Utrecht University

Arria Data2Text Ltd

Master Thesis Project

Worth more than gold

A deep-dive model diagnostic on the mT5 in an attempt to uncover the origin of errors

---

Author: Simon van de Fliert (5708656)

1st Supervisor:	Dr. Albert Gatt	Utrecht University
1st Daily Supervisor:	Dr. Yaji Sripada	Arria Data2Text Ltd
2nd Daily Supervisor:	Dr. Ross Turner	Arria Data2Text Ltd
2nd Reader:	Dr. Guanyi Chen	Utrecht University

A thesis submitted in fulfillment of the requirements for the Utrecht University Master of Science degree in Artificial Intelligence

March 23, 2023

Abstract

Research within Natural Language Generation is evolving rapidly as new models are consistently reported to outperform previous works based on popular metrics, such as n-gram-based BLEU and ROUGE or model-based, such as BLUERT and BERTScore. Nonetheless, it is still unclear why state-of-the-art models make errors, thus making it difficult to identify a viable solution. This project aims to help fill this gap by performing model diagnostics on the mT5 transformer and attempting to uncover the origin of errors within Transformers. The mT5-base, T5-base, Yeb Havinga's T5-base Dutch-case, and BART models are fine-tuned on the RDF-to-text dataset called CACAPO. Following this, the mT5-base's generations are manually and automatically reviewed using evaluation metrics BLEU, METEOR, ROUGE, BERTScore, BARTScore, and PARENT. Additional experiments are conducted where CACAPO's training set is augmented, and underspecified input is provided with additional contextual information. These experiments led to several insights. First, observations are made involving CACAPO, where the reverse engineering nature highlighted the difficulties of capturing all contextually relevant data in the input. Furthermore, CACAPO often leaves relevant information from the reference text out of the input, as no attribute could be connected to the corresponding value. However, experimentation showed that this is too restricted and that CACAPO could be extended using inter-subject attributes. Furthermore, data augmentation experiments highlighted the need for a structured augmentation method for multilingual use cases. Following this, experimentation showed it to be more beneficial to add contextual data to underspecified inputs compared to augmenting the data. This also highlights the need for an improved content selection process, so that all contextually relevant information in the reference text is captured in the input data. Another observation was made where a model trained on improved input data performs better during the inference stage, even when the input data during inference was underspecified. This

indicates that purely improving the specificity of the training set could lower the number of errors made by the model. Moreover, comparisons between Dutch and English records showed that Dutch records improved more due to the additional input data, which could be caused by the amount of training data the model has seen during both pre-training and fine-tuning. This could highlight a correlation between improved contextual input data and the necessary training set size, where smaller data set sizes might be usable if the input data had all contextually relevant information. Another possible reason could be the difference in sentence complexity, where the majority of Dutch records are relatively simple, whereas a large part of English records is complex. This could indicate that an improved specification of input data could be more impactful for relatively simple texts, highlighted by the increase in performance for Dutch records, but lacking increase of BARTScore performance for the English records. Furthermore, analysis between languages showed no difference in error counts, showing that error types are consistent between English and Dutch records. However, the severity of these errors was not captured in this project. Finally, each model showed difficulty capturing the correct order of attributes, thereby generating incorrect conclusions. This is likely due to the lacking relational information in the CACAPO dataset for end-to-end models.

Acknowledgements

I would like to express my gratitude to my professor and supervisors Dr. Albert Gatt, Dr. Yaji Sripada, and Dr. Ross Turner for their support, patience, and guidance, which was integral to the completion of this project. I would also like to thank Dr. Guanyi Chen for his time and feedback supporting this research as second supervisor.

Lastly, I would like to thank my family for providing essential feedback and support during this project.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

"... is the new gold!". A common phrase that countless have uttered and many more will, as the availability and usefulness of products has and will keep fluctuating over our lifetime. Where it used to be steel, diamonds, and oil, over the recent decades a new commodity has risen above all else; Information (3). As our society benefits from the incredible advancements in technology, more and more information is becoming available, and in turn, is required to conduct operations. Take for example the advancements in journalism, where dozens of credible organizations need to quickly report on occurrences around the globe, or the vast world of entertainment, from sport to film, where hundreds of organizations write articles on their favourite celebrity, trend, genre, or game. With the rising need for documentation, organizations and researchers have looked towards Artificial Intelligence to support their endeavours, namely in the well-researched field of Natural Language Generation (NLG), which focuses on automatically generating information from a given input. This input can be based on text or data and depending on the goal and generation focus, a different output can be requested (4, 5). Advancements in technology and business requirements thus gave a new rise to the commercial use of NLG (6).

Within this rise is a continuous pursuit to improve the research and capabilities of the technology. This thesis, which was carried out in collaboration with Arria NLG - a leading company in the NLG field, intends to contribute to the efforts to improve the understanding and research in the NLG field.

1.1 Problem Definition

Within the field of NLG, the majority of research has focused on optimizing the generation quality of models, such as described in Puduppuly et al.'s paper (7), through tuning hyperparameters and modifying models with modules that seemed to work in the past, for example, the addition of a content selection module (8). Moreover, with the abundance of data, it was common to provide models with as much data as possible in the hopes of it generating relevant and high-quality texts, as data-to-text generation models have to take relevant data from the input data to focus on the generated sentences (9). Whilst these research paths are fruitful, they do not fully answer why NLG models make mistakes, thus rarely helping others better understand the workings of neural models themselves. With this in mind, this project diverges from the popular path and investigates the nature of state-of-the-art models, to better understand the actions of these models. Instead of optimizing the model with slight increments, a deeper understanding of the origin of errors is sought. As such, the research goal for this project is as follows:

> *To gain insights into the workings of state-of-the-art models and attempt to uncover the origin of generation errors.*

To achieve this goal, the state-of-the-art, pre-trained end-to-end neural models T5, mT5, and BART are fine-tuned on the data-to-text, bilingual dataset called CACAPO, after which they are evaluated with a combination of automatic metrics and manual evaluation. Following this, hypotheses will be set and tested through various experiments.

1.2 Thesis Outline

The paper continues with a literature review in Chapter 2, where the current state of the field of Natural Language Generation is described, alongside two current research directions, popular models, datasets, and evaluation methods. Then, Chapter 3 details the project setup, describing implemented models, the CACAPO dataset, the fine-tuning setup, and baseline evaluation results. Following this, Chapter 4 delves deeper into the executed experiments and their results, after which Chapter 5 describes the insights gained from and the limitations of the executed experiments. Lastly, this project will end with a conclusion and potential future works. The Appendix describes additional information, such as individual model performances and additional executed experiments.

CHAPTER 2

RELATED WORK

2.1 NLG

As described by Reiter et al., NLG is "the subfield of artificial intelligence and computational linguistics that is concerned with the construction of computer systems that can produce understandable texts in English or other human languages from some underlying non-linguistic representation of information"(10). Where Reiter et al. focused on the generation of linguistic texts through the use of non-linguistic data, a large body of research has also been conducted on deriving linguistic generations from linguistic texts. For example, focusing on text-to-text models, tasks such as machine translation, summarization, automatic generation of peer reviews for scientific papers (4), and text expansions (5) are widely researched. As the name suggests, a text-to-text model is trained to generate a linguistic text based on linguistic input (4). For example, a summarization model is trained on a dataset containing pairings of larger documents and corresponding summarizations, with which the model can learn to perform the intended task. Redirecting the focus to data-to-text models, tasks such as automatically creating weather (11) and financial reports (12), patient information summaries (13), and automatic news and sports coverage (14) have been researched.

In contrast to text-to-text models, a data-to-text model takes as input non-linguistic data and generates from that a linguistic text (4). The creation of a data-to-text model is the focus of this project.

After identifying what type of task to investigate, the focus then changes to how to transform the input data into a linguistic generation. Traditionally this task was separated into six smaller tasks. The first task, *content determination*, was to decide which information should be included in the generation (4). Each domain differs in this aspect. For example, the financial domain often has different priorities compared to the medical domain, thus different information would be needed to be shown in the generation. Following this, the creator had to determine the *text structuring* and, if possible, combine sentences to improve fluency, also called *sentence aggregation*. The next task would be to choose the right words, thereby determining the correct *lexicalisation* and then how to refer to the domain objects within the generation, called *referring expression generation*. Finally, the creator combines all previous tasks to create a fluent generation, called *linguistic realisation* (4). With the advancement of neural networks and deep learning in Artificial Intelligence, several previously mentioned tasks are combined, showing impressive results (15). These recent advancements have pushed the entire NLG research field in the direction of neural networks. For a more in-depth description of this advancement, refer to the following surveys (4, 5, 16).

2.2 State of the Art of NLG

To conduct data-to-text generation, researchers often build upon the best or state-of-the-art models. A wide variety of models have been proposed by previous works, as highlighted by Li et al. (17). Many of these models have been tested in a variety of domains and many have shown promising results, especially end-to-end models based on the Transformer architecture, an encoder-decoder model, which have become increasingly adept at combining content selection, the identification of what content from the input data is contextually relevant and required for the generation, and surface realization, the act of generating a text based on the selected content (15). Within this, the transformer models T5 and Bart showed promising results based on metrics such as ROUGE, METEOR, and BERTScore, and shown through manual evaluation to be able to generate realistic texts. (18, 19, 20). Furthermore, other researchers have adapted these models to their specific domain, resulting in more promising results, where the authors could generate more precise, informative, and factually consistent texts (21, 22, 23).

2.3 Transformers

But how do transformers work? The transformer architecture was first proposed by Vaswani et al. (1) in their paper *Attention is all you need*, where they built upon the encoder-decoder architecture, an architectural overview of which can be found in Figure 2.1. Their transformer consisted of both six encoder- and six decoder layers, where each of the six layers in the encoder module consists of an attention sub-layer and a feed-forward sub-layer, whilst each of the six layers in the decoder consists of two separate attention sub-layers and a feed-forward sub-layer. The size and amount of these layers are not fixed, and previous works often change the number of layers in each module, for example in (24).

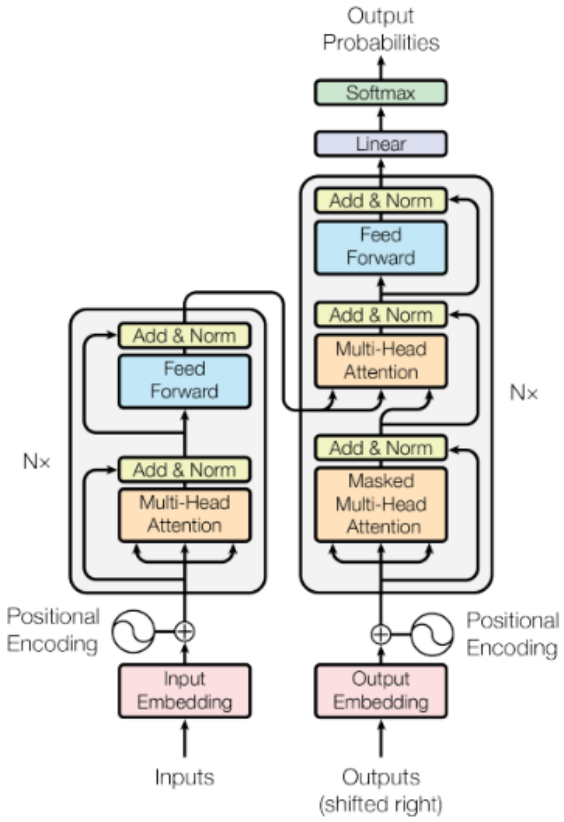

Figure 2.1: Transformer architecture proposed by Vaswani et al.(1)

Let's focus on the encoder, the left-hand block in Figure 2.1. First, words from the input are changed into embeddings, after which a positional encoding is given to each embedding.

This encoding lets the model keep track of the position of the token in the sentence. Then, the embedding is given to the multi-head attention layer. This layer is the breakthrough of Vaswani et al.'s paper, where the model gains the ability to calculate how much attention each word needs to receive in the context of the words around it. This layer performs multiple calculations. First, three vectors are calculated using the embeddings given to the layer, those being the *Query*, *Key*, and *Value* vectors. The query is the word or token the model has under consideration, whereas the Key metric refers to potential words to compare the query with, in this case, other words in the sentence. The value matrix can be thought of as the best matches returned from the comparison.

