the consistent and the inconsistent version of the same dataset though (this would result in examples appearing twice).

3.4.2 Training RoBERTa-large

Initially we planned on executing every semantic fidelity classification experiment with both the base version and the large version of RoBERTa. However, `roberta-large` turned out to require very careful tuning of the batchsize and the learning rate. Otherwise, `roberta-large` would output the same label for each data-text pair.
It was thereby beneficial to pick the maximum batch size given the GPU capacity, and a very small learning rate. We picked batch size 4 for WebNLG and 8 for ViGGO, and learning rate 3×10^{-6} for WebNLG and 5×10^{-6} for ViGGO. We used the HuggingFace AdamW Optimiser implementation[25].
Sometimes even those precautions would not suffice, so that not all experiments could be carried out on both `roberta-base` and `roberta-large`.
While `roberta-large` did on average provide a small classiification accuracy increase, we decided to focus on `roberta-base` instead, as the best-performing model turned out to be a `roberta-base`.

3.4.3 Bug-fix: Configuring T5

When implementing the `t5-base` for text generation using the original HuggingFace instructions, the T5 always stopped generating at exactly 20 characters for any example, making it futile for any experiments. Initially, my supervisor and I suspected an optimisation-related problem, such as not finding a suitable minimum. However, numerous other users seemed to have encountered this problem. In many cases, the HuggingFace team had responded to posts, but none of the tricks suggested seemed to help. After 3 days of investigation, I discovered that the advice given by the HuggingFace team would never work for people with exotic tasks (i.e. not summarisation or translation). Much of the advice given, such as using a different `generate` function or including a `max_len` parameter in the `generate` function, which later turned out to be obsolete, did not make a difference. Once I configured a new task `text-generation` in the T5

[25] `https://huggingface.co/transformers/main_classes/optimizer_schedules.html#transformers.AdamW`

`config` file, with a `max_len` parameter too, the `max_len` instruction was suddenly understood by T5. However, the existence of a task configuration, where a universal `max_len` parameter value would be set, was neither mentioned as a condition in the Huggingface documentation, nor in the forums maintained by the HuggingFace team.

3.5 Measuring generator quality

The quality of the T5 generator is affected by two factors: *faithfulness* and *fluency*.
Faithfulness has been mentioned throughout this report, and is present if *all and only* information from the data source are present in the generated sentence. Repetitions of semantic content do not add to the fact pool and thus do not harm faithfulness.
Fluency is a concept detached from semantics, purely judging whether the order in which the words appear is correct. For instance, "*The flute killed the stone*" is a fluent sentence.
With the evaluation techniques described below, we will attempt to capture both fluency and faithfulness.

3.5.1 Automatic metric scores

As laid out in Background Subsection 2.3.1, we have selected three automatic metrics:

1. **PARENT:** A metric specifically for data-to-text tasks, considering a quantity related to n-gram precision and recall, comparing the generation to both the reference and the data source.

2. **METEOR:** A more complex n-gram based metric, which understands more vague semantic relationships due to its understanding of stems & synonyms

3. **BLEU:** The n-gram precision of generation in the reference, borrowed from translation

Parameter values for each metric are listed in Table 2.
The `nltk` BLEU implementation[26] we use returns a final score of 0 when for any $n \in \{1, 2, 3, 4\}$ no n-grams can be found, disregarding the contribution

[26]`nltk.translate.bleu_score.sentence_bleu`

metric name	param. settings
PARENT	$\lambda = 0.5$
METEOR	$\alpha = 0.9$; $\gamma_1 = 0.5$; $\gamma_2 = 3$
BLEU	all ns have equal weight ($w_n = 0.25$ for $n \in \{1, 2, 3, 4\}$)

Table 2: **Parameter values used in automatic metrics**. All values correspond to the default.

of any lower-order n-grams. This effect can be mitigated by using smoothing functions.
We used smoothing method[27] 3, which assigns to any p_n that is 0 a small quantity $\frac{1}{2^n}$. In our case, the lowest order n-gram that could not be found had $n = 3$.
Since all metrics to some extent rely on spotting n-grams in the reference, their precision in detecting fluency is high, provided the reference is fluent (which mostly is the case). High scores would correlate with fluent examples.
However, the automatic metrics might not detect fluency if few n-grams overlap. This makes a human evaluation indispensable in measuring overall fluency.

As discussed in 2.3.1, the ability of automatic metrics to detect faithfulness given in PARENT, and depending on the reference quality, also in METEOR. Being purely a precision measure, BLEU technically decreases with hallucination, however mostly suffers from inflexibility in measuring semantic overlap, due to considering just n-gram overlap.

3.5.2 Small subsample human evaluation

Due to the lack of reliability in detecting both faithfulness and fluency, which we have identified in 3.5.1, human judgement should be the most reliable verdict in both aspects. However, due to the strong focus required to detect minor inconsistencies in fluency & faithfulness, only a small subsample of generations can be evaluated. With every judgment, we also judge whether the reference sentence was fluent and faithful. Furthermore, due to our focus on hallucination, we ask reviewers to indicate whether extra information was present in case they

[27]`https://www.nltk.org/_modules/nltk/translate/bleu_score.html#SmoothingFunction.method3`

label the generation as unfaithful.

3.5.3 Classifying generative faithfulness with RoBERTa

Finally, in order to obtain an end-to-end system such as the DataTuner, which is able to filter out less fluent/faithful generations before outputting a sentence, we require judgment more reliable than METEOR and BLEU in particular. We will investigate to what extent our RoBERTa semantic fidelity classifier can pick faithful generations from a set of 5 candidate generations $\{c_1, c_2, c_3, c_4, c_5\}$. Outputting the generation c^* that was deemed most faithful by the classifier, instead of just an unfiltered output, will hopefully increase semantic fidelity of the overall system.
We determine c^* by picking the generation associated the highest activation score for the *accurate*-class.
Alternatively, one might choose c^* by picking $c_i = c^*$ with the lowest *hallucination* activation, however this does not ensure low activation scores for other error types, encouraging unfaithful generation still.

3.5.4 Measuring classification accuracy in RoBERTa

The evaluation of the RoBERTa classifier is by far less ambiguous than the evaluation of the T5 generator. Although assessing classification quality comes down to comparing predicted labels to target labels, it is yet important to consider which quantities to consider:
To assess the overall quality of the classifier, we will of course measure *accuracy*, i.e. the percentage of labels correctly predicted. However, to better answer our research question of how to control hallucination, we will isolate the hallucination class from all other classes and compute a *binarised accuracy*:

Let

$$B : \{\text{accurate}, \text{hallucination}, \text{omission}, \text{repetition}, \text{value\_error}\} \longmapsto \{\text{hallucination}, \text{not\_hallucination}\}$$

be the label-binariser function, i.e. for any label y_0 from the domain, B is defined as

$$B(y_0) = \begin{cases} y_0 & \text{if } y_0 = \text{hallucination} \\ \text{not\_hallucination} & \text{else} \end{cases}$$

Then, for target and prediction labels $\{y_n, \hat{y}\}_n^N$, the binarised accuracy $\mathcal{B}_\backslash$ computes as

$$\mathcal{B} = \frac{\sum_{n=1}^{N} \mathbb{1}[B(y) = B(\hat{y})]}{N} \tag{6}$$

For the classification of T5-generated sentences, the evaluation of RoBERTa is less straightforward. Given that the faithfulness of data-to-text generation is best evaluated with human judgment, which is however difficult to obtain, we will only evaluate the system which shows the largest performance increase compared to pre-filtering on the automatic metrics. write summary of approach chapter

4 Experiments

In the previous Section, we have presented our approach to the data-to-text generation problem, naming some configuration details of the two system sub-components - the T5 generator and the RoBERTa classifer, how datasets were pre-processed and different consistency level versions of the datasets were instantiated. We have also presented our cross-training approach, where we included alien corpora into the training set, marking each training example with dataset-identifying prefixes. Finally, we have given some technical details on automatic metric evaluation, the questions asked during human judgement, and introduced RoBERTa as a semantic fidelity classifier for our own generations.
Having laid out the setup in close detail in the Approach Section, we will now present and put into context every experiment we performed.

4.1 Text generation

The first set of experiments, described in this Subsection, concerns the T5 generator, where the task is to generate fluent and faithful text, given a linearised version of an abstract meaning representation.
All trained T5 model instances will be tested on the test set of `webnlg-consist`. This makes WebNLG our *target* corpus; examples from any other corpus are *alien* examples.

4.1.1 Dataset configurations

Since our approach lies in harnessing the power of strong language models, by for example training them in an optimal manner, we have tested a number of dataset configurations.
In our invented names for the dataset configurations, `+` signifies joining of datasets, as described in 3.4.1. `0.5*` implies that only half of the dataset has been added. A dataset is referred to as "consistent" when, in its examples, the text is (largely) semantically faithful to the data source.

Our experiments are guided by a number of questions, which are presented below. For each question, we have listed the set of relevant dataset configurations below the question. Some dataset constructions occur in more than one set of experiments.