Following their calculation, the dot product is taken between the Query of the word in focus and the Key vector of each word in the sentence. Then, these values are divided by the square root of d, which stabilizes the gradients, after which the softmax of each value is taken. This results in all scores adding up to one, and when multiplied with the Value vector, will result in low-scoring words being drowned out, whilst high-scoring words that the model would want to focus on are pronounced even more. Finally, the sum of the weighted value vectors is taken, resulting in one value. This results in formula 2.1.

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V \tag{2.1}$$

However, by using a softmax in the formula, the likelihood that a single token obtains most of the focus is high, which is not ideal, as it can be useful to keep track of other tokens in the sentence. The solution Vaswani et al. found was to not have a single head calculate the attention but to have several heads running simultaneously, thus allowing the model to place attention at different points in the sentence. However, formula 2.1 generates a single matrix, and if this is done with multiple heads, it would mean multiple matrices are stored. This is an issue, as the Feed Forward Neural Network only accepts a single matrix, which corresponds with the vectors of each word in the input. Thus the attention scores need to be concatenated into a single matrix and then multiplied with a weight matrix to obtain a single attention matrix. This is shown in the formula 2.2 below.

$$MultiHead(Q, K, V) = Concat(head_1, ..., head_h)W^O \tag{2.2}$$

This attention score is then given to the feed-forward layer, thereby letting it learn from the attention score. Finally, the encoder ends with the three Q, K, and V matrices. The K and V matrices are given to the decoder, which uses them to focus on the correct words

in the sentence. The decoder works similarly to the encoder, but where the encoder only takes in the words from the input sentence, the decoder also takes previously generated words. However, these inputs are not used at the same time. First, the decoder takes in the most recent, previously outputted word and gives it to the Masked Multi-Head Attention layer. This layer then calculates the attention score of the outputted word, similarly to the encoder. Then the word with the attention score is given to the second attention layer, alongside the output of the encoder. This second attention layer calculates the attention with both inputs, combining the knowledge of the already outputted sentence alongside the knowledge of the encoder, thereby giving the decoder more information regarding the token in question. This updated attention score is then given to a feed-forward layer (1). Finally, the decoder passes its information to the linear layer. This layer transforms the matrices into a logit layer, after which the following softmax layer transforms each logit value, such that the sum of all values is one. This once again drowns out poor choices, and the cell with the highest value is chosen. This cell is connected with a word out of the model's vocabulary, and the chosen cell is translated into said word.

2.4 Limitations of Transformers

Whilst transformers show great results, these models are not perfect. First, transformers still struggle with hallucinated content, thus providing generated text with information that does not exist in the given input (25, 26). Furthermore, transformer-generated texts can also suffer from both missing information (25, 26) and duplicate information (26). Lastly, transformers occasionally struggle with providing fluent generations, which could cause users to misinterpret the information provided (25, 26). All these issues result in the model generating less relevant data from the input data, thus lowering the quality of the generated texts. These limitations are the driving force for the research goal described in section 1.1. By getting a better understanding of the origin of these errors, effective measures can be taken to counteract these pitfalls.

2.5 Known Research Directions

Ever since these limitations have been known, researchers have been investigating ways to enhance the performance of transformers for generation tasks. Within this, two research focus areas have emerged. Some researchers focus on the input of the transformer and the

encoder, whilst others turned their attention toward the decoder in an attempt to facilitate better and more fluent generations. Each direction of focus will be briefly discussed below.

2.5.1 Encoder Investigation

As shown before, transformers proved to be a great advancement in text generation, however where they succeeded in verbalizing a structured text, they struggle with the ordering and structuring of data (27) and often consider too much noisy data. This struggle led some researchers to look into the past, where they questioned whether the transition from pipelines of specific modules to end-to-end models generates better text. For example, Ferreira et al. (27) created several models from both an end-to-end and pipeline architecture. Some models were fully divided where each aspect of the pipeline had its own module, other models had a combination of separated models feeding into an end-to-end system, and finally, they also included a stand-alone end-to-end model. They then trained the models on RDFs (Resource Description Frameworks) extracted from the WebNLG corpus (28) and compared the models. They found that their neural pipeline approaches performed better than fully end-to-end models, where the difference is most clear in unseen domains, as the performance of the end-to-end model drops off significantly (27). Similar to Ferreira, Puduppuly et al. wanted to explicitly define the content structure and ordering steps before giving it to an end-to-end model. With the addition of a content planner, the authors argue that the decoder will only have to deal with sentence planning and surface realization, which are easier tasks to conduct. Furthermore, as the content plan is less likely to contain redundant information, the output will be of higher quality (8). The authors implement an LSTM model to encode the input and a bidirectional LSTM as the decoder. Introducing a plan isn't unique to Pudupully et al.'s paper, as several other researchers have proposed to introduce a plan to facilitate the ordering of inputs and enrich the input data (29).

Harkous et al. (30) attempted to circumvent the transformers' weakness of producing lower quality generations when used on out-of-domain data by combining a full end-to-end system with a semantic fidelity classifier, which facilitates the end-to-end system by removing poor generations, such as hallucinations. The first aspect of this 2-stage approach, the generator, is based on the pre-trained OpenAI GPT-2 model, whilst the latter reranker implements RoBERTa as a classifier. The addition of the classifier proves to improve semantic evaluation significantly (30).

However, where previous results show good performance, Puduppuly et al. found that generations proved imprecise at document level (7). To remedy this the authors propose to add a content planner on a macro level, structuring the content for multi-paragraph documents. Such a macro plan would consist of a sequence of paragraphs split by a discourse marker $< P >$. Once the plan is created, the authors encode the plan with a BiLSTM and decode using an LSTM (7).

Continuing the trend of pipeline modules, Kasner et al. (31) identify the costly requirements of collecting domain data and the decreasing performance of transformers in out-of-domain situations. Due to this, they proposed to create a pipeline where pre-trained language models are used as modules. Each module would consist of a separate language model that was trained on specific lexical operations corresponding with its module. The authors argue that this pipeline allows users to produce more semantically consistent output with fewer input data (31).

Taking a different approach, Yang et al. (32) focus on the transformers themselves. They propose to improve the generations of transformers by first improving the model's ability to learn the semantic representations of the input. They then add the field and position information from the input into the target text, thus allowing the model to copy the contents of the input into the output. The authors show that these advancements prove an advancement of previous analysed models (32). Like Yang et al., Rebuffel et al. focused their work on the transformer itself (33). The authors argue that by forcing the input into a fixed-size vector representation by encoding it sequentially, the encoding loses important information that could help the transformer in its content selection. The authors propose adding another encoder module to the transformer. This addition allows the low-level encoder to focus on record embeddings and the high-level encoder to encode the data structure of the input. The authors show through experimentation that this addition shows improvement, especially regarding content selection, however, it does not unequivalently outperform all tested models (33).

2.5.2 Decoder Investigation

A different focus within research is to enhance the decoder's ability to transform embeddings into generated text. With the advancements in language models, the initial decoders focused on generating the most probable sequence of words, which is referred to as the maximum a posteriori (MAP) decoding (34, 35). This task is also called a search problem

(35), as the decoder has to search over the entire search space, defined by the vocabulary size, to find a suitable candidate token to generate. As this search space often proved to be too large to practically search (36), researchers developed several heuristics to make the task practical. The first heuristic is called *Greedy Search*, where the decoder always chooses the token with the highest probability (34, 35, 36). However, this quickly proved inadequate, as it would produce repetitive sentences (35). To remedy this issue, researchers created a second heuristic, called *Beam Search*. This extension takes several tokens into account, where the parameter k allows the researcher to extend the number of tokens the model is allowed to consider (34, 35, 36). Nevertheless, once the parameter k has been set, it will remain constant. This rigidness proved to be problematic for certain tasks that require more dynamic token choices. Due to this, many variations have been proposed, such as *Group diverse Beam Search (GroupBS)* (36), *Diverse Beam Search (DBS)* (34), or other variations in the stopping criteria, coverage, and pruning thresholds referenced by Zarrieß et al. (35).

Whilst these heuristics proved to be an improvement, the resulting generations often lacked diversity (36) because the search only returns the top k tokens, thereby only allowing the model to consider these. To add diversity to this search problem, researchers proposed sampling tokens from the search space and adding these to the tokens under consideration. However, by randomly selecting tokens for diversity, researchers found that the model's generations drastically decreased in logic and fluency (34). This was caused by infrequent words replacing words with significantly higher probabilities. To remedy this issue, researchers proposed the *top-k sampling* method, (34, 35) where the sampling was conducted in a truncated search space, thus increasing diversity without adding infrequent words into the selection. However, this still showed issues, as some cases require a larger search space to choose tokens from, whilst other cases require fewer tokens. The top-k sampling method fixed the search space from which a token could be sampled, which in some cases caused redundant sampling (34, 35, 37). To remedy this issue, the *top-p (or nucleus) sampling* method was proposed, which allowed the model to dynamically alter the sampling space (34, 35, 37). This quickly became the default choice for many researchers, however, whilst a sampling method improves the diversity in a text, it does lower the verifiability (factuality or correctness) of texts, as a less-frequent token might be sampled that changes the conclusion of the generation (36). For example, if a token is sampled in the place of a Named Entity, then the correctness of the generation would suffer. To remedy this trade-off, researchers often combine search and sampling in their models. For

example, Massarelli et al. proposed a hybrid where they first sample a set of tokens and use Beam Search on the tokens to select tokens. The smaller the sample set, the closer it is to Beam Search (36).

Due to these advancements, the default choice in research often fell to choose a variation of Beam Search in combination with top-p sampling (35). However, Wiher et al. (34) argue that this should not be the case. They argue that whilst Beam Search and Nucleus Sampling proved fruitful, these methods only outperform on specific tasks, thus they caution taking such methods across different tasks without testing. Moreover, whilst methods such as Beam Search are often used in research, a worrying trend has risen where parameters are under-reported (35). This results in a lower degree of reproducibility, which makes it more difficult to find the best sampling and search methods for the designated task.

Furthermore, whilst the main focus for decoders was on search and sampling methods, Welleck et al. (38) argue that the actual use of any search method results in degenerate sentences full of mistakes, repetitions, and hallucinations because these methods are not constrained enough. The authors propose a new training method, called *Unlikelihood training*, where instead of focusing on the highest probability tokens, the focus is changed to decrease the model's probability of certain tokens called *negative candidates*. The authors argue that by minimizing the unlikelihood loss it will become less likely to see both incorrect repeating tokens and frequent tokens. This method should prove an advancement on both the token and sequence level (38).

2.6 Potential Datasets

After choosing a research direction and model for experimentation the creator will need to carefully choose the data with which the model will be trained.

First, the choice needs to be made whether to pre-train a new model or fine-tune an already pre-trained model on a downstream task. Pre-training a model generally requires a lot of data and can require a lot of resources to train, whereas creators can also choose to take an existing pre-trained model, take advantage of its knowledge, and fine-tune it on a specific downstream task, thereby attempting to specialize the model. The latter is more resource-efficient, but it limits the creator to a specific architecture and model type, as these models need to have been pre-trained by someone else beforehand.

These considerations might push the creator to start from scratch, thus pre-training a model. However, in the case of NLG, many variations have already been created, including the models mentioned in section 2.2. Thus this project can leverage these models to fine-tune them on a specific downstream task.

Depending on the task at hand, a different dataset will be required. For example, if the model will need to summarize large documents, then it will need to be trained on a dataset containing a combination of large documents and corresponding summaries. On the other hand, if the model's goal is to produce a linguistic description of non-linguistic patterns (e.g. a description of statistical patterns), then the dataset will need to contain both the non-linguistic data as input and its corresponding linguistic description as requested output, thereby giving the model a goal to learn (2, 4, 5). However, do these datasets exist? Several datasets are described below and a summarization is shown in table 2.1.

When looking at publicly available financial datasets, numerous datasets can be found. Take for example the linguistic-based dataset of financial news articles of (39), which contains the daily financial news of over 6000 companies between 2009 and 2020. This source also contains a ready-to-use scraper which was used to collect the articles. Another example also comes from Kaggle (40), which contains both news and opinion articles alongside their release dates, which have been initially collected to analyse the relationship between price movements and stock news.

Other datasets have a combination of both numerical and financial news from stocks. Take, for example, the Kaggle dataset of (41), which contains large sets of scraped news articles and sets of non-linguistic data describing the stock at the time, such as its price at its opening, closing, high, and lows.

Another promising dataset is the FinQA dataset of Chen et al. (42), which contains 8,281 financial QA pairs extracted from annual reports, where each QA pair contains information important enough that readers look for it. Abdaljalil et al. (43) published their paper which uses a public dataset of annual reports retrieved from UK firms listed on The London Stock Exchange (LSE). These annual reports are paired with 3 to 4 human written summaries that serve as the gold standard, thus making the dataset fantastic for summarization tasks. Another dataset widely used in many research papers was introduced by Vargas et al. (44), which contained 106.494 news articles from the Reuters news website, focussing on financial news published between 20 October 2006 and 21 November 2013. However, due to copyright, this dataset cannot be used anymore.

Extending the search for datasets outside financial data, several other widely used datasets come to the forefront. For example, several datasets have leveraged Wikipedia for its collection of information in a variety of domains. Of these, the most popular are ToTTo (45), which crowdsourced thousands of tables from Wikipedia and provides generated sentences describing the contents of said tables, and the enriched WebNLG (46), in which Ferreira et al. transformed contents from Wikipedia into triples of resource description format (RDFs) and added these to the original WebNLG dataset (47), thus combining data from both DBpedia and Wikipedia into one dataset. Models trained on crowdsourced datasets tend to suffer less from noise and inaccuracies, as the data is a direct verbalization of the aligned data (2). However, crowdsourcing also comes with its own style, which differs substantially from the writing style of professionals. For example, a reporter has to keep their text interesting and varied, with the goal to capture and keep the interest of the reader, whilst a crowdsourced writer will lack this desire. This has resulted in these datasets receiving critique for lacking real-world representativity (2).

Due to these concerns, researchers have attempted different sourcing methods, such as scraping and crawling information from websites. For example, Kantharaj et al., the authors of Chart2Text (48), extracted thousands of tables and their descriptions from *Statista* and *Pew*, whilst Cheng et al. extracted complex hierarchal tables from *Statistics Canada* and *National Science Foundation* (49), both to create a dataset for chart to text generation tasks. Looking further, sites such as Yelp provide a wealth of human-written reviews, for example on restaurants. Datasets such as YelpNLG (50) and E2E (51) leverage this vast source of information to create datasets that challenge neural and end-to-end generation systems with syntactically diverse and lexical rich texts and with extracting meaning representations, which force the users of these datasets to think about scalability, content selection and dealing with human-like, diverse texts (50, 51). Going further, one widely used domain in research both due to its complexity and general availability is the domain of sports. With millions of avid and passionate followers around the globe, thousands of articles and statistics are kept on sports and are stored in articles, tables, and images. Take for example the RotoWire dataset (52), which contains thousands of articles and statistics on basketball games, or the MLB dataset (53), where Puduppully et al. collected thousands of MLB statistics and paired them with human written summaries. Both datasets highlight the variety in which data is stored and how they can be used for a wide variety of tasks, such table to text generation and summarization tasks.

However, whilst scraping and web crawling offers a better variety of texts, it does have their limitations. First, many of the currently available datasets that are built through scraping contain document-level texts (2). This forcibly excludes generation tasks that require short sentences or phrase-level texts. For example, a chatbot model will likely have more success training on short sentences and phrases than long document-level texts. Second, many datasets also cover a single domain, such as restaurants (YelpNLG), basketball (RotoWire), and baseball (MLB), thereby decreasing the generalizability of the model across different domains. Third, the majority of datasets contain a single language, that being English (2).

To remedy these limitations, authors have taken different approaches. One such approach is to combine several known datasets into one. For example, Nan et al. (54) created DART, an open domain structured **DA**ta **R**ecord to **T**ext generation dataset, by combining records from a cleaned E2E, WebNLG, WikiSQL and WikiTables. The authors argue that their framework allows for the creation of more challenging datasets whilst maintaining the ontology of the original datasets. Another approach is to start with varied texts and ensure that previously mentioned limitations are less prevalent. For example, van der Lee et al. create a dataset called CACAPO, short for **C**ombinations of **A**ligned Data-Senten**c**es from N**a**turally **P**r**o**duced Texts (2). Van der Lee et al. reverse-engineered the dataset, extracting non-linguistic information from articles describing finance, incidents, weather, and sports and transformed these articles into sentence-level texts. By doing so, the authors claim to have created one of the first scraped sentence-level, bilingual, multidomain datasets that is viable for both pipeline and end-to-end systems and thus fills the gap between the previously mentioned datasets. The dataset contains the chosen articles, a set of tokenized words corresponding with the article, and a set of annotated words. Some example annotated words are amountNumber, companyName, moneyAmount, and stockChangePercentage (the full list can be found in Appendix B), and the full version can be found on Dataverse (2).

If none of the previously mentioned datasets are deemed fit for the task and no other existing and fitting dataset can be found, then it is also possible to create a new dataset, which is a time-consuming task. A new dataset can be created by developing or using an already existing scraping algorithm that scrapes information. For example, financial articles can easily be collected from popular financial news websites such as Financial Times, Wall Street Journal, MarketWatch and Morning Star. Following this, non-linguistic

data must be paired with the target text, thus creating an input-output pairing, where the non-linguistic data would be the input and the linguistic data would be the output. Several methods can be used when extracting numerical data. For example, following Lee et al. (2) example, the numerical data can be extracted directly from the article and manually annotated. Another method, although susceptible to incorrectness and biases, is to scrape the non-linguistic data from other websites, such as Yahoo Finance, and manually pair this data with the articles. Finally, it is also possible to combine different datasets, for example, a linguistic dataset with a non-linguistic dataset. Here the creator could filter the data based on date and create pairings on non-linguistic data as input and financial texts as output. Combining different datasets also increases the risk of introducing faulty pairings, which can negatively impact the model's performance.

Note that the majority of datasets mentioned here purely contain English records, whilst only a few mentioned above are bilingual or even multilingual. This highlights a significant bottleneck in current research, where a majority of papers published only evaluate English texts. For example, Ruder et al. evaluated 461 of the 779 ACL papers published in 2021 and found that roughly 70 per cent of the 461 papers only evaluate English texts (55). Moreover, the authors found that a majority of research in multilingualism only considers accuracy metrics, whilst leaving out metrics such as fairness or interpretability. Ruder et al.'s results highlight a focus on English texts and datasets, which in turn shows that less common languages are underrepresented within research. Moreover, there lies an opportunity to test different metrics on underrepresented languages. Furthermore, as researchers for this project know both English and Dutch, it offers an opportunity to leverage a bilingual or multilingual dataset for this project, which could contribute to the lacking representation of languages and use of metrics outside of accuracy.

Dataset	Contains	Size	Multi-domain	Lang	Availability	Location
Daily Financial News for 6000+ Stocks (39)	Stock ticker, News headline, News article, Publisher for 6000 stocks from 2009-2020	4m	N	EN	Public	Kaggle

Historical financial news archive (40)	the historical news archive for the last 12 years of the US equities publicly traded on NYSE/NASDAQ which still has a price higher than 10$ per share.	222k	N	EN	Public	Kaggle
Financial Markets DataSet-Prices and News (41)	Stock Highs, Lows, Start and End, Financial News	5k	N	EN	Public	Kaggle
CACAPO (2)	News on Dutch and English sport, stocks, weather, and incident, contains annotated non-linguistic data retrieved from aforementioned news	20K	Y	EN, NL	Public	Dataverse
FinQA (42)	QA pairs based on the earnings reports of S&P 500 companies	8.3k	N	EN	Public	Github
RotoWire (52)	Articles describing basketball games, paired with box and line score tables	4.9k	N	EN	Public	Github
MLB (53)	MLB statistics paired with human written summaries	26.3K	N	EN	Public	Github
2020 FNS Shared task Dataset (43)	UK annual reports published in PDF file format, with documents around 80 pages on average, some annual reports could span over more than 250 pages, while the summary length should not exceed 1000 words	3.8k	N	EN	Public	Unknown

YelpNLG (50)	The corpus consists of 300,000 MR-to-NL (meaning representation to natural language reference) created using freely available restaurant reviews from Yelp.	300k	N	EN	Public	NLDS Corpora
E2E (51)	Texts and dialogue-act-based MR regarding restaurants	50k	N	EN	Public	Github and Website
ToTTo (45)	Wikipedia tables and corresponding generated sentences of these tables	120K	Y	EN	Public	Github
Chart2Text (48)	Charts with corresponding textual descriptions and summaries covering a broad range of topics and a variety of chart types.	44k	Y	EN	Public	Github
WebNLG (46)	Sets of Resource Description Framework (RDF) triples in 15 domains, in which, each one of them is formed by a Subject, Predicate and Object. The Subject and Object are constants or Wikipedia entities, whereas predicates represent a binary relation between these two elements in the triple.	25k	Y	EN, RU, DE	Public	Github
HiTab (49)	A set of complex hierarchal tables with annotated QA pairs that simulate natural language usage	3.5k	Y	EN	Public	Github
DART (54)	an open domain structured DAta Record to Text generation datase	82k	Y	EN	Public	Github

Table 2.1: A brief, non-exhaustive overview of available datasets for NLG.

2.7 Evaluation Methods

The act of evaluating a generated text has seen a lot of research and a wide variety of methods have been proposed. The main cause for this is the information these generation models deal with. First, there is no standardised input type or format for generation systems. Depending on the chosen input, some proposed evaluation methods would need significant changes to extract the necessary information, which would require a lot of effort. Second, there can be an enormous amount of variation in the format of open-ended texts. Evaluating such texts is difficult: a generation which is good but which differs too much from the template might be incorrectly deemed a bad generation - an undesirable outcome (4)

The cause for variation does not end there though, as depending on how a system is evaluated, a different set of evaluation metrics could be considered. A system can be evaluated intrinsically, where a focus is placed on measuring the performance of the system whilst ignoring other aspects, or it can be evaluated extrinsically, where the focus is placed on whether the system achieves its goal or purpose (4, 56).

Looking further, intrinsic evaluation methods can be separated into human evaluation and using automatic metrics. With human evaluation, both experts and non-experts can be consulted and asked to evaluate the system through common metrics. Several metrics are used in the literature, such as *fluency*, *accuracy*, and *relevance* (4), however many metrics are often mentioned under different names. For example, some works (like Harkous et al. (30)) describe *semantic fidelity*, whilst others refer to the metric as *factuality* and *consistency* (57). In the present work, the term *fluency* refers to whether the generated text is readable, grammatical, and coherent (57). Furthermore, when speaking of *faithfulness*, it refers to how accurately the generation matches with the information from the input data (30, 57). For example, mistakes such as hallucinations lower the faithfulness of the generations. Moreover, the metric *informativeness* is often described under different terms. In this paper, informativeness refers to how diverse and specific the generated text is whilst also considering how little redundant, general, or meaningless text it contains. The less redundant information the generation contains, the more informative the generation would be considered (57).

Besides testing the generations, it is also possible to test the evaluators with the use of metrics such as *per cent agreement*, *Cohen's k*, *Fleiss'* κ, and *Krippendorff's* α, which measure how well the evaluators agree with each other. Having a high inter-evaluator agreement indicates that the task is well-defined and the points of interest are consistently noticed. However, it is not necessarily the best idea to maximize this metric due to the possible variations in natural language (56). Whilst human evaluation is seen as the gold

standard, it is not an all-encompassing evaluation method. First, finding evaluators can be expensive and the evaluations time-consuming to run, even with websites such as Amazon's Mechanical Turk (56). Second, whilst humans are reliable at identifying certain metrics, such as fluency, they are often unable to detect diversity (56, 58). To alleviate this issue, Hashimoto et al. designed a metric combining human evaluation and statistics to capture both fluency through human evaluation and diversity through statistics using the model's sampling probabilities. This metric is called Human Unified with Statistical Evaluation, *HUSE* for short (58).

To support human evaluation and offer more cost-effective evaluation methods, researchers looked into providing automatic metrics. Some metrics are specialized in evaluating a model on one metric, whilst others have been created as a general evaluator. Other metrics often use n-grams to measure how well the output contains information obtained from the given input (4). Examples of these methods are *BLEU* (59) and *ROUGE* (24), which have been widely used in research, such as in (20, 21, 22, 23). However, metrics using n-grams have recently received critique, as the results of these metrics can differ from human evaluation, especially when the generated text is of higher quality (60). Furthermore, they struggle to evaluate longer sentences, as the longer the sentence, the less likely there will be a precise match of words (4, 60). This can lead to cases where the automatic metric evaluates a good generation as bad, whilst human evaluators could evaluate the same generation as good. Moreover, Novikova et al. (60) analysed why these automatic metrics can perform so poorly, and they found that these metrics perform far better when human evaluation deemed informativeness and naturalness of the generated text as low, compared to when the human evaluation deemed a generated text to be of high quality. Following their findings, a shift is slowly set in motion, where more neural-based evaluation metrics are used, such as in Yermakov et al. (18). However, it is important to note that n-gram metrics could still prove useful when evaluating poorly generated texts, as argued by Novikova et al. (60).