A. Does the addition of **alien corpora** increase performance[28] on the target dataset (`webnlg-consist`)?
(All datasets involved are consistent.)

`webnlg-consist`

`webnlg-consist + wikiinfo-consist`

`webnlg-consist + wikiinfo-consist + viggo-consist`

B. Does **training dataset consistency** matter?

- consistency among the *target dataset* (while alien datasets are held consistent)

  `webnlg`

  `webnlg-consist`

  `webnlg + wikiinfo-consist`

  `webnlg-consist + wikiinfo-consist`

- consistency among the *alien datasets* (while the target dataset is held consistent)

  `webnlg-consist + wikiinfo`

  `webnlg-consist + wikiinfo-consist`

  `webnlg-consist + wikiinfo-consist + viggo`

  `webnlg-consist + wikiinfo-consist + viggo-consist`

C. Can **alien examples compensate for** a lack of target corpus examples?

`wikiinfo-consist`

`wikiinfo-consist + 0.5*webnlg-consist`

`wikiinfo-consist + webnlg-consist`

`wikiinfo-consist + viggo-consist`

`wikiinfo-consist + viggo-consist + 0.5*webnlg-consist`

`wikiinfo-consist + viggo-consist + webnlg-consist`

[28]fluency & faithfulness, measured in various ways (automatic metrics, human judgment, RoBERTa semantic classifier)

4.1.2 Special experiments with the best-performing generator

To investigate the effects of practical researcher choices, we trained with the best-performing setup found so far, changing only one variable each time.
The best-performing model had been a `t5-base`, trained for one epoch, on the triple joint dataset `webnlg-consist + wikiinfo-consist + viggo-consist`. The different scenarios attempted were:

1. **Uniform separators:** Instead of using positional separators `<subject>`, `<predicate>` & `<object>` in our target dataset `webnlg-consist`, we employed a uniform separator `|`.
2. **No prefixes:** We left out the dataset-indicating prefixes at the beginning of each example over the entire joint dataset, leaving it to the model to work out which meaning representation is being used.
3. **3 epochs:** Instead of training for one epoch, we trained for 3.
4. **T5-large**: Instead of `t5-base`, we employed the next-larger version[29] of T5.

In this Subsection, all data-to-text generation experiments with T5 have been introduced, as well as the rationale behind them. The T5 experiments aim to show whether the addition of alien-dataset examples can improve generation quality, considering also scenarios where fewer target dataset examples are available. Furthermore, the experiments investigate whether consistency among the training data matters, for both the target dataset and the alien datasets. Some more experiments at the end seek to push best-performing model to our maximum attainable performance.

4.2 Semantic fidelity classification

The second set of experiments, described in this Subsection, concerns the RoBERTa classifier, whose task it is to classify data-text pairs according to the faithfulness of the text to the data. Given a linearised version of an abstract meaning representation and some fluent text, RoBERTa will output one out of $K \in \{2, 3, 5\}$ labels (more in 4.2.2).
We will employ two differently sized versions of RoBERTa, training on ViGGO,

[29]Configuration file with technical details: `https://huggingface.co/t5-large/blob/main/config.json`

WebNLG and a joint version thereof, using prefixes again for the joint version. A more detailed account of the different setups is given in the remainder of this Subsection.

4.2.1 Datasets

We will investigate four different dataset configurations, all based on ViGGO and WebNLG. The semantic-classification adaptations of those corpora were manually synthesised by Harkous, Groves, and Saffari, 2020 (more detailed description in 3.1.2).

Raw vs. clean ViGGO

As explained in 3.2.1, the version of ViGGO used to train the DataTuner contained repetitions of dialogue acts and slot names. Clearing those repetitions, we have two datasets, `viggo` and `viggo-clean`. Although the repetitions do not add semantic value to the data source, the effects of additional text might pose practical problems by exploiting RoBERTa's capacities unnecessarily. To find out whether the repetitions actually affect performance, we train models of the same configuration on both `viggo` and `viggo-clean`.

Cross-training with prefixes

Similar to our approach in generating with T5 (3.4.1), we will again conjoin both corpora, by mixing examples of `viggo-clean` with `webnlg`[30] to obtain `webnlg+viggo-clean`. Each example in `webnlg+viggo-clean` will again carry a prefix indicating its original corpus.

4.2.2 Exploring classification settings: Number of labels

In the discussion of approaches to semantic accuracy evaluation with neural models in 2.3.2, several multi-class classification settings were analysed, mainly in search of a sensible set of data-to-text semantic relationship classes. Remaining close to Harkous *et al.*'s work on the DataTuner, we will adopt their setup, however testing variations of it as well. Below, three possible classification problem setups are listed:

[30]Here `webnlg` refers to the hand-labelled dataset of data-text pairs, not to be confused with T5's inconsistent generation dataset.

- **5 labels:** accurate, hallucination, repetition, omission, value error (DataTuner)
- **3 labels:** accurate, hallucination, other error
- **2 labels:** other, hallucination

Our main interest lies in the *accurate* class and the *hallucination* class. The DataTuner setting has 5 labels, considering 3 other classes (*repetition*, *omission*, *value error*). Given that the interest in the 3 other classes is low, maintaining 5 labels might perhaps result in an unnecessarily high-dimensional decision boundary. A simpler setting would file all extraneous error types with one class (*other_error*, see "3 labels"). This reductionist approach can be taken to the extreme, filtering *hallucination* against everything else, leaving us with only 2 classes.
We will test each of the settings, to identify the most suitable setting for our research goal of reducing hallucination.
Having laid out all T5 generator and all RoBERTa semantic fidelity classifier experiments, in the next Subsection, we can finally turn to the experiments with combining the two.

4.3 Combined system

The goal of instantiating both a T5 data-to-text generator and a RoBERTa semantic fidelity classifier was to eventually assess the T5's generation faithfulness with the RoBERTa classifier.
In the combined system, T5 generates 5 candidate outputs, which the classifier then labels according to their semantic fidelity to the common data source.
To obtain the output candidates, we train 5 equivalent instances of a T5 generator in the best-performing setup. Generating candidates from the same model often involved a lack of variety, so that training 5 different instances seems a valid trade-off. Now, even in simple data sources, we have considerable variation among the candidates (see Listing 8 for an example).

4.3.1 Picking a generation output from candidate generations

Once we have 5 candidate generations $\{c_1, c_2, c_3, c_4, c_5\}$ in place, we need to label them to obtain c^*, the most faithful one.
The classifier will be an instance of RoBERTa with the **largest binarised accuracy** in previous experiments.

Listing 8: **Linguistic variance among candidate generations.** Despite only one WebNLG triplet being present in the data source, among the 5 generations we find 3 different ways of expressing the content. All generations are fluent and faithful to the data source.

```

"data": "<subject> Farrar & Straus <predicate> parent company <
    object> Macmillan Publishers",

"gen": [
"Farrar & Straus is a parent company of MacMillan Publishers.",
"Farrar & Straus are the parent company of Macmillan Publishers.",
"Macmillan Publishers is the parent company of Farrar & Straus.",
"Macmillan Publishers is the parent company of Farrar & Straus.",
"Farrar & Straus is owned by Macmillan Publishers."
],

"ref": "Macmillan Publishers is the parent company of Farrar,
    Straus and Giroux."
```

We select the generation to output by filtering out the candidate with highest activation value for whichever class contains *accurate* examples.

Notice that since we are classifying real-life examples (our generations), no target labels are available. Hence, there is no way of evaluating the quality of the classifier, apart from inspection. We will not perform a human evaluation on the classification accuracy, since the classifier's most relevant performance metric is whether its filtering has boosted faithfulness among the final system's generations.
Since we *will* evaluate the final system's output, i.e. the filtered generations, we will obtain a quantity reflecting on the classifier's performance.

In this Section, we have described the various experiments we will perform, where we discussed data-to-text generation experiments first, and then semantic fidelity classification experiments, to finally combine our knowledge gained about the best-working techniques into a combined generator-classifier system. For data-to-text generation, we are attempting to find which factors among the dataset configuration are most beneficial whilst cheap to implement, such as the addition of alien datasets, faithfulness (in target/alien corpus), and cleaning/linearisation methods. We additionally investigate running more epochs or using a larger T5.
The semantic fidelity classification experiments target similar conditions, again

comparing different dataset configurations in terms of cleaning/linearisation (`viggo` vs. `viggo-clean`) and dataset joining (`webnlg+viggo-clean`).
Having analysed the two sub-compontents of our combined system thoroughly, our final experiment will investigate the effect of filtering the candidate text, aiming to produce the most faithful set of output generations.

In the next Section, the experimental results will be revealed and discussed, upon which our system and its sub-components will be critically assessed in the current research context.

5 Results & Evaluation

In the previous Section, we have described various experiments, concerning the T5 generator, the RoBERTa classifier, and the combined system emerging from those two components. For both components, we experimented with different dataset configurations, exploring the effect of dataset cleaning and semantic de-noising, as well as joining datasets. Other notable aspects to analyse were the choice of the number of classification labels, and cleaning and linearisation techniques.
This Section will now present the results to all experiments with the RoBERTa classifier and the T5 generator, as well as the combined system.