When performing an extrinsic evaluation, a focus is placed on studies that are questionnaire-based or performed by user volition. These often rely on an earlier set goal. Whilst these studies often proved useful, they also come with a significant time- and cost commitment, whilst they also need excellent supervision to control for confounding and intervening variables (4).

CHAPTER 3

METHODOLOGY

As described in section 2.5.1, the majority of existing research focuses on improving large language models (LLM) by altering model architectures, creating model pipelines, or adding modules to existing models. Whilst this could result in improved performance, the origin of errors made by LLMs often remains unknown. Uncovering these origins can help add focus to research seeking to resolve specific generation errors. This project attempts to uncover some possible origins of common errors in LLM output.

3.1 Models

First, a model is chosen which will be used for testing purposes. As described in 2, the T5 and BART models performed as state-of-the-art models. These models are available in several sizes and also in multilingual versions, offering a wide flexibility for this research project. However, the BART family is less available than the T5 model, where BART is only available in its base and large versions, whilst mBART, the multilingual version, is only available in a large format. The T5 family is available from small to XXL, for both the original and the multilingual variants. Given that neither model consistently outperforms the other, the choice was made to focus on the T5 family, as this would offer consistent size comparisons within the project's limited resources.

Meta's BART base model (3.1.4) was trained as a cross-reference comparison to the T5 family. These models are retrieved from Hugging Face (61) and are described in more detail below. The training results for each model are shown in Appendix A.

3.1.1 T5-base

The T5 model, created by Raffel et al. (62) closely follows the original transformer designed by Vaswani et al. (1). The authors slightly adapted the architecture, as they changed the original sinusoidal position signal and use a simplified position embedding where a scalar is added to the logit of the corresponding word. This positional embedding is shared across all layers for efficiency, however, each attention head has a different learned position embedding.

The data on which the model has been pre-trained is taken from Common Crawl, a public website that offers web-extracted texts which have been cleaned from any impurities, such as HTML and non-text content. This results in 20TB of data, but much of this data is made up of less interesting content, such as restaurant menus and duplicate texts, or unwanted content, such as offensive language (62). The authors, therefore, took the extracted content from April 2019 and applied several additional cleaning methods, such as the removal of content with too little textual data, content with offensive language, content without proper terminal punctuation, any content containing filler phrase "lorem ipsum", and any content that was not mostly English. This resulted in 750GB of data dubbed **C**olossal **C**lean **C**rawled **C**orpus, or C4 for short.

The authors then fine-tuned the T5 model on several downstream tasks, including text summarization and machine translation. For these tasks, the authors used the datasets described by the GLUE and SUPERGLUE benchmarks. Due to this fine-tuning, the T5 model can recognize English, French, Romanian, and German texts (62).

3.1.2 Yeb Havinga's T5-base-dutch-cased

To test the multilingualism in this project, a focus was set on adding a purely monolingual model. For this, a variation to the T5 model was found, namely, the *yhavinga/t5-v1.1-base-dutch-cased* retrieved from Hugging Face. This model has a slightly different architecture from the base, where instead of a ReLU activation function, it uses a Gated-ReLU. Furthermore, instead of the original C4 training set, the creator has filtered the C4 dataset for Dutch texts instead of English texts, thus resulting in a C4-Dutch variant. This resulted in the largest Dutch Dataset available, consisting of 151 GB of data (63). Unlike the original T5, this model has not been fine-tuned on downstream tasks, which is left as future work by the creators.

3.1.3 mT5-base

Another variant on the T5 model is Google's mT5 model, where the authors attempted to create a multilingual version of the T5 whilst keeping the recipe as close as possible to the

original. This model also changes the activation function into a Gated-ReLU instead of the original ReLU function. Moreover, instead of being pre-trained on a single language such as Havinga's model, the mT5 model is trained on 101 languages, including English and Dutch. Similar to Havinga's model, this model has also not been fine-tuned on downstream tasks (64).

3.1.4 BART-base

BART, short for Bidirectional and Auto-Regressive Transformers, is a denoising autoencoder that is pre-trained on masked, or corrupted, texts and given the task to reconstruct this text using a sequence-to-sequence model. The base version contains six different layers of encoders and decoders. Similar to T5, BART is based on the standard Transformer architecture proposed by Vaswani et al. (1), except the ReLU activation functions are changed to GeLU activation functions (65).

3.2 Dataset

As mentioned in section 2.6, van der Lee et al. created CACAPO to fill current gaps in the literature, where CACAPO is one of the first scraped sentence-level, bilingual, multidomain datasets. Taking a deeper look into this dataset, it was created by scraping news articles from a multitude of domain-specific websites. The dataset covers four different domains in both English and Dutch, those being *weather*, *sports*, *stocks* and *incidents* (such as traffic accidents and gun violence). Whilst the domains are the same for both languages, the scraped texts differ in their events and topics. For example, due to the (general) absence of guns in the Netherlands, Dutch articles are about traffic-related incidents, whilst the English incident section contains more gun-related articles. After the initial scraping, the dataset contained $51,575$ texts.

Following the first collection phase, the authors purged any text with more than 325 words, as previous research showed that basic news reports normally do not exceed that amount, resulting in the original $51,575$ being reduced to $20,630$ texts. The authors then randomly selected 200 texts per domain and language to obtain a representative number of sentences in the dataset. The resulting 1600 texts were then split into individual sentences, resulting in the final version of CACAPO containing $20,149$ sentences. A full dataset overview is shown in table 3.1.

Domains	Contains	Sentences #
Dutch Sports	Soccer match reports from the 2015/2016 and 2016/2017 seasons of the Dutch Eredivisie, the highest professional soccer league in The Netherlands.	2,559
Dutch Stocks	Daily reports on stock exchanges, company stock listings, (crypto)currency exchange rates, and oil prices.	3,022
Dutch Weather	Several daily short-term weather forecasts for The Netherlands from the Royal Netherlands Meteorological Institute (KNMI)	2,668
Dutch Incidents	News articles about traffic incidents	1,468
English Sports	Baseball reports from the American MLB League, the top league in American professional baseball	4,411
English Stocks	Daily reports on stock exchanges, company stock listings, (crypto)currency exchange rates, and oil prices	2,197
English Weather	Weather forecasts for several countries (e.g., Canada, United States, India, Ireland).	2,443
English Incidents	Gun violence incidents from the Gun Violence Archive	1,381
Total All domains		20,149

Table 3.1: An overview of CACAPO

Following this, the sentences were tokenized and manually annotated by two experts who reached an agreement level of 70.92%. As the dataset only contained journalistic texts, the choice was made to create the labels and attributes that follow the *5W, 1H* rule, that being the *Who*, *What*, *When*, *Where*, *Why*, and *How* (2). A full overview of the chosen attributes for each domain can be found in Appendix B.

To highlight the advantage of this dataset, the authors also compared it to the existing Enriched WebNLG dataset, as this dataset is also multilingual, multidomain, and built with different architectures in mind. However, they differ in collection method, as the Enriched WebNLG dataset was crowdsourced, whilst CACAPO has been scraped (2). First, the size of both datasets was compared, where the CACAPO dataset is slightly bigger than the Enriched WebNLG dataset in size in terms of instances and unique meaning representations, however in turn the Enriched WebNLG dataset contained more references to the meaning representations, shown in table 3.2.

	No. of instances	No. of unique MRs	Refs/MR	Slots/MR	W/Inst	W/Sent	Sents/Int
CACAPO (Dutch)	10,486	8,883	1.19 (1-285)	2.74	15.19	15.19	1 (1-1)
CACAPO (English)	10,566	9,352	1.17 (1-290)	2.83	18.52	18.52	1 (1-1)
WebNLG (English)	9,674	9,604	2.63 (1-12)	2.95	20.03	14.26	1.4 (1-6)
WebNLG (German)	7,812	7,753	2.63 (1-12)	2.96	19.22	13.64	1.4 (1-6)

Table 3.2: Size and Meaning Representation comparison between CACAPO and WebNLG (2)

The increase in reference to meaning representation, shown in column 3 of table 3.2 allows data-to-text systems to have an easier time learning alignments between meaning representations and text when trained on the Enriched WebNLG dataset (2). Second, whilst CACAPO contained more sentences than WebNLG, the sentences themselves are smaller in size. However, the authors found that CACAPO is more lexically diverse than the WebNLG dataset, as shown in figure 3.3.

	Tokens	Types	LS	TTR	MSTTR
CACAPO (Dutch)	147,770	10,152	0.87	0.07	0.87
CACAPO (English)	175,860	11,485	0.87	0.07	0.89
WebNLG (English)	491,731	5,521	0.84	0.01	0.75
WebNLG (German)	376,184	6,433	0.86	0.02	0.78

Table 3.3: Lexical diversity comparison between CACAPO and WebNLG, where LS refers to Lexical Sophistication, TTR refers to type-token ratio, and MSTTR refers to mean segmental TTR (2)

Furthermore, the authors also compared the sentence complexity by calculating the revised Developmental Level scale. This scale consists of eight different levels, zero being the simplest and seven being the most complex, where complexity is influenced by complex syntactic structures and referring expressions. The comparison between CACAPO and the Enriched WebNLG resulted in the Dutch texts in CACAPO being the simplest, then the English texts of WebNLG and finally the English texts of CACAPO (2). This is also

shown in table 3.4, where each column sums up to 100%, thus representing each split in their corresponding complexity level.

	CACAPO (Dutch)	CACAPO (English)	WebNLG (English)
0	49.3%	37.31%	49.27%
1	4.38%	2.57%	0.11%
2	28.13%	10.44%	20.24%
3	6.45%	9.14%	9.62%
4	0.4%	2.12%	0.22%
5	5.89%	9.22%	4.66%
6	3.21%	1.13%	3.94%
7	2.24%	28.08%	11.93%

Table 3.4: Content complexity comparison between CACAPO and WebNLG (2)

With the previously mentioned advantages that CACAPO offers, it was chosen as the preferred dataset to use for the experiments described in this paper. CACAPO was retrieved from Dataverse (66), where the data was downloaded in XML format, of which an example entry is shown in figure 3.1.

```
<entry category="EnglishSports" eid="Id8" size="1">
    <originaltripleset>
        <otriple>locationPlayed | St._Louis</otriple>
    </originaltripleset>
    <modifiedtripleset>
        <mtriple>locationPlayed | St._Louis</mtriple>
    </modifiedtripleset>
    <lex comment="good" lid="Id1">
        <sortedtripleset>
            <sentence ID="1">
                <striple>locationPlayed | St._Louis</striple>
            </sentence>
        </sortedtripleset>
        <references>
            <reference entity="St._Louis" number="1" tag="PATIENT-1" type="description">St. Louis</reference>
        </references>
        <text>After two taut games in St. Louis, this suddenly turned into a messy slugfest.</text>
        <template>After two taut games in PATIENT-1 , this suddenly turned into a messy slugfest .</template>
        <lexicalization>After two taut games in PATIENT-1 , DT[form=demonstrative] this suddenly VP[aspect=simple,
        tense=past, voice=active,person=null,number=null] turn into DT[form=undefined] a messy slugfest .</lexicalization>
    </lex>
    <entitymap>
        <entity>PATIENT-1 | St._Louis</entity>
    </entitymap>
</entry>
```

Figure 3.1: Entry example in CACAPO dataset (2)

In the preparation of this data, the appropriate information needed to be extracted into a usable format. As the dataset was modelled after WebNLG, it was possible to use and adapt the code used in these challenges. The WebNLG challenge provides an XML

Converter, named $WebNLG\_xmlReader$. Furthermore, as this research focused on end-to-end applications, only select pieces of each CACAPO entry were required for training. The XML reader was used to extract original triples and corresponding articles, after which these were saved into a CSV document. Following the data storage, the data was examined to confirm the creators' claims that the dataset does not contain any noise. This analysis showed this claim to be true and all values from the attribute value pairs in the input are represented in the output. For a quick overview of subject distributions, refer to 3.2.