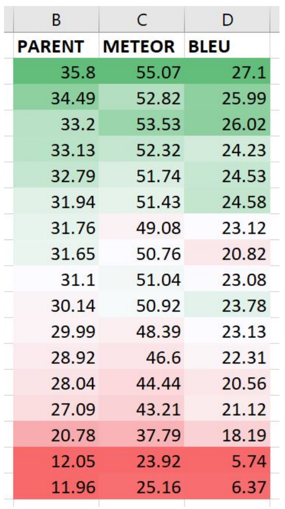
5.1 Data-to-text generation results

As previously explained, we deem the PARENT score the most accurate and appropriate metric for data-to-text generation among BLEU, METEOR and PARENT. However, in our experiments, we found the scores to be highly correlated, giving very similar judgements about the models (see Figure 4). While BLEU and METEOR will be kept for reference, their values do not make a fundamentally different statement from PARENT.

5.1.1 Dataset expansion with alien examples

In the first series of experiments, we added examples from other corpora (ViGGO, WikiInfo2Text) to our WebNLG dataset, pre-pending each example with a dataset-identifying prefix and then randomising the order of the examples.
The observed effects of expanding the training set in this manner are described below, answering two of our questions.

Can alien datasets increase performance on the target dataset?
To see whether the addition of alien datasets can increase generation performance, we first added `wikiinfo-consist` to target dataset `webnlg-consist`, and in the next step also added `viggo-consist`.
We compared the resulting performances to a system trained on just `webnlg-consist`.
Detailed results on all three automatic metrics can be found in Table 3.
Figure 5 presents the PARENT scores of the three generators. Whereas the generator trained on just `webnlg-consist` scores 31.10, training on a mixed corpus increases PARENT to 31.94 by adding one other corpus, and to 34.49 by adding



B	C	D
PARENT	METEOR	BLEU
35.8	55.07	27.1
34.49	52.82	25.99
33.2	53.53	26.02
33.13	52.32	24.23
32.79	51.74	24.53
31.94	51.43	24.58
31.76	49.08	23.12
31.65	50.76	20.82
31.1	51.04	23.08
30.14	50.92	23.78
29.99	48.39	23.13
28.92	46.6	22.31
28.04	44.44	20.56
27.09	43.21	21.12
20.78	37.79	18.19
12.05	23.92	5.74
11.96	25.16	6.37

Figure 4: The **strong correlation between all three automatic metrics** can be observed by applying the Excel colour scale tool to each column separately.
The small colour variations in each row reveal that essentially each metric would rank the models similarly, not fundamentally disagreeing on any of the models' performance.

two corpora. The gain from adding a second alien dataset (2.55) is considerably larger than the gain from adding the first alien dataset (0.84), suggesting that one benefit of mixed-corpus training lies in learning to generalise.
One theory as to why different datasets can contribute to the performance on the target dataset is that T5 learns to generalise w.r.t. meaning extraction from the linearisation, adapting to different syntaxes swiftly. Having overcome the syntactic hurdle, adding more datasets results in having more examples of fluent text, which of course is conducive to learning the task.

Mixing datasets can thus harness more of the language model's power to generate from a particular corpus.

Alien examples and target-data scarcity

Having observed how adding alien datasets can boost generation quality on the target dataset, we now investigate whether the mixed-corpus training technique can compensate for a lack of target training examples. To that end, we tested tested mixed-corpus training in three different scenarios:

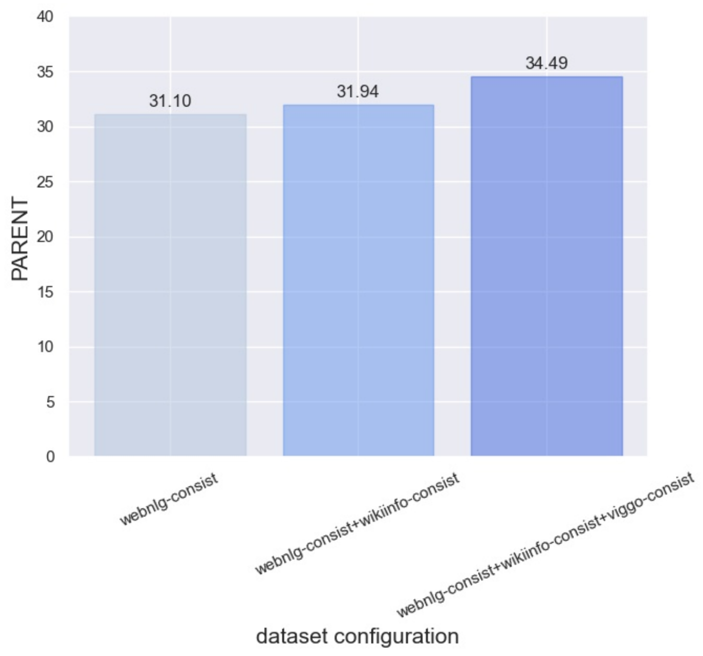


Figure 5: **Adding alien examples to target dataset `webnlg-consist`.** With each new corpus added, the PARENT score increases, when testing on `webnlg-consist`. A more significant increase is observed when adding the *second* alien dataset.

1. *all* of `webnlg-consist` is present
2. *half* of `webnlg-consist` is present
3. *none* of `webnlg-consist` is present

The three different scenarios were tested under the addition of both *one* alien dataset and *two* alien datasets.

dataset configuration	PARENT	METEOR	BLEU
`webnlg-consist`	31.1	51.04	23.08
`webnlg-consist` + `wikiinfo-consist`	31.94	51.43	24.58
`webnlg-consist` + `wikiinfo-consist` + `viggo-consist`	34.49	52.82	25.99

Table 3: Automatic metric scores on text generated from data, when we **add alien examples to target dataset** `webnlg-consist`.

dataset configuration	PARENT	METEOR	BLEU
`wikiinfo-consist`	11.96	25.16	6.37
`wikiinfo-consist + 0.5*webnlg-consist`	28.04	44.44	20.56
`wikiinfo-consist + webnlg-consist`	31.94	51.43	24.58
`wikiinfo-consist + viggo-consist`	12.05	23.92	5.74
`wikiinfo-consist + viggo-consist + 0.5*webnlg-consist`	29.99	48.39	23.13
`wikiinfo-consist + viggo-consist + webnlg-consist`	34.49	52.82	25.99

Table 4: Automatic metric scores on text generated from data, in the attempt to **compansate for target data scarcity** with alien examples.

Figure 6 shows that in target data scarcity situations, the generator benefits from having more alien data. However, with many alien datasets, the maximum attainable score is naturally higher (see the previous experiment's Figure 5), so that compared to the highest attainable score, `wikiinfo-consist+viggo-consist+0.5*webnlg-consist` model loses a higher percentage of PARENT than `wikiinfo-consist+0.5*webnlg-consist`. Comparing `wikiinfo-consist+viggo-consist+0.5*webnlg-consist` to training on just our entire target dataset `webnlg-consist` (Figure 5), we obtain respective PARENT scores 30.14 and 31.10, so that the absence of half of WebNLG cost us 0.96. It is therefore possible that, with statistical variance among results, the addition of alien datasets can compensate for small target datasets. With larger alien datasets however, full compensation seems within reach.
Alien dataset size effects have not been investigated here, providing an opportunity for further research. We have however shown that it is fundamentally possible to compensate for a lack of target dataset examples.
Find the complete set of scores in Table 4.

5.1.2 Influence of dataset consistency on generation fidelity

5.1.1 has presented results on dataset joining from a quanititative perspective, i.e. how many datasets were joined. We will now shift the focus to dataset quality, examining the effects of *training set consistency*, among both the target dataset and any alien datasets.

Consistency among the target dataset

For both WebNLG alone and the combination of WebNLG and `wikiinfo-consist`, Figure 7 shows significant performance loss in case target corpus WebNLG was

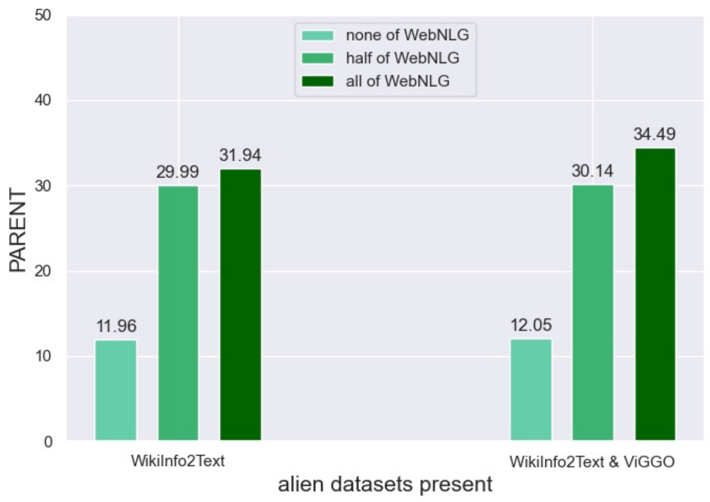


Figure 6: **Using alien examples to compensate for a lack of target examples.**
For either one or two alien datasets (left & right), this plot shows PARENT when WebNLG is either not present, half-present or fully present.
Even when not training on any `webnlg-consist` examples, PARENT is non-zero for both one alien dataset and two alien datasets (see lightest series).
With just half of `webnlg-consist` available, the performance is almost fully exploited in the one-alien-dataset setting (29.99 vs. 31.94). While in the two-alien-dataset setting, the performance gap between half and full `webnlg-consist` is more considerable, the system yet benefits from having an extra alien dataset $(30.14 - 29.99 > 0)$.

dataset configuration	no. of train. ex.	PARENT	METEOR	BLEU
`webnlg`	100’000	20.78	37.79	18.19
`webnlg-consist`	20’000	31.1	51.04	23.08
`webnlg + wikiinfo-consist`	149’917	27.09	43.21	21.12
`webnlg-consist + wikiinfo-consist`	69’917	31.94	51.43	24.58

Table 5: **Varying consistency among the target dataset,** in absence and presence of a consistent alien dataset.
Regardless of alien data being present, the performance on the consistent dataset is substantially larger, despite only a fraction of the number of examples being available.

noisy. (Alien corpus WikiInfo2Text was held consistent.)
We lose ca. 10 points on the PARENT scale training on `webnlg` instead of `webnlg-consist`, and ca. 5 points training on `webnlg+wikiinfo-consist` instead of `webnlg-consist+wikiinfo-consist`.
As `webnlg` contains 100′000 examples, whereas `webnlg-consist` contains only 20′000, the significantly higher performance on configurations involving `webnlg-consist` is even more notable. Consistency weighing therefore seems to weigh much more heavily than training set size.
Given the factor of 5 between the training set sizes, our findings also suggest that training data consistency is an indispensable factor, which cannot be compensated for by dataset size, if absent.

Find the complete set of scores and training set sizes in Table 5.

Consistency among the alien datasets

Beyond the consistency among the *target dataset*, we have also investigated the effect of consistency among alien datasets.
We investigate adding either *one* or *two* inconsistent alien datasets to `webnlg-consist`. Figure 8 reveals no significant effect of alien dataset consistency on performance on the target set. This invariance of performance under e.g. noisy alien datasets opens up an opportunity: A consistent-target-noisy-alien dataset allows for benefits from joint-dataset synergy, while requiring little care for the alien datasets.

A possible explanation as to why the performance on WebNLG remains unchanged under different alien dataset consistency levels, is that T5 learns to

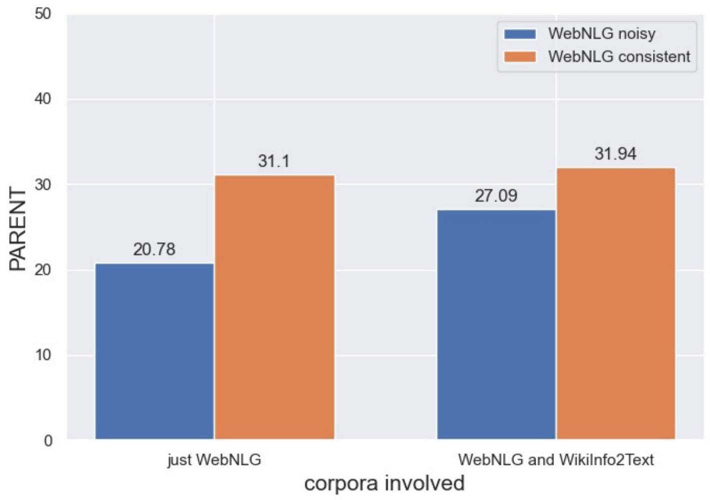


Figure 7: **The effect of target dataset consistency on PARENT scores.** *Left:* When training on just WebNLG, having consistency among the dataset boosts PARENT by more than 10. *Right:* When another dataset (`wikiinfo-consist`) is present, using the consistent version of WebNLG still increases PARENT by 4.85.

treat each present dataset as a separate task, developing an individual technique for generating from each dataset. The addition of dataset-indicating prefixes helps this separation, as suggested by the successful use of task prompts such as `translate-en-to-fr` or `summarize` (P. Liu et al., 2021), which move well-trained modern-day language models into a completely different generation mode.

For sections 5.1.1 and 5.1.2, T5 has been trained on various dataset configurations, to investigate how performance can be increased in terms of joining a dataset with alien datasets, and in terms of using training data as consistent as possible. Since various settings beyond the dataset configuration remained unchanged throughout the experiments, we will in 5.1.3 present some brief experiments which aim to cover likely performance boosters.

Find all results in Table 6.

5.1.3 Testing the best-performing model under different conditions

From the above experiments, the `t5-base`, trained on one epoch, performed best when trained the largest possible joint dataset, fully consistent, i.e. `webnlg-consist+wikiinfo-consist+viggo`

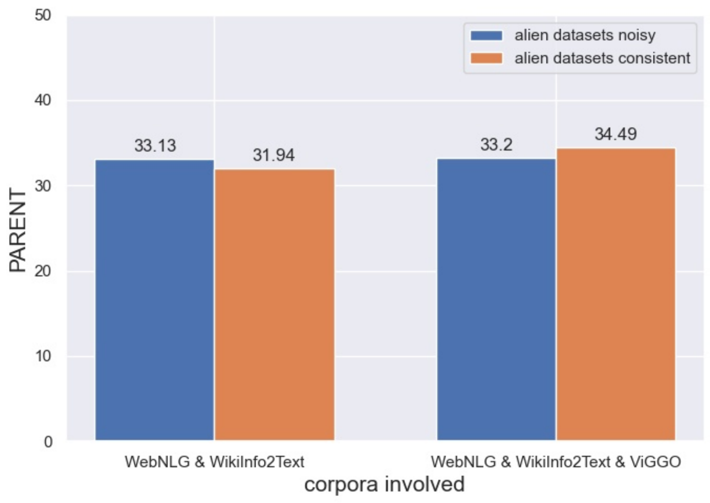


Figure 8: **Alien dataset consistency has little effect on PARENT scores.** *Adding one (left) vs. two (right) inconsistent datasets to consistent WebNLG.* The addition of inconsistent alien datasets (*orange compared to blue*) does not have a strong effect on performance on the target dataset. Given a decrease in one scenario, and a decrease in the other, the variations likely fall under the statistical variance of model performance.

dataset configuration	PARENT	METEOR	BLEU
`webnlg-consist` + `wikiinfo`	33.13	52.32	24.23
`webnlg-consist` + `wikiinfo-consist`	31.94	51.43	24.58
`webnlg-consist` + `wikiinfo` + `viggo`	33.2	53.53	26.02
`webnlg-consist` + `wikiinfo-consist` + `viggo-consist`	34.49	52.82	27.1

Table 6: **Varying alien consistency** has little effect on generation performance. No major score differences are observable.

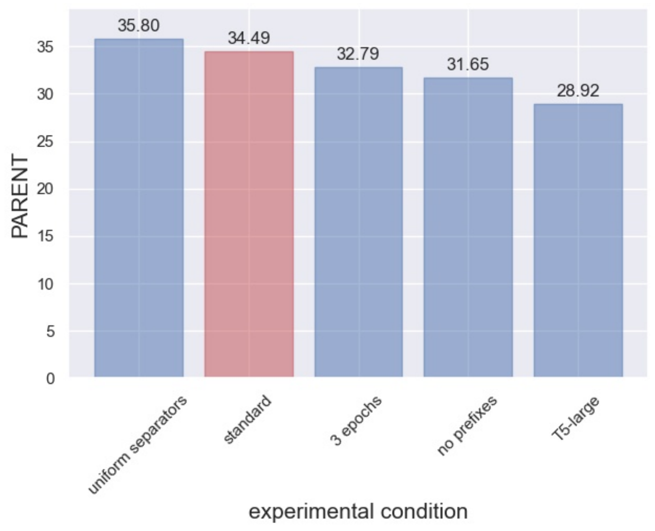


Figure 9: **Investigating the model under changed conditions.**
Each bar corresponds to one condition being changed (except for the red bar). Only by using uniform separators when linearising graph-based dataset WebNLG, we obtain a performance increase. Training for 3 epochs or swapping T5-large for T5-base does not help. The removal of datast-indicating prefixes unsurprisingly results in a decrease as well.

(covered in 5.1.1). Figure 9 presents the PARENT score for this model configuration, with one condition altered each time.
All metric scores are available in Table 7.