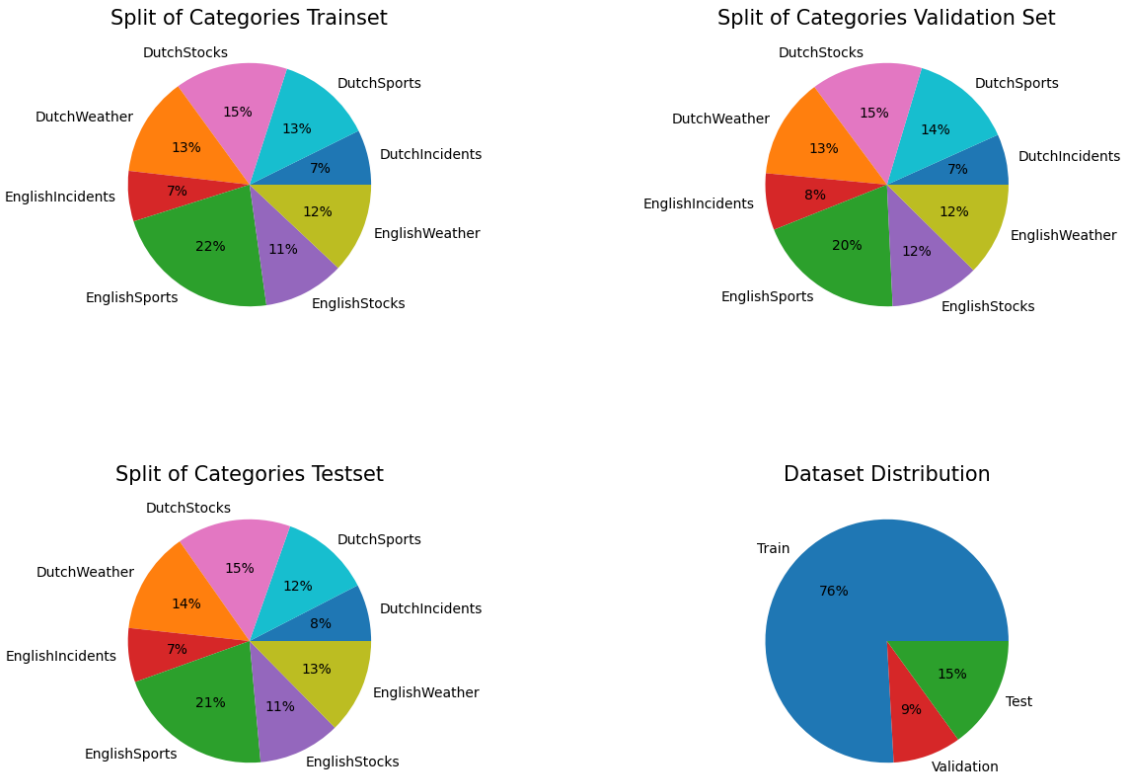

Figure 3.2: Train, validation, and test distributions of CACAPO

3.3 Model Training

Having chosen the models (T5-base, mT5-base, Bart-base) and dataset (CACAPO), the model can be fine-tuned. This was done with Hugging Face's AutoModelForSeq2SeqLM and AutoTokenizer, which selected the appropriate model and FastTokenizer as the appropriate default tokenizer for said models. Following this, the data was preprocessed into tensors. For this, the maximum token length for both input and output was set to 256 tokens, as no sentence would surpass 256 tokens. Moreover, the input data was prepared with ast's $literal\_eval$ before being tokenized, as loading through CSV a list of inputs to

one continuous string. Using *literal_eval* returns the input to its original form. Following preparation, the training arguments were set such that the model would be evaluated per epoch with a learning rate of $5e-5$ and using the optimizer *adafactor*, following Yeb Havinga's experimentation (67). Moreover, gradient accumulation was set to 2 steps, gradient accumulation was set to True, and the batch size was set to 16 to optimize for memory usage. The model was tasked to generate using Beam Search and Nucleus Sampling, where the total number of beams was set to 5 and the generation length set to 100. Moreover, the sampling space of *top_p* was set to 0.7 and a repetition penalty was set to 1.3. Finally, the model was given an early stopping criteria of 3, evaluated on the validation loss during training. These parameters are shown in table 3.5.

Parameter	Value
Model	AutoModelForSeq2SeqLM
Tokenizer	AutoTokenizer
Max padding length	256
Learning Rate	$5e-5$
Optimizer	Adafactor
Gradient Accumulation Steps	2
Gradient Checkpointing	True
Batch Size	16
Early Stopping	3
Decoding Strategy	Beam Search and Nucleus Sampling
Beam Count	5
Maximum token generation length	100
Top p	0.7
Repetition Penalty	1.3

Table 3.5: An overview of training parameters

Before training, a clean model was loaded in to ensure no accidental continuous learning occurs. Furthermore, each epoch was evaluated using popular metrics, n-gram methods BLUE, ROUGE, METEOR, and model-based methods such as BertScore and BartScore. After completion of the model's training, the model was tested on an unseen test set. The best model was loaded in and tasked to generate texts based on test set inputs. These generations were then decoded and evaluated by earlier mentioned metrics.

3.4 Evaluation Methods

Following model training and automatic evaluation, it was evident that the models were not perfect and making numerous errors. This called for additional analysis or model diagnostics. The results of each model's test run were therefore gathered and sorted based on the corresponding BARTScore, following Yuan et al.'s (68) observation that a model-based metric outperforms n-gram evaluation models on several perspectives, such as fluency, coherency, and factuality. After ordering these values, the worst 200 generations were gathered for analysis, as logically these records would contain the most amount of errors.

Following this, several evaluation buckets were created, taking inspiration from Thomson and Reiters' paper *"A Gold Standard Methodology for Evaluating Accuracy in Data-To-Text Systems"* (69), those being *No Generations*, *Incorrect Named Entity*, *Incorrect Word*. The manual analysis can be found in this project's Github repository[1].

After analysing the records, several patterns emerged. First, generations often contained both completely hallucinated content and omitted input data. This was highlighted by input records having very few attribute value pairings, whilst the reference text contained more contextual information that could have been captured in the input data, which would pose a challenge for any generation method, such as rule-based generation. An example is shown in Table 3.6.

Input	Generation
'locationPlayed | Baltimore'	The game was closed to fans out of concern for their safety following recent rioting in Baltimore after Freddie Gray, a 25-year-old black man, died in police custody.

Table 3.6: Example of Underspecified Input Data

Additionally, the generation contained different attributes than those mentioned in the input data and it even occasionally seemed that the model ignored certain attributes.

A significant portion of the recorded errors came from the 'Dutch Sports' subject. Analysis showed that the mistaken attributes were often attributes that occurred less often in the dataset and that these attributes were switched with attributes that occur more often. An example of such a switch is shown in Table 3.7 below:

[1] `https://github.com/SimonvdFliert/MscThesis`

Input	Generation
'redCardName | Linssen'	Linssen scoorde voor het eerst in de Eredivisie.

Table 3.7: Example of Mistaken Attributes

This observation sparked the question of whether augmenting the dataset such that attributes always occur at least a few hundred times could resolve this issue.

CHAPTER 4

EXPERIMENTS AND RESULTS

4.1 Manual Analysis

Taking the transformer models into account, two possible places were identified that could be the cause for the previously mentioned errors, that being the model and at its core the attention mechanism, or the dataset. The previously made observations sparked hypotheses regarding the dataset, however, it is useful to test proposed changes in a small setting, so that the resources at hand are not wasted. Hence pilot experiments were devised to test several possible changes. Additional Learning Interpretability Tools, such as Ecco (70) and BertViz (71), of which an example is shown in figure 4.1 below, were used as supporting visualisations. Note that all following experimentations were conducted on the mT5-base model.

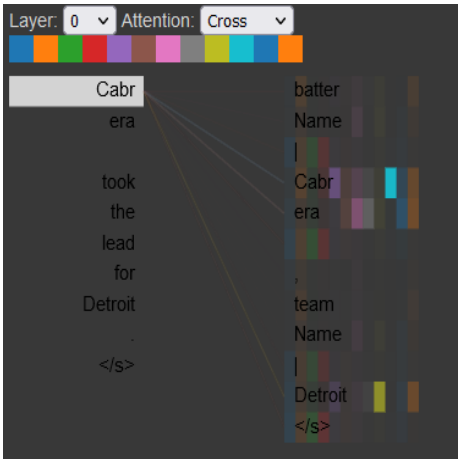

Figure 4.1: Example of BertViz model view

The first experiment focused on the initial observation where input data was missing contextual data that was present in the reference text. To test how models would react to additional data, records were manually reviewed to decide whether the reference texts contained contextually relevant information that was not present in the input data. Several instances were then selected for this analysis. Each additional attribute value pair was created with subject-specific attributes known in the dataset, of which an example is shown in Table 4.1 below.

Input Type	Generation
Original Input Data	'victimNumber | five'
Manually Elongated Input Data	'victimNumber | five', 'victimNumber | three', 'victimGender | women', 'victimGender | men'
Reference Text	The victims include five women and three men.

Table 4.1: Example of Subject Specific Attribute Addition

These additions resulted in a decrease in hallucinations and generations matching the target text better than the original generation. Furthermore, it was observed that many target texts had contextual information that was either difficult or impossible to connect to an attribute, thereby still leaving gaps in context. BertViz analysis showed that the distribution of attention scores was similar in few and many input RDF records. However, it was observed that records with hallucinated content had the hallucinated content focused on attributes or separators, whilst non-hallucinated content often focused on the values. This indicates that hallucinated content does come from the attributes and that the model does not derive any hallucinations from seen values.

Following these results, a similar question arose regarding contextual input data but now focused on removing attributes from well-performing generations and analysing how the model reacts. It was observed that removing a bit of context could quickly result in the model generating hallucinated content, thereby drastically changing the usability of the generation. The addition of hallucinated content was not consistent across all subjects and instances, but when it occurred, it often had a significant effect on the conclusion of the text. Moreover, it was observed that some subjects still generated the same texts, even with removed input rdfs. After analysing the dataset representations, it was found that this was due to the lack of diversity for these cases. For example, if the model sees "zonnige", it will also always output "perioden", as this is always occurring together, 38 out of 38 times. BertViz analysis showed similar results as experiment one, where hallucinated content is focused on the attribute and separator tokens, as shown in Figure 4.2.

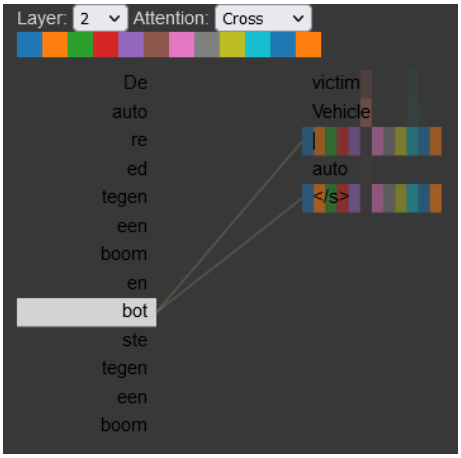

Figure 4.2: BertViz example of hallucinated content quickly focusing on the separator and attribute present in the input.

After this, another experiment was conducted similar to the first, where additional attribute value pairs were added. However, where the first experiment focused on attributes known within the subject, this experiment focused on inter-subject attributes, as shown in Table 4.2. For example, if a Sports record was manually elongated, the new attribute might have been from the Incidents subject. The evaluation showed that these additions worked well, given that the attribute is shown often enough within the dataset. For less occurring attributes, the model still had issues generating the target text. Furthermore, the model had trouble ordering the values, thus making mistakes and connecting attributes that should not be related.

Input Type	Generation
Original Input Data	temperatureCelsius | above average, timePoint | this summer
Manually Elongated Input Data	temperatureCelsius | above average, timePoint | this summer , ORG | the department, weatherArea | the province
Reference Text	The department said the temperature will remain above average this summer, according to the department.

Table 4.2: Example of Inter-Subject Attribute Addition

In the fourth experiment, new attribute-value pairs were added to those lacking input records, however, the attributes were *unseen* in the CACAPO dataset, as shown in Table 4.3 below. The new attributes resulted in generations containing numerous mistakes, where the model had no previous recollection of the attributes and had trouble placing them in

the right context.