Uniform WebNLG separator tokens

Harkous *et.* al have linearised the original WebNLG data using semantics-based separator tokens `<subject>`, `<predicate>` & `<object>`.
Replacing the semantics-based separators `<subject>`, `<predicate>` & `<object>` by a uniform one (`|`) has improved the PARENT score by 1.31, from 34.49 to 35.8. This performance increase suggests that the information extraction process does not gain from the semantic structuring provided by the positional separators. T5 might be able to keep track of semantic roles itself. In fact, losing performance, T5 seems to be distracted by the changes in separators, perceiving them as noise.

Training for 3 epochs

experiment condition	PARENT	METEOR	BLEU
uniform separators	35.8	55.07	27.1
standard	34.49	52.82	25.99
3 epochs	32.79	51.74	24.53
no prefixes	31.65	50.76	20.82
`t5-large`	28.92	46.6	22.31

Table 7: **Selected experiments on the best-performing model** (`t5-base`, one epoch, `webnlg-consist+wikiinfo-consist+viggo-consist`).
Except for the use of simple separators, none of the changed conditions bring a positive change.

Training on 3 instead of one epoch decreased performance slightly (from 34.49 to 32.79), implying that one epoch is sufficient for `t5-base` to yield its full potential.

Leaving out the dataset-indicating prefix

The aim of this experiment was not to boost performance, but to investigate how strongly the dataset tagging of training examples contributes to performance. And indeed, PARENT drops from 34.49 to 31.65 without prefixes, emphasising the helping function of prefixes. Given that task-specific prefixes work successfully with modern-day language models, the addition of dataset-specific prefixes might help the model to treat the datasets as different tasks, reducing the capacity spent on identifying the linearisation syntax type.

Using T5-large

Similarly to the number of epochs being increased, `t5-base` seems to suffice for the task, so that `t5-large` is not required. Performance drops significantly with `t5-large` (from 34.49 to 28.92), which has many potential explanations; the most probable one being overfitting.

5.1.4 Human faithfulness & fluency evaluation of the best-performing model

We have obtained 114 human judgements on fluency and faithfulness of reference and generation in the best-performing model (`t5-large`, one epoch, `webnlg-consist+wikiinfo-consist+viggo-consist`). Although evidently the model was tested on the consistent version of WebNLG, only 50.0% of the reference sentences were fluent and faithful to the data source at the same time.

While references were generally fluent, they displayed minor semantic inconsistencies, such as forgetting to mention several letters:

- **data:** `<subject> Aaron Hunt <predicate> club <object> SV Werder Bremen ; <subject> Aaron Hunt <predicate> club <object> VfL Wolfsburg`
- **gen:** `Aaro plays for Vfl Wolfsburg and SV Werder Bremen.`

In order to have a meaningful evaluation, even for small inconsistencies, the sentence had to be marked as unfaithful.
74.56% of the generations were fluent, where the main culprit was the repetition of substrings, such as:

```
A.F.C. A.F.C. A.F.C. Blackpool's ground is The Mechanics.
```

We only evaluated faithfulness on fluent generations, since the meaning might not be clear otherwise. Out of all fluent references, 50.59% were also faithful, where minor inconsistencies weighed most heavily. When the WebNLG example had more triplets (e.g. 5 to 7), the the generator would frequently forget to addseveral facts. Though even for lower numbers of triplets, omissions accounted for the vast majority of unfaithful generations. In many cases, very subtle semantic mistakes occurred, which we marked as errors. However, they did not even belong into any of the semantic fidelity classes identified in 4.2.2. One example of subtle mistakes is not picking up on the past tense implied by "work*ed*":

- **data:** `<subject> Abdulsalami Abubakar <predicate> office (worked at, worked as) <object> Chief of the Defence Staff (Nigeria) ; ...`
- **gen:** `Abdulsalami Abubakar [...] is the Chief of the Defence Staff of Nigeria.`

The semantic error was hallucination in 9.52% of the unfaithful examples, amounting to 4 hallucinations, which is only 3.51% of the entire dataset. Upon close inspection, none of the instances classed as hallucinations were typical examples - no entirely new piece of information was made up, except for one case with a long data source, the generator added several instances of

```
The author of the book is Sumac.
```

at the end of its generation. While `Sumac` had occurred in the data, a book was never mentioned.

In summary, the human reviews have revealed that the best-performing model is indeed not prone to hallucination, however frequently omits facts instead, especially for long input data. We have also observed that the references of `webnlg-consist` were of lower consistency than anticipated, which makes it harder for generators to learn and score well on automatic metrics. In fact, in 15.79% of the examples, the reference was labelled as "bad" (not fluent or not unfaithful), whereas the generation was labelled as fluent and faithful. The model output being superior to the reference in such a number of cases reveals that the model has acquired some skill underlying fluent & faithful data-to-text generation, instead of just copying what has been seen during training.

5.2 Semantic fidelity classification results

This Subsection presents the results on the RoBERTa semantic fidelity classifier, which will serve as 1-out-of-5 candidates selector in the combined generator-classifier system.
To increase RoBERTa's potential to classify data-to-text semantic fidelity, we have tested different dataset configurations, varying in levels of syntactic & semantic cleaning, as well as the number of corpora joined. Furthermore, settings with $K = \{2, 3, 5\}$ classes were contrasted, as well as `base` and `large` version of RoBERTa.

In search of methods to control hallucination, we report *binarised* accuracy (Eqn. (6)). For all classification experiments, odels were tested on the test set given for the training corpus.

5.2.1 Classification accuracy by dataset configuration

In order to later employ our classifier for picking the best candidate generation, the classifier must be trained on the generator's target dataset, WebNLG.
We have tested training on just WebNLG vs. on a joint datset (adding ViGGO). Results are visualised in Figure 5.2.1.
Having evaluated `viggo-clean` against `viggo`, we conjoined WebNLG with `viggo-clean`, which generally achieved marginally higher binary accuracy than raw `viggo`.

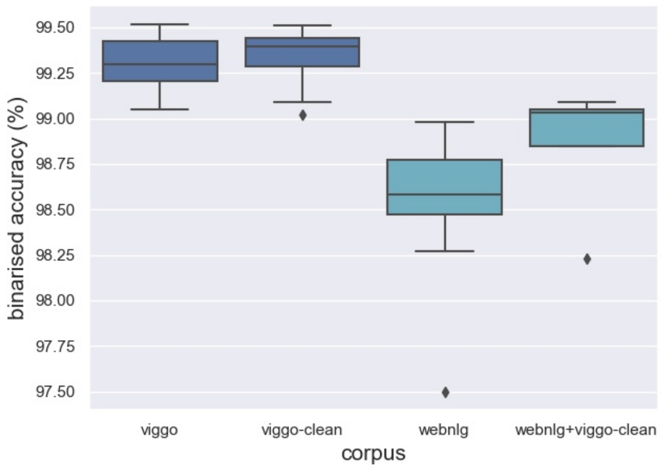


Figure 10: **Binarised classification accuracy: A comparison different dataset configurations.**

The data were collected during training various models in preparation of this thesis. In total, just under 48 (= 4 datasets × 2 no. of epochs × 3 no. labels × 2 model sizes) data points are available (excluding a few non-working `roberta-large`s - see Section 3.4.2).

Dark blue: ViGGO, before vs. after cleaning repetitive dialogue acts & slot names. We do in fact obtain a higher classification accuracy when cleaning repetitions of dialogue acts / slot names in `viggo-clean`.

Light blue: `webnlg`, without vs. with `viggo-clean`. Conjoining `webnlg` with `viggo-clean` helps.

Notably, the percentage of correct classifications for all models on average lies around 99%, on a dataset-specific held-out test set.

Since `webnlg+viggo-clean` achieves on average 0.5% more than `webnlg` alone, we observe the same synergetic effects of joint training as in the generator experiments. The gain might seem negligible, however considering that the majority of results lies above 98.5%, a jump to 90.0% is worth noting.
For the system combination, we will thus train on `webnlg+viggo-clean`.

5.2.2 Optimal classification setting: Number of classes

We have tested $K = \{2, 3, 5\}$ numbers of semantic fidelity clases, where *hallucination* always constituted a class (details in 4.2.2).

As Figure 11 illustrates, when considering *accuracy*, we are led to believe that $K = 2$ (low) is the most suitable setting. Higher K naturally lower *accuracy*, as guessing more frequently leads to picking the wrong class. However, with high K we also count errors that are irrelevant to controlling hallucination (e.g. for $K = 5$: *omission* vs. *value error*). When we however isolate the relevant class *hallucination* through *binary accuracy*, we observe the highest performance with the most refined class setting ($K = 5$).
Thus, with $K = 5$, we identify hallucination the most accurately.