Input Type	Generation
Original Input Data	LOC | Qatar, timePoint | public holiday
Manually Elongated Input Data	LOC | Qatar, event | closed, timePoint | public holiday
Reference Text	Qatar, where the public holiday closed, said a statement.

Table 4.3: Example of Unknown Attribute Addition

Finally, an experiment was conducted whereby records have their input data reordered, which showed that a different order of input data has no effect on the generation. An example is shown in Table 4.4 below.

Input Type	Generation
Original Input Data	temperatureCelsius | above average, timePoint | this summer , ORG | the department, weatherArea | the province
Switched Input Data	ORG | the department, weatherArea | the province, temperatureCelsius | above average, timePoint | this summer
First Generation	The department said the temperatures will be above average this summer, according to the department.
Second Generation	The department said the temperature will remain above average this summer, according to the department.

Table 4.4: Example of Input Order Experiment

These experiments highlighted that changes made to the dataset would need to be well-considered, as incorrect changes could lead to an increase in errors. From these experiments, it became clear that input data could easily be elongated with inter-subject attribute value pairings and that little attention needed to be placed on the order of these pairings. This laid the groundwork for the next experiments.

4.2 Augmented Data Experiment

Returning to the manual review, observations were made that attributes were often changed to different attributes or even left out altogether. Analysis showed that several of these changed attributes were rare within the dataset. This sparked the hypothesis that the infrequency of these attributes caused the model to gravitate to more frequent attributes,

as these attributes are similar in semantics, thus the model will naturally prefer to choose a token it has seen often enough. To test this hypothesis an experiment was devised to ensure that each attribute within the dataset would have the opportunity to be learned. One way to do this is by decreasing the relative difference of occurrences between frequent and infrequent attributes, however, this would require either the augmentation of very few records, which would decrease variety and increase the chance of duplications, or it would require to create entirely new records, which would be time-consuming and come with new considerations a creator would have to decide over. Another method is by focusing on the absolute occurrence of attributes, which was more applicable to the CACAPO dataset.

The topic of data augmentation has been considered widely in existing literature, not only in the domain of natural language generation. Due to the advancements in deep learning and model performance, other languages have been getting attention from researchers, however, many languages often lacked the required data to train a model on. This has sparked several variations on how to properly augment a dataset for training purposes, such as Ribeiro et al.'s *checklist* (72). This method was taken as inspiration for the augmentation script in this project.

The first step was to extract the current attribute count within the dataset and identify attributes that occur less often than 250 times. The cap of 250 attributes was set to avoid the dataset from quickly exploding to far greater sizes. Then, for each stored attribute, the dataset would be filtered on records containing said attribute and the corresponding values of these attributes were extracted and stored. Following this, the number of augmented records would be calculated to hit the cap of 250 attributes. Then for each filtered record, an augmented variant would be created where the value of the desired attribute would be changed with one of the earlier stored values. If all values were used for this record, then a second record would be taken, such that no duplicated values would occur for a specific record. Once the cap of 250 records was reached, the next attribute would be chosen and augmented.

After creating the augmented dataset, a new mT5-base model was created similarly to the baseline, but this time with the augmented training set instead of the baseline training set. All other factors remained consistent. After training, the augmented variant was compared to the baseline variant. Focusing on the automatic metrics, the augmentation did not improve the performance of the model, in fact, it performed worse compared to the baseline, as more records received lower BARTScores. Moreover, the model's performance was manually compared with the baseline by looking at the worst 200 generations from the manual review. The augmented version showed an increase in hallucinations and duplications, whilst omitting fewer data from the input, as shown in table 4.5.

Models	Baseline (% of total)	Augmented (% of total)
Hallucinations	141 (70.5%)	163 (81.5%)
Ommissions	21 (10.5%)	10 (5%)
Duplications	30 (15%)	35 (17.5%)
No Generation	14 (7%)	7 (3.5%)
Incorrect Number	1 (0.5%)	0
Incorrect Named Entity	21 (10.5%)	5 (2.5%)
Incorrect Word	31 (15.5%)	33 (16.5%)
Incorrect Addition	112 (56%)	126 (63%)
Other	2 (1%)	7 (3.5%)

Table 4.5: Comparison of models on the 200 worst performing generations from the original test set (N = 200). Lower values are best.

However, whilst this makes it seem that the augmentation results in no improvements based on BARTScore and minor improvements in the manual review, the augmentation was done to lower the chances of the model switching out attributes with different, more frequent attributes, and the model has been able to do so. Take the example shown in Table 4.6 below, where both the generation of the baseline and the generation of the augmented model is shown.

Input	Generation
tackleRecipientName | Kamohelo Mokotjo	Kamohelo Mokotjo scoorde.
tackleRecipientName | Kamohelo Mokotjo	De bal was op de stip beland nadat hij Kamohelo Mokotjo neerhaalde.

Table 4.6: Example of improvement in swapped attributes, likely due to the Augmentation

However, the augmentation has not completely eradicated the problem, where even the augmented model still mistakes attributes for different attributes, as shown in Table 4.7 below.

Input	Generation
chanceForName | Christian Santos	Christian Santos scoorde met een enorme kans.
chanceForName | Christian Santos	De ploeg van Christian Santos kreeg een enorme kans.

Table 4.7: Example of a mistaken attribute, even after augmentation

Another observation is the increase in hallucinations, mainly in the bucket of *Incorrect Additions*. This highlighted that when the augmented model hallucinates, it hallucinates severely, where half of the generation is often hallucinated. An example of a particularly egregious hallucination is shown in Table 4.8 below:

Input	Generation
shootingType | shooting , incidentType | asked to contact , ORG | Violent Crimes Tip Line at (912) 525-3124	Additionally, officers responded to a report of violations of the auto laws, firearm related offenses, and the shooting may be gang-related and retaliatory based on previous acts of violence.

Table 4.8: Example of Severe Hallucinations

4.3 Elongated Data Experiment

Another observation made in the manual review was that input data often lacked contextual information, even though this information is represented in the reference texts, for example shown in table 4.1. This would give the model the sheer impossible task of generating a text close to the reference with too few attribute value pairings, which often resulted in the model either not generating anything and only returning the values in the input, a *No Generation*, or that the model widely hallucinated. This led to the hypothesis that many hallucinations were caused by the lack of contextually relevant input data.

To test this hypothesis, it was decided to manually review the dataset and elongate the input data of records with relevant attribute value pairings, thereby taking earlier analysis into consideration, where all additional attributes are known within the dataset, not limited to the subject itself, and the order these attribute-value pairs are placed in did not matter.

With this in mind, the original training set was manually reviewed up to $10,000$ records and $1,504$ records were deemed to have underspecified input data compared to their corresponding reference text. These records were highlighted and manually elongated. An

example of such elongation is shown in Table 4.9 below, where additional pairings are highlighted in red.

Input Type	Generation
Original Input Data	'shootingType | shooting', 'suspectAge | 21-year-old', 'suspectName | Alvin_Bell'
Manually Elongated Input Data	'shootingType | shooting', 'suspectAge | 21-year-old', 'suspectName | Alvin_Bell', 'suspectStatus | arrested', 'datetime | Thursday'
Reference Text	Police said Thursday they arrested 21-year-old Alvin Bell, identified by mul- tiple witnesses as the gunman in the shooting, without incident.

Table 4.9: Example of Elongated Input Data

After the elongation of the training set, a new, clean, mT5-base model was fine-tuned on this altered training set, whilst keeping the validation-, test set, and hyperparameters the same as both the baseline and augmented models. After training, the model was compared to both the baseline and the augmented model, where the elongated model outperformed all others based on their BARTScores, as visible in Figure 4.3 below.

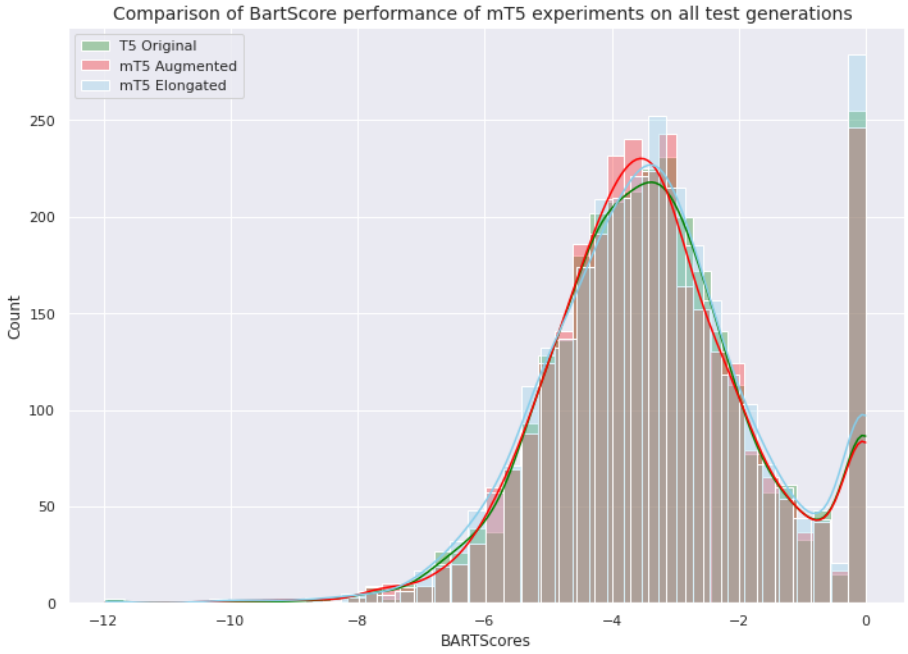

Figure 4.3: Overview of BartScore performances for each experiment based on the original test set

Furthermore, the model was compared with the other models based on the records of the manual review, and whilst the elongated model did not significantly improve over the baseline, the model still showed lower errors despite being confronted with records containing lacking input data. This entails that even only elongating records in the training set could improve the performance of the model during inference time. The results of this manual review are shown in Table 4.10 below.

Models	Baseline (% of total)	Augmented (% of total)	Elongated (% of total)
Hallucinations	141 (70.5%)	163 (81.5%)	135 (67.5%)
Ommissions	21 (10.5%)	10 (5%)	15 (7.5%)
Duplications	30 (15%)	35 (17.5%)	24 (12%)
No Generation	14 (7%)	7 (3.5%)	14 (7%)
Incorrect Number	1 (0.5%)	0	0
Incorrect Named Entity	21 (10.5%)	5 (2.5%)	5 (2.5%)
Incorrect Word	31 (15.5%)	33 (16.5%)	26 (13%)
Incorrect Addition	112 (56%)	126 (63%)	104 (52%)
Other	2 (1%)	7 (3.5%)	10 (5%)

Table 4.10: Comparison of models on the 200 worst-performing generations from the original test set (N = 200). Lower values are best.

4.3.1 Bilingualism

Moreover, as CACAPO contains both Dutch and English records, the question quickly arose if these experiments have differing effects dependent upon language. Thus, the original test set was split into both Dutch and English records and their corresponding BARTScores are plotted separately, one graph per experiment per language, as shown in Figure 4.4. Focusing on the top row, it is visible that the model trained on elongated data (far right) slightly outperforms the baseline, and that the model trained on augmented data (middle) has more generations with lower BARTScores. This is in line with previous mentions. However, compared to the Dutch generations, shown in the bottom row, the improvement of BARTScores for the elongated model is substantially more noticeable, where more generations have a near-perfect BARTScore and fewer generations have a BARTScore around -6.

Possible reasons for the difference between Dutch and English records could be that the pre-training data contains more English records compared to Dutch records, thus the models see more English during pre-training and fine-tuning. It is more likely that the model has seen more English records compared to Dutch records, as the internet is predominantly English and the dataset used for pre-training is the mC4 dataset, a multilingual dataset of cleaned, web-scraped data. For example, as explained in section 3.1.2, the Dutch C4 contains 151 GB of data, compared to the 750 GB of English data from the same data set.

Another possible reason could be the complexity of sentences. As table 3.4 shows, the majority of Dutch records are relatively simple, whilst there are significantly more complex English records, 28 per cent compared to the 2 per cent Dutch complex sentences. This is also likely the reason for the difference in model performance shown in Appendix E, where Dutch records slightly outperform English records.

These results might indicate that properly managed input data could allow users to use smaller datasets, as the higher quality of input data allows the model to learn as much as possible from underspecified data. Moreover, these results might also indicate that an improved specification of input data could be more impactful for relatively simple texts, highlighted by the increase in performance for Dutch records, but lacking increase of BARTScore performance for the English records. This might be caused by the higher amount of simple sentences for the Dutch sentences, ranging from level 0 to 2.