To see why a seemingly more complicated setup (e.g. $K = 5$) results in best hallucination detection, consider the following data source and hypothetical generations:

data source: `<subject> Mary Stuart <predicate> crowned_year <object> 1542`

generation (value error): `Mary Stuart was crowned in 1555.`

generation (repetition): `Mary Stuart was crowned in crowned in 1542.`

Whereas value error detection requires comparing generation and datasource, repetition requires comparisons among the generation. Therefore, the detection of *value error* vs. *repetition* are two fundamentally different tasks. The lower

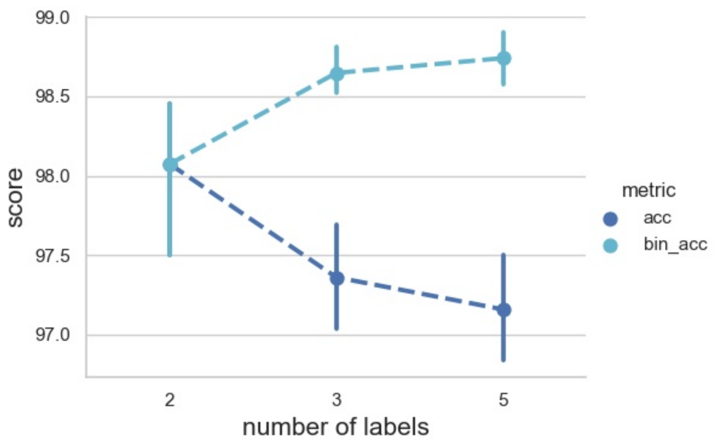


Figure 11: **Evolution of classification accuracy vs. binarised accuracy** as the number of classes increases.
Evaluated on the WebNLG dataset, but a similar trend is visible for any dataset.
As the number of labels increases, accuracy decreases, since guessing results more likely in the wrong prediction.
Binarised accuracy however increases with more refined labels, so that for the individual class of hallucination, we actually detect with higher accuracy.

binary accuracy for $K = 2$, where accurate and faulty generations are mixed, and $K = 3$, where all non-hallucination errors are mixed, is thus less of a surprise.
With $K = 5$ being most suitable for hallucination-detection, and $K = 3$ being almost as good, the classifier seems to prefer homogeneity among the classes, which is not given for $K = 2$. $K = 3$ allows for some homogeneity, not mixing up accurate and faulty examples.

Given our findings, we will utilise 5 semantic error classification labels in our combined system.

5.2.3 How the RoBERTa classifier thinks

To understand more about the semantic fidelity classifier's 'reasoning', we have counted the number of confusions between the $K = 5$ classes (see Figure 12).
We observe that classes containing shorter texts (*accurate*, *omission*, *value error*) are frequently confused. Furthermore, an elevated level of confusion is present among the elongated classes *hallucination* & *repetition*. Instead of com-

paring the semantic content of data and text, RoBERTa seems to derive labels from macro-features in those cases. However, the vast majority (97.56%) of examples are still classified correctly. Upon inspection, false labels occur only in highly ambiguous cases, such as:

- **data:**
  `<subject> AIDAstella <predicate> christening date <object> 2013-03-16`
- **text:** `The AIDAstella was completed on March 11th 2013.`
- **target label:** *hallucination*
- **predicted label:** *accurate*

Annotators have probably labelled this as *hallucination* as christening is different from *completion*, however those two events are very much related to the same concept. Notice that being as strict, *value error* would also have made sense.
The 2.44% of false classifications might overlap with the ambiguous examples of the dataset.

The results obtained so far in this Section, on data-to-text generation (5.1) and semantic fidelity classfication (5.2), have helped us identify a set of well-working whilst practicable design choices for our generator-classifier combined system, which we will turn to in 5.3.

Scores on all configurations relevant to the combined system are listed in Table 8.

5.3 Combined system results

Drawing together the expertise gained in the previous two subsections, the design choices for the combined system's components are summarised below:

1. **T5-generator**:
   - `t5-base`
   - 1 epoch
   - semantics-based separators (for comparability with Harkous, Groves, and Saffari, 2020)

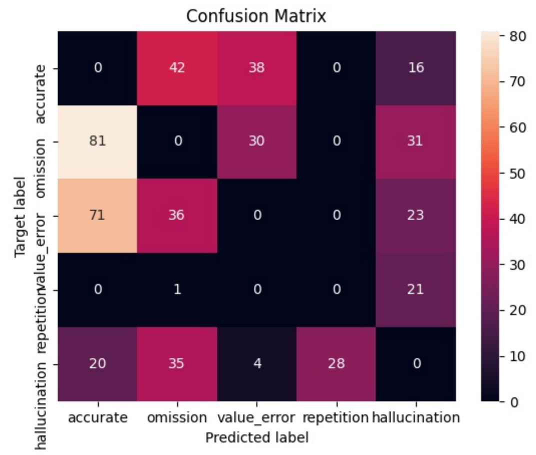


Figure 12: The **confusion matrix** of the best-performing semantic fidelity classifier.
Diagonal values have been set to 0 *since they do not constitute confusions.*
We observe frequent confusions between *value error*, *omission* & *accurate*. *Repetition* and *hallucination* are confused with some frequency as well.
Among those two groups of common confusion, a length pattern can be spotted (resp. *short* vs. *long*).

number of epochs	number of labels	accuracy	bin. accuracy
1	2	**98.23**	98.23
1	3	97.02	98.85
1	5	97.29	99.03
3	2	-*	-*
3	3	97.86	99.05
3	5	97.56	**99.09**

Table 8: **Classification performance of `roberta-base` on best dataset, `webnlg+viggo-clean`.**
*: Training failed due to the same pathology as in `roberta-large`.
Bold: Highest score - configuration will be used in combined system.
For more labels, 3 epochs are beneficial, since the classification problem is more complex.

- trained on `webnlg-consist+wikiinfo-consist+viggo-consist`, with dataset-indicating prefixes

2. **RoBERTa semantic fidelity classifier**

   - `roberta-base` (due to training complications with `roberta-large`)
   - 3 epochs
   - 5-label classification problem (*accurate*, *hallucination*, *omission*, *repetition*, *value error*)
   - trained on `webnlg+viggo-clean`

5.3.1 Automatic metric scores before semantic filtering

Before employing RoBERTa to filter out one out of five candidate generations, we must check for sufficient diversity among the candidates, to ensure that the classifier has a range of choices. With insufficiently diverse candidates, the filtering through the classifier can barely change the system output.
Sufficient diversity among the candidate generations has been confirmed by inspection (Listing 8), as well as considerable variance in automatic metric scores (see Figure 13).

5.3.2 Combined system results: Automatic metric scores after semantic filtering

In Figure 14, we report pre-filtering metric scores (candidate average), in comparison to post-filtering metric scores, i.e. the combined system's output. Every metric score is increased notably. All score changes are contrasted in Table 9.
With BLEU having the largest relative increase (9.34%), the selection process of a generation seems to have looked for n-gram overlap. This hypothesis is further supported by METEOR undergoing a 4.72% increase, about half the change of BLEU. METEOR pays attention to both n-grams and general semantic overlap.
PARENT, which computes an n-gram F1-score w.r.t. to the data source,

metric	pre-filter score	post-filter score	absolute increase	relative increase (%)
PARENT	33.16	36.06	2.9	8.75
METEOR	53.81	56.35	2.54	4.72
BLEU	25.59	27.98	2.39	9.34

Table 9: **Automatic metric scores** in the T5-generator (average of all candidate generators) vs. after filtering with a RoBERTa semantic fidelity classifier.

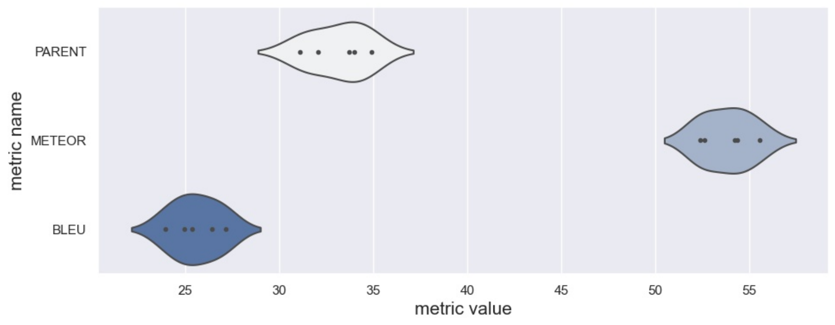


Figure 13: **Automatic metric scores of the** 5 **candidate generators.**
The plot shows 5 *datapoints for each automatic metric, corresponding to the* 5 *candidate generator instances.*
The training procedure for each candidate generator was identical.
PARENT varies between 31.11 and 34.9.
METEOR varies between 52.38 and 55.54.
BLEU varies between 23.98 and 27.19.
The automatic metric scores vary enough to assume diversity among the candidate generations.

increases almost as strongly as BLEU, by 8.75%. This large increase confirms that the RoBERTa classifier increases the generations' faithfulness to the data source.

The maximum boost on PARENT is 2.9.