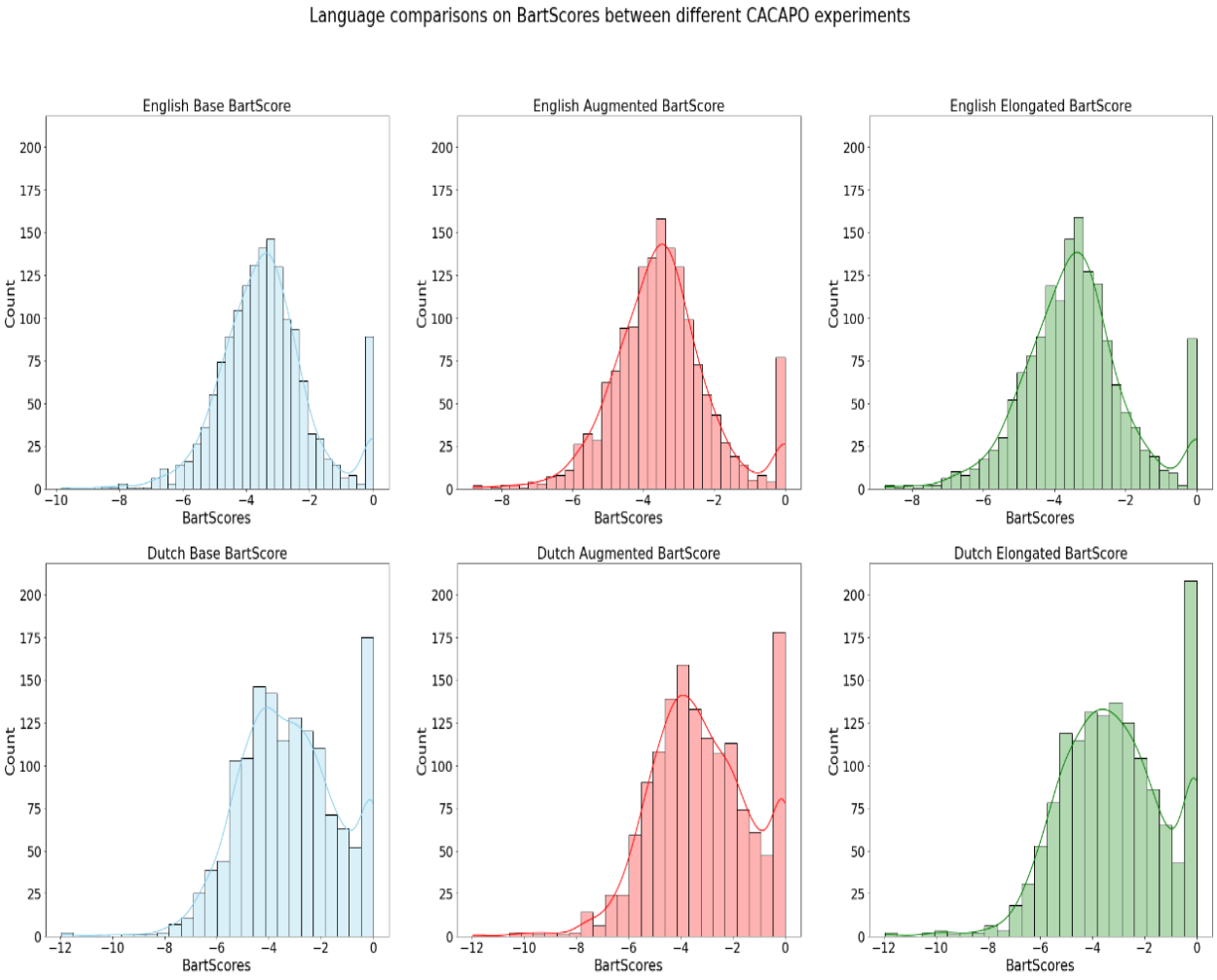

Figure 4.4: Overview of BartScore performances for each experiment on the original test set of CACAPO

4.4 PARENT Experiment

Even though BARTScores and manual evaluation give us an indication of how the model reacts to additional input data, it would be good to quantify how much the input data is taken into consideration for the models. An automatic metric that offers this is PARENT, which considers both the reference and the input data when calculating the Precision, Recall and F-1 score of a model's generations. Whilst it is created for table-to-text datasets, the metric is viable for RDF triples and usable for this project. The code for the PARENT metric was taken from this GitHub (73).

The PARENT metric first collects the number of n-grams in the generations and references and compares both the generations and references to the input values. Overlapping n-grams found between generation and input or reference and input are counted and stored, after which they are used to calculate the Precision, Recall, and F-1 score.

First, precision is calculated by taking each n-gram in the generation and comparing its occurrence in the reference, called the probability of occurring in the reference, shown in 4.1a. However, occasionally the values are not present in the reference but could be entailed from the input. Then, the probability of the n-gram not being in the reference is multiplied by the possibility of the value being entailed from the table, as shown in 4.1b.

$$Pr(g \in R_n^i), \tag{4.1a}$$

$$Pr(g \notin R_n^i)w(g) \tag{4.1b}$$

Where g refers to the n-gram in question, R_n^i refers to the reference text, and w refers to the probability the n-gram could be entailed from the input.

These two values are added, after which it is divided by the total amount of n-grams, thus giving the precision for that order of n-gram, as shown in formula 4.2, where $g \in G_n^i$ refers to the n-gram in the generation. This calculation is done for several orders, after which the geometric average is taken to get one final value.

$$E_p^n = \frac{\sum_{g \in G_n^i}[Pr(g \in R_n^i) + Pr(g \notin R_n^i)w(g)]\#_{G_n^i}(g)}{\sum_{g \in G_n^i}\#_{G_n^i}(g)} \tag{4.2}$$

The recall metric is separated into two different values, that being the reference recall and the input recall. shown as $E_r(R^i)$ and $E_r(T^i)$ in formula 4.3 below. The reference recall is calculated similarly to the precision, but instead of looking at all n-grams in the generation and comparing it to the reference, the n-grams are taken from the reference and analysed for overlap with the generation. This probability of overlap is multiplied by the count of

n-gram occurrence in the reference and the value which the n-gram could be entailed from the input data. This final value, the numerator, is divided by the denominator, which is the count of n-gram occurrence multiplied by the entailment possibility.

In contrast, the input recall is calculated by first analysing the overlap between input values and the generation, summing the overlap and dividing by total input values. The geometric average of the recall values is taken once again to receive a single value. However, depending on the overlap between reference and input, it might be necessary to have the metric weigh the recall values differently. The intuition here is that if the overlap between reference and input data is high, it indicates that the reference already captures the majority of the information and it would be unnecessary to have these values influence the metric twice. This overlap is calculated and stored in the variable λ. As the CACAPO dataset was created with a reverse engineering method, where the input values were exactly extracted from the reference text, most records in the dataset would have a λ of 0, which would set the input recall to a value of 1, thus ignoring it.

$$E_r = E_r(R^i)^{(1-\lambda)} E_r(T^i)^{(\lambda)} \tag{4.3}$$

To compare the impact of additional data in underspecified records, it was decided to create a subset of manually selected data to elongate with additional data, as described in section 4.3. To test each trained model fairly, it was not possible to use the already elongated training records, as the elongated model would have seen these records during training and thus have an advantage. As no models were saved before having seen the test set, it was possible to create a subset of elongated data from the test set without giving a model an advantage. For each addition, it was ensured that the attributes were known, not subject-specific, and the values were contextual information from the target text. This resulted in a test set size of 380 ($n = 380$). Furthermore, for each elongated record in the test set, its original record was stored as well, to compare an improved version with the original records. The original set is called "poor" and the improved version is called "good". Then the best models from the baseline run, the augmented run, and the elongated run were tasked to generate texts using both the poor and the good test subsets. This resulted in six sets of generations, which were evaluated on the PARENT metric. Table 4.11 shows the results of this experiment.

Model	Base good	Base poor	Augmented good	Augmented poor	Elongated good	Elongated poor
Sum Prec	165.589	67.642	152.142	65.0455	178.393	74.932
Sum Recall	54.968	30.506	52.962	28.212	59.522	37.704
Sum F score	64.542	15.996	59.487	15.133	70.384	20.102
PARENT Precision	0.436	0.178	0.400	0.171	0.469	0.197
PARENT Recall	0.145	0.080	0.139	0.074	0.157	0.099
PARENT F score	0.170	0.042	0.157	0.040	0.185	0.053
Mean BERTScore	0.941	0.887	0.935	0.884	0.945	0.888
Mean BARTScore	-3.014	-4.621	-3.155	-4.666	-2.943	-4.739
BLEU	0.317	0.067	0.301	0.064	0.328	0.052
Meteor	0.644	0.280	0.621	0.278	0.647	0.257
Rouge1	0.701	0.354	0.667	0.343	0.720	0.344
Rouge2	0.487	0.165	0.450	0.151	0.511	0.158
RougeL	0.595	0.317	0.570	0.304	0.626	0.311
RougeLsum	0.595	0.317	0.568	0.304	0.626	0.311

Table 4.11: Results of PARENT Experiment (n=380) on the three different models. A good model has generated on the test subset with elongated input data, whilst the poor model has generated on the original records. A higher value is best.

The results show that the model trained on the elongated training set performed the best. However, the increase in performance compared to the base model was not as strong as expected. Furthermore, compared to other works, the PARENT results are low, especially the recall. For example, in the paper by Parikh et al.(45), the authors show the results of different experiments done with three different models, one Bert model and two Seq2Seq models. Here the authors show PARENT results ranging between 22.2 and 52.6. One important note is that the size of the models is not stated (45). Another reference is the paper of Kale and Rastogi, who study the pre-train and fine-tune strategy for data-to-text tasks. One experiment shows the PARENT results over all T5-sizes on the ToTTo dataset and shows PARENT metrics ranging from 55.9 to 57.8. (74) A third reference is a paper

by Nan et al., who introduce the open-source DART dataset. In their paper, they tested their, and other datasets, with a range of models, including the T5 model. The T5 was shown in 3 different sizes, those being small, base, and large, which return PARENT ranges of 0.56, 0.57, and 0.58 respectively (54). Another example is the paper by Wang et al.(75), who propose a Transformer-based generation framework to achieve faithfulness between the generation and the original input data. They test their framework on the Wikiperson dataset using four different models. They state a recall and precision performance of 48.83 and 62.86 respectively (75). These papers highlight that the PARENT results of the previous experiment are low regardless of the model. However, these models have all been trained on different datasets, and taking CACAPOs creation method described in section 3.2, could explain this discrepancy.

Moreover, another potential reason could be the changeable parameters offered by the metric. Ideally, the same method is used as previous works have used, however, several previous works have not been clear on which changes have been made to make the PARENT metric work. Thus, a second experiment was executed after several changes were made to the PARENT code to make it more suitable for the CACAPO data, thereby leaving the actual calculation of values untouched, thus keeping the PARENT metric intact. These changes include the removal of punctuation, excluding punctuations such as *a.m.*, changing True booleans to the values seen in the reference text, the addition of both Dutch and English NLTK stopwords, and finally the separation of multiple token values into a group of single values, for example, *[critically_ wounded]* to *[critically], [wounded].* The resulting improvements are shown in table 4.12

Model	Sum Prec	Sum Recall	Sum F score	PARENT Precision	PARENT Recall	PARENT F score
Base good	339.393	119.877	149.591	0.8931	0.3155	0.3937
Base poor	248.089	22.978	28.730	0.6529	0.0605	0.0756
Aug good	328.191	109.252	136.493	0.8637	0.2875	0.3592
Aug poor	245.358	20.963	26.199	0.6457	0.0552	0.0689
Elon good	347.412	125.288	154.617	0.9142	0.3297	0.4069
Elon poor	251.085	19.356	24.252	0.6608	0.0509	0.0638

Table 4.12: Results of the PARENT Experiment on the three different models using the altered PARENT variant. A good model has generated on the test subset with elongated input data, whilst the poor model has generated on the original records. A higher value is best.

4.4.1 Qualitative Evaluation

Besides considering the PARENT metric, the set of good input records gave another opportunity to compare the models for a manual review. A subset of 64 English and Dutch records was taken and reviewed for each model, of which the results are shown in table 4.13.

Model	Baseline (% of total)	Augmented (% of total)	Elongated (% of total)
Hallucinations	64 (49.6%)	80 (62%)	59 (45.7%)
Omissions	27 (20.9%)	27 (20.9%)	38 (29.5%)
Duplications	27 (20.9%)	29 (22.5%)	15 (11.6%)
No Generation	7 (5.4%)	1 (0.77%)	5 (3.9%)
Incorrect Number	0	6 (4.6%)	1 (0.77%)
Incorrect Named Entity	4 (3.1%)	3 (2.3%)	4 (3.1%)
Incorrect Word	40 (31%)	49 (38%)	37 (28.7%)
Incorrect Addition	15 (11.6%)	31 (24%)	8 (6.2%)
Other	18 (13.9%)	19 (14.7%)	19 (14.7%)

Table 4.13: Results of a manual review conducted on 64 English and 65 Dutch records (n=129). The records were taken from the improved test subset created in the PARENT experiment. A higher value is best.

Comparing these results with earlier manual reviews shows a similar pattern, where the model trained on an augmented training set hallucinates more often. An interesting side effect that is not immediately shown in the numerical values is the severity of these hallucinations. Compared to the baseline, the augmented hallucinations are more severe, where the hallucination could be made of eight or more tokens. An example of this is shown in Table 4.14 below.

Data Type	Generation
Input Data	victimName | Adrian Potts, ORG | Police, datetime| Saturday night
Generation	Police said Adrian Petts, a senior at Roosevelt Senior High School, was arrested Saturday night for violations of the auto laws, according to police spokeswoman Sonya M. Toler.
Reference Text	Police on Saturday night identified him as Adrian Potts.

Table 4.14: Example of a Severe Hallucination

Another observation is the increase of omissions for the model trained on improved input records, where it now omits more data than comparable models, whilst originally it omitted fewer data.

One possible reason for this increase in omissions could occur due to more records with more RDFs in the input, however, no evidence was found in the manual review to support this hypothesis.

A second reason could be that the omitted RDFs were seen as not important by the model and as such the model would omit them from the generation. If this were to be the case, an argument could be made that the omitted RDFs are not contextually relevant. Further research could train a model to generate input from reference texts to get a better understanding of how a model reviews contextually relevant information.

Another observation was made where the quality of generations differed drastically between models. The baseline and augmented model hallucinated more often and the fluency of its generations was often not great, however the elongated often generated a fluent sentence, even though it occasionally omitted input data.

However, even though the fluency was decent, once the generation was compared with the reference text, it often became apparent that the order of tokens was not correctly captured, which is likely due to the lack of relational information between attributes in the input data. This often led to the generation containing different conclusions than the reference text, for example in Table 4.15 below.

Data Type	Generation
Input Data	batterName | Dukes, batterName | Josh Willingham, incidentType | one-upped in that department
Generation	Dukes and Josh Willingham were one-upped in that department.
Reference Text	Teammate Josh Willingham one-upped Dukes in that department.

Table 4.15: Example of Incorrect Conclusions within Generations

CHAPTER 5

DISCUSSION AND LIMITATIONS

5.1 Insights

The executed experiments provide interesting insights for the field of NLG. First, as highlighted in section 3.2, the act of reverse engineering a dataset comes with difficult challenges, where attributes need to be created that are unique enough to be informative, whilst still being general enough to be connected to values in the reference text. The manual analysis in section 4.1 highlighted that CACAPO could benefit from more contextual data in the input, which could be enhanced with inter-subject attributes, which was later confirmed in the elongation experiment described in section 4.3. Reverse engineering thus does offer a good opportunity to fill the gap left by scraped and crowdsourced datasets, as described in section 2.6, however, it would be advised to have more general attributes than those used in CACAPO, as contextually relevant information, such as events or quotes, could not be captured with the attributes used in CACAPO. Generalizing certain attributes will also make it easier to apply them across domains, which could improve the frequency of these attributes in the dataset. This could reduce Incorrect Word Hallucinations as shown by the augmentation experiment in section 4.2.

Furthermore, the augmentation experiment highlights the potential for data augmentation, however simultaneously showed that this process needs to be done carefully, such that as many duplicated records can be avoided and as much variety can be introduced. The field of NLG would benefit from a standardized augmentation method that crosses multiple languages, such that datasets could easily be expanded with underrepresented languages and would simultaneously suffer less from duplicated records. The method used in this project is limited, as it is unavoidable to generate duplicate attribute value pairings, due to infrequent pairings often being in the same record as frequent pairings. As this method

only changed the value of the infrequent pairing, it resulted in many duplicated frequent pairings and near-duplicate reference texts. Moreover, this duplication of frequent pairings could be the reason for the observation made in section 4.4, where hallucinations made by the augmented model were often more severe than comparison models, as shown in the example in table 4.14.

Going further, during previous experimentation it was observed that the order of attribute value pairings often was mistaken, thus resulting in incorrect conclusions. This is likely caused due to the lacking relational information between pairings in the input. This indicates that CACAPO is less suitable for base end-to-end models and highlights the need for an improved content selection process or module within end-to-end systems, such that the required information can be properly captured. Moreover, an interesting observation was made in section 4.3, where the elongated model performed better on underspecified input data compared to the base transformer and augmented variant. This could indicate that purely training a model on better-specified input data results in improved performance during inference, regardless of the specificity of the input data at that time.

Moreover, analysis between Dutch and English records in section 4.3.1 showed that the Dutch records improved noticeably for the elongated model, whilst English records stayed rather consistent between model variants. This could be caused by the difference in training set sizes between languages, as the models have seen more English records during pre-training and fine-tuning compared to Dutch records. This could highlight a correlation between improved contextual input data and the necessary training set size, where smaller data set sizes might be usable if the input data had all contextually relevant information. Another possible explanation could be the relative difference in sentence complexity highlighted by table 3.4, where Dutch records are simpler compared to English records. This could indicate that an improved specification of input data could be more impactful for relatively simple texts, highlighted by the increase in performance for Dutch records, but lacking increase of BARTScore performance for the English records. Future works would need to prove these possibilities. Furthermore, a manual review was conducted on a subset in section 4.4, where 64 English and 65 Dutch records were analysed. This review showed that the types and amount of errors are similar between languages, which could indicate that the origin of errors is consistent between the languages for this dataset. Future works would need to prove that this is the case for other languages as well.

5.2 Limitations

Although the experiments in this project showed interesting results, no experiment is without its limitations, which will be discussed in more detail below.

5.2.1 Manual Analysis

The manual analysis of the original T5 and mT5 models' generations provides the backbone for the experimental choices made in this project, however, this analysis is limited. First, the manual review was conducted by one individual, thus increasing the risk of unknown biases or errors seeping into the review process. Furthermore, only the worst 200 records were evaluated from the total 3028 and these records are not evenly weighted between languages and subjects. This could result in the findings of this review not translating well to other domains. Moreover, the review focused on whether a mistake was made, not the severity of the mistake made. This distinction is not only lacking in this current project but also in the field of NLG as a whole.

Other limitations arise for the pilot experiments on the input data. First, the experiments were conducted with hand-picked records, which introduces the risk of randomly choosing records that the model generates well. Moreover, due to the hand-picking of these records, a limited sample size is used to test each scenario, again increasing the risk of missing difficult records for the model to generate.

5.2.2 Augmented Experiments

Another limitation is found in the augmentation script used in this project. Taking inspiration from Ribeiro et al.'s checklist method (72), a script was created to switch the value of corresponding attributes with other known values in the dataset. This had the effect of creating more records, however, these records were almost identical in nature, thus increasing repetitiveness and increasing the chance of generations containing similar phrases, reducing the variety. Moreover, whilst this method increased the absolute number of attributes and ensured each attribute is seen at least 250 times, more frequent attributes would likely be augmented as well, as infrequently occurring attributes would be in the same record as more frequent occurring attributes. The latter attributes would not be augmented, and thus each additional record would contain duplicated values for the attributes that are not changed. This could be the reason for additional hallucinations, where the likelihood of certain values being generated is increased as these are seen more frequently in the training set.

5.2.3 Elongated Experiments

The elongation experiment is limited in its creation, where records were manually selected for elongation. This introduces selection biases, where records might not have been selected in favour of other records. Moreover, the measure of required contextual data was not corroborated with a second annotator, which introduces the risk of differences between the original dataset annotators and current additions. This might create confusion during training time, as considerations during the creation of the dataset, such as the 5Ws and 1H questions, might not be taken in the elongation of input records. Moreover, the elongation of input records was done with inter-subject attributes, and these attributes might not be able to capture all necessary information. For example, articles often contain quotes, however, CACAPO lacks a proper attribute to capture quotes. This results in records containing more contextual information, but not all, which could lead to the model still generating incorrect conclusions. The more data is left out of the input, the farther the model is from its original task, that being template-filling the input data into a good generation.

5.2.4 PARENT Experiments

Several changes were made to the PARENT code to make CACAPO more applicable to the metric, which is described in more detail in section 4.4. A limitation of these changes is located in other works, where decisions made by other researchers are often not clearly captured. This makes it difficult to know which changes have been made, for example, whether researchers use or leave out stopword overlaps, which makes it complicated to compare the results of this project with previous works. However, each decision made in this project is discussed to counteract this issue for future researchers.

5.2.5 Bilingualism Analysis

The analysis of bilingualism conducted in section 4.4 also holds several limitations, similar to the elongation experiment, where potential record selection bias, a lack of corroboration with a second annotator, and the dichotomy of capturing as much contextual information using known attributes could impact the results. Moreover, these limitations could have a stronger effect, as the subset of records analysed is small.

5.3 Relevance to the field of AI

The goal of this project was to gain a better understanding of Transformers and to attempt to uncover the origin of generation errors. Current research places a lot of focus

on improving the performance of models based on automatic metrics, by tuning hyperparameters or introducing modules in line with historic model improvements. This results in research focusing on improving the "What", where incremental improvements are chased. However, this leaves a gap in understanding how state-of-the-art models currently work and why certain activities result in fewer errors. Whilst this project has not been able to give a definitive answer to the research goal, it does give insights into the origin of certain errors in the curation and use of data, whilst also offering insights into bilingualism, which remains a focused research topic as the field of AI develops towards the foreground. Moreover, several additional questions and future works are raised, which are described in more detail in section 6.

CHAPTER 6

CONCLUSION AND FUTURE WORKS

This research has sought to uncover the origin of generation errors in state-of-the-art transformer models. The T5, mT5, and BART models were trained on the CACAPO dataset (2) and experimentations regarding the quality and quantity of the dataset were executed, alongside manual reviews.

Manual reviews highlighted that the creation of a dataset by reverse engineering input data from the reference texts is a difficult task and that a significant amount of records in the CACAPO dataset lack appropriate contextual data for the required task of template-filling a generation close to the reference text.

Furthermore, CACAPO contains many unique attributes, whilst containing a relatively low number of records compared to other popular datasets. This results in several attributes occurring infrequently in the dataset and thus lowering the chances of the model learning these attributes appropriately. This dichotomy of unique attributes and capturing all contextual data gives dataset creators the difficult task of having attributes unique enough to be informative, whilst still plentiful enough to capture contextual information from varied human speech patterns. The pilot analysis showed the possibility of using inter-subject attributes to partially alleviate this issue.

Moreover, the experiments highlighted that end-to-end systems have difficulties correctly structuring generations. The CACAPO dataset for end-to-end models lacks relational information between attributes, which resulted in several ordering errors, thus leading to incorrect conclusions. This highlights the need for an improved content selection process or module within end-to-end systems so that the required information can be properly captured.

The elongation experiment highlighted the need for improved specification of input data, where the overal BARTScore performance during inference on underspecified input data

was better compared to the baseline and augmented variants. This could indicate that the addition of input specification could prove beneficial, even when input specification quality cannot be guaranteed during inference.

Furthermore, Dutch records' performance improved the most from the additional specifications, which could be caused by the relatively simple sentence complexity of these records compared to English records, which might show that improved specification has a greater impact on simple sentences compared to complex sentences. Another possible reason is the amount of Dutch records seen by the model during pre-training and fine-tuning, as this naturally would be lower than English records. This could highlight a correlation between improved contextual input data and the necessary training set size, where smaller data set sizes might be usable if the input data had all contextually relevant information.

Manual analysis of Dutch and English records highlighted that the amount and type of errors were consistent between languages, indicating that the origin does not differ between languages. However, the severity of these errors was not captured.

Finally, experimentation showed the usefulness