5.4 Evaluation summary & findings

In this Section, we have presented our results to experiments concerning data-to-text generation, as well as semantic fidelity classification.
For both sets of experiments, we have added alien corpora to target corpus WebNLG, forcing the underlying language model to learn generalising over input structures. Given that the joining was highly beneficial for both tasks,

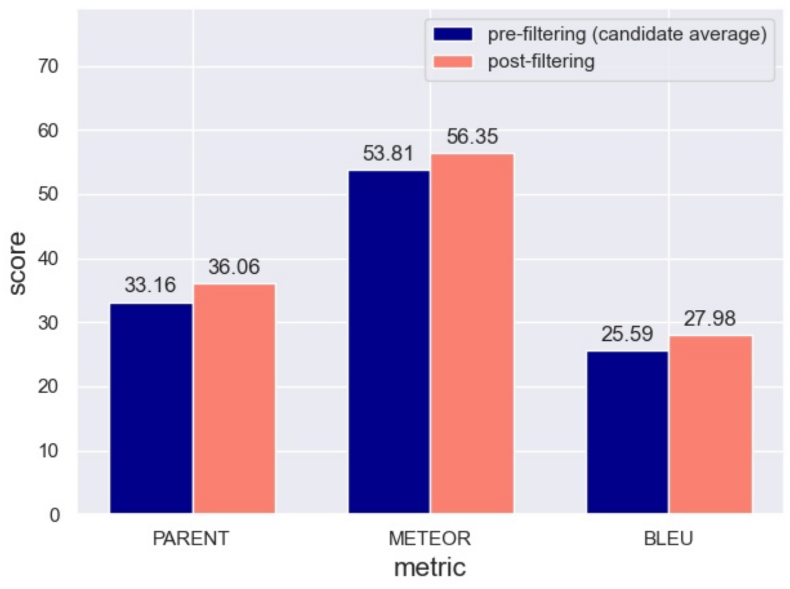


Figure 14: **Pre-filtering vs. post-filtering performance**: The effect of filtering candidates according to their semantic fidelity before outputting.
The pre-filtering automatic metric scores are averages of the 5 *candidate generators.*
The semantic fidelity filtering increases each automatic metric score notably.
PARENT is increased from 33.16 to 36.06, by 2.9 points, corresponding to a plus of 8.75%.
METEOR increases from 53.81 to 56.35, by 4.72%.
BLEU undergoes the largest relative change, increasing from 25.59 to 27.98, by 9.34%.

language models seem to be highly flexible in adapting to structural differences between corpora. From the positive effects of dataset-indicating prefixes measured in T5, we have inferred that in general addition of such prefixes allows language models to perceive each corpus as a separate task, which ultimately enables parallel learning from several corpora.
For data-to-text generation, we have also found that training set consistency matters only among the target dataset. This allows for adding alien datasets without performing extensive semantic cleaning on them in advance.
In fact, when the training set was consistent, considerably higher PARENT scores were achieved, with just a fraction ($> 20\%$) of the training data. While this finding highlights a further dial in increasing performance, obtaining high-fidelity datasets for data-to-text generation is often not possible without some cleaning effort on the researcher's side (Parikh et al., 2020).
Surprisingly, using uniform data separator tokens `|` in WebNLG was empirically beneficial to the performance, questioning the gain from semantics-based tokens.
We have also established that the base versions of T5 and RoBERTa are sufficient for our problem, training them on a small number of epochs (resp. 1 and 3).
Despite the seemingly complicated setup of having 5 semantic fidelity classes when our main interest lies in *hallucination* and *accurate*, having one class for each possible case facilitated drawing a meaningful decision boundary, and led to more faithful detection of *hallucination.*

Finally, the human reviews of the pre-filtering outputs of T5 have revealed a much stronger prevalence of omissions than hallucinations. We thus have little hallucination left to control with RoBERTa. Due to the overwhelming number of omissions observed, in contrast to the tiny amount of hallucination, it is possible that a scale of loquacity exists for language models, on which our generator is rather low.

Whereas T5-generator's quality was thoroughly assessed with three automatic metrics and human reviewing, potential weaknesses of the classifier within the combined system have not directly been investigated. Evidently, we have evaluated the classifier on an artificial test set in 5.2, where the accuracy was mostly $> 98\%$. However, the only available token of classification accuracy on *the generated set* is the metric score change induced by filtering with the RoBERTa classifer. Since the increase percentages are quite considerable, also in comparison to Harkous, Groves, and Saffari, 2020, we will for now assume a satisfying level of classification accuracy on the set of generated sentences. Future work might hand-label a subset of the generations, to obtain an estimate of classification accuracy.

In their evaluation of the DataTuner on WebNLG, Harkous *et al.* obtain a pre-filtering METEOR score of 41.9 on WebNLG, whereas we achieve 52.82 with training a `t5-base` on `webnlg-consist` + `wikiinfo-consist` + `viggo-consist` (5.1.1). With the simplified separators, we even obtain 55.07. Harkous *et al.* can increase their METEOR score to 42.4 (+1.19%) through filtering, compared to our 56.35 (+4.72%). However, given that we trained our generator on a consistent version of WebNLG, it might be easier to score higher than on raw WebNLG[31], used by Harkous *et al.*. Ultimately, a direct comparison to Harkous *et al.*'s METEOR score is therefore less informative, but substantial deficits compared to the DataTuner seem unlikely.

[31]When the reference sentences are noisy, and METEOR measures the overlap between generations and (noisy) reference, which is naturally lower.

6 Conclusion & Future directions

In this thesis, we have chosen a three-stage approach to increasing semantic faithfulness in data-to-text generation. We first ensure consistency among the data, and conjoin distinct datasets for mixed-corpus training. Second, we generate 5 candidate texts with a T5-base, and lastly classify the semantic faithfulness of each candidate to the data source with a RoBERTa-base. The candidate with the highest activation for class *accurate* is selected as system output.
This approach was first explored by Harkous, Groves, and Saffari, 2020, in the DataTuner.

The pre-existing hallucination pathology in conditional language generation is in the case of data-to-text generation additionally fuelled by noisy training sets - the of overlap between data and text among corpora has called for serious measures of faithfulness control in generators. Existing remedies, such as including a copy-gate (W. Chen et al., 2020) or adding a Hidden Semi-Markov Model to learn latent templates (Wiseman, Shieber, and Rush, 2018) tend to tackle unfaithfulness *during* the generation process, making architectural additions to their generators.
Contrarily, we interfere with neither the architecture of our core generator, nor the classifier, in fact using the base configuration for both (`t5-base`, `roberta-base`). Yet we were able to achieve considerable performance increases compared to as-is use by presenting our training data in a strategic manner:

First, we performed additional cleaning to the DataTuner's dataset. Then, for the WikiInfo2Text dataset, we applied a semantic de-noising procedure, which ruled out slot-value pairs if their slot name was deemed irrelevant in advance, or if a Latent Semantic Indexing model could not find semantic overlap between the reference and the value. Examples with a high text-to-datapoints ratio were also excluded. In absence of semantic inconsistency, models trained on a small number of faithful examples outperformed models trained on much larger, but inconsistent, datasets. Given that in most of previous research, generators are trained on data-to-text corpora without much semantic cleaning, dataset consistency seems an underexplored dial in generating faithfully.
Furthermore, considerable performance gains were obtained, for both generation and classification, by merging the target dataset (WebNLG) with other datasets of the same task, pre-pending dataset-indicating prefixes to the linearised data

sources. Even mixing vastly different datasets (slot-value pairs, triples, hierarchical encoding with dialogue acts), neither system component got confused by the structural differences, and benefitted instead. Mixed-corpus training is thus an efficient means of training a better language model.
Another convenience of mixed-corpus training, relevant to data-to-text generation, is that faithfulness among the dataset must only be ensured for the target set among the mixed corpus. Alien dataset consistency does not seem to contribute.
The unexpected performance increase from replacing semantics-based separator tokens with a uniform separator once again underlines the potential in dataset pre-processing. It also challenges the widespread perception that Transformer-based language models will always benefit from special tokens emulating syntactic structure.

Beyond the positive influence of cleaning, semantic de-noising and mixed-corpus training on generation, faithfulness among generations could be boosted using an external module - a semantic fidelity classifier, which selected one out of five generator outputs as the final output. (faithfulness being measured through automatic metrics here).
In our approach, we thus do not modify the architecture of the generator, in contrast to existing approaches. Our approach differs from Harkous *et al.*'s DataTuner in that we unlock greater potential of the generator through cleaning, semantic de-noising and joining corpora.

Finally, letting humans judge the end-to-end system's output, in comparison to the pre-filtered output, would have provided more insight into the effectiveness of the semantic fidelity classifier. Due to the small number of hallucinations found in the pre-filtering output, it would have taken a large sample size to establish a significant change. However, we might have used human judgement for checking whether the large number of omissions has decreased, since the RoBERTa semantic fidelity classifier has been trained to detect omission as one of its classes. The large-scale comparison of pre-filtering to post-filtering outputs is thus a further opportunity for evaluating the effectiveness of semantic fidelity classifier approaches.

We have observed the performance boost in our T5-generator when training examples were faithful. In answer to usually high levels of noise in data-to-text corpora, quick-to-use semantic cleaning methods, such as the procedure described for WikiInfo2Text (3.3), for different types of meaning encodings, are an opportunity for further research.

References

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

Castro Ferreira, Thiago et al. (2018). “Enriching the WebNLG corpus”. In: *Proceedings of the 11th International Conference on Natural Language Generation*. INLG’18. Tilburg, The Netherlands: Association for Computational Linguistics.

Chen, Shuang et al. (Nov. 2019). “Enhancing Neural Data-To-Text Generation Models with External Background Knowledge”. In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Hong Kong, China: Association for Computational Linguistics, pp. 3022–3032. DOI: 10.18653/v1/D19-1299. URL: https://aclanthology.org/D19-1299.

Chen, Wenhu et al. (2020). “KGPT: Knowledge-Grounded Pre-Training for Data-to-Text Generation”. In: *CoRR* abs/2010.02307. arXiv: 2010.02307. URL: https://arxiv.org/abs/2010.02307.

Chen, Zhiyu, Harini Eavani, Wenhu Chen, et al. (July 2020). “Few-Shot NLG with Pre-Trained Language Model”. In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online: Association for Computational Linguistics, pp. 183–190. DOI: 10.18653/v1/2020.acl-main.18. URL: https://aclanthology.org/2020.acl-main.18.

Chen, Zhiyu, Harini Eavani, Yinyin Liu, et al. (2019). “Few-shot NLG with Pre-trained Language Model”. In: *CoRR* abs/1904.09521. arXiv: 1904.09521. URL: http://arxiv.org/abs/1904.09521.

Colin, Emilie et al. (Sept. 2016). “The WebNLG Challenge: Generating Text from DBPedia Data”. In: *Proceedings of the 9th International Natural Language Generation conference*. Edinburgh, UK: Association for Computational Linguistics, pp. 163–167. DOI: 10.18653/v1/W16-6626. URL: https://aclanthology.org/W16-6626.

Dale, Edgar and Jeanne S. Chall (1948). “A Formula for Predicting Readability”. In: *Educational Research Bulletin* 27.1, pp. 11–28. ISSN: 15554023. URL: `http://www.jstor.org/stable/1473169`.

Dale, Robert (2019). “NLP commercialisation in the last 25 years”. In: *Natural Language Engineering* 25.3, pp. 419–426. DOI: `10.1017/S1351324919000135`.

Deerwester, Scott et al. (1990). “Indexing by latent semantic analysis”. In: *JOURNAL OF THE AMERICAN SOCIETY FOR INFORMATION SCIENCE* 41.6, pp. 391–407.

Devlin, Jacob et al. (2018). “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding”. In: *CoRR* abs/1810.04805. arXiv: `1810.04805`. URL: `http://arxiv.org/abs/1810.04805`.

Dhingra, Bhuwan et al. (2019). “Handling Divergent Reference Texts when Evaluating Table-to-Text Generation”. In: *CoRR* abs/1906.01081. arXiv: `1906.01081`. URL: `http://arxiv.org/abs/1906.01081`.

Gardent, Claire et al. (2017). “Creating Training Corpora for NLG Micro-Planners”. In: *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. ACL’17. Vancouver, Canada: Association for Computational Linguistics, pp. 179–188. DOI: `10.18653/v1/P17-1017`. URL: `http://www.aclweb.org/anthology/P17-1017`.

Hamilton, William L. et al. (2018). “Embedding Logical Queries on Knowledge Graphs”. In: *NeurIPS*.

Harkous, Hamza, Isabel Groves, and Amir Saffari (Dec. 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*. Barcelona, Spain (Online): International Committee on Computational Linguistics, pp. 2410–2424. DOI: `10.18653/v1/2020.coling-main.218`. URL: `https://aclanthology.org/2020.coling-main.218`.

Juraska, Juraj, Kevin K. Bowden, and Marilyn A. Walker (2019). “ViGGO: A Video Game Corpus for Data-To-Text Generation in Open-Domain Conversation”. In: *CoRR* abs/1910.12129. arXiv: `1910.12129`. URL: `http://arxiv.org/abs/1910.12129`.

Lavie, Alon and Abhaya Agarwal (June 2007). “METEOR: An Automatic Metric for MT Evaluation with High Levels of Correlation with Human Judgments”. In: *Proceedings of the Second Workshop on Statistical Machine*

Translation. Prague, Czech Republic: Association for Computational Linguistics, pp. 228–231. URL: `https://aclanthology.org/W07-0734`.

Le, Quoc and Tomas Mikolov (22–24 Jun 2014). "Distributed Representations of Sentences and Documents". In: *Proceedings of the 31st International Conference on Machine Learning*. Ed. by Eric P. Xing and Tony Jebara. Vol. 32. Proceedings of Machine Learning Research 2. Bejing, China: PMLR, pp. 1188–1196. URL: `http://proceedings.mlr.press/v32/le14.html`.

Li, Xiang Lisa and Percy Liang (2021). "Prefix-Tuning: Optimizing Continuous Prompts for Generation". In: *CoRR* abs/2101.00190. arXiv: `2101.00190`. URL: `https://arxiv.org/abs/2101.00190`.

Lin, Chin-Yew (July 2004). "ROUGE: A Package for Automatic Evaluation of Summaries". In: *Text Summarization Branches Out*. Barcelona, Spain: Association for Computational Linguistics, pp. 74–81. URL: `https://aclanthology.org/W04-1013`.

Liu, Pengfei et al. (2021). "Pre-train, Prompt, and Predict: A Systematic Survey of Prompting Methods in Natural Language Processing". In: *ArXiv* abs/2107.13586.

Liu, Yinhan et al. (2019). "RoBERTa: A Robustly Optimized BERT Pretraining Approach". In: *CoRR* abs/1907.11692. arXiv: `1907.11692`. URL: `http://arxiv.org/abs/1907.11692`.

MacCartney, Bill and Christopher D. Manning (Aug. 2008). "Modeling Semantic Containment and Exclusion in Natural Language Inference". In: *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*. Manchester, UK: Coling 2008 Organizing Committee, pp. 521–528. URL: `https://aclanthology.org/C08-1066`.

Mikolov, Tomas et al. (2013). "Efficient estimation of word representations in vector space". In: *arXiv preprint arXiv:1301.3781*.

Nie, Feng et al. (July 2019). "A Simple Recipe towards Reducing Hallucination in Neural Surface Realisation". In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy: Association for Computational Linguistics, pp. 2673–2679. DOI: `10.18653/v1/P19-1256`. URL: `https://aclanthology.org/P19-1256`.

Papineni, Kishore et al. (Oct. 2002). "BLEU: a Method for Automatic Evaluation of Machine Translation". In: DOI: `10.3115/1073083.1073135`.

Parikh, Ankur P. et al. (2020). "ToTTo: A Controlled Table-To-Text Generation Dataset". In: *CoRR* abs/2004.14373. arXiv: `2004.14373`. URL: `https://arxiv.org/abs/2004.14373`.

Pennington, Jeffrey, Richard Socher, and Christopher Manning (Jan. 2014). "Glove: Global Vectors for Word Representation". In: vol. 14, pp. 1532–1543. DOI: `10.3115/v1/D14-1162`.

Radford, Alec and Karthik Narasimhan (2018). "Improving Language Understanding by Generative Pre-Training". In:

Reiter, Ehud and Robert Dale (1997). "Building applied natural language generation systems". In: *Natural Language Engineering* 3.1, pp. 57–87. DOI: `10.1017/S1351324997001502`.

Ribeiro, Leonardo F. R. et al. (2020). "Investigating Pretrained Language Models for Graph-to-Text Generation". In: *CoRR* abs/2007.08426. arXiv: `2007.08426`. URL: `https://arxiv.org/abs/2007.08426`.

Rohrbach, Anna et al. (Oct. 2018). "Object Hallucination in Image Captioning". In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Brussels, Belgium: Association for Computational Linguistics, pp. 4035–4045. DOI: `10.18653/v1/D18-1437`. URL: `https://aclanthology.org/D18-1437`.

Shazeer, Noam and Mitchell Stern (2018). "Adafactor: Adaptive Learning Rates with Sublinear Memory Cost". In: *CoRR* abs/1804.04235. arXiv: `1804.04235`. URL: `http://arxiv.org/abs/1804.04235`.

Sutskever, Ilya, Oriol Vinyals, and Quoc V. Le (2014). "Sequence to Sequence Learning with Neural Networks". In: *CoRR* abs/1409.3215. arXiv: `1409.3215`. URL: `http://arxiv.org/abs/1409.3215`.

Vaswani, Ashish et al. (2017). "Attention is All You Need". In: *Proceedings of the 31st International Conference on Neural Information Processing Systems*. NIPS'17. Long Beach, California, USA: Curran Associates Inc., pp. 6000–6010. ISBN: 9781510860964.

Wiseman, Sam, S. Shieber, and Alexander M. Rush (2018). "Learning Neural Templates for Text Generation". In: *EMNLP*.

Zhu, Yukun et al. (2015). "Aligning Books and Movies: Towards Story-like Visual Explanations by Watching Movies and Reading Books". In: *CoRR* abs/1506.06724. arXiv: `1506.06724`. URL: `http://arxiv.org/abs/1506.06724`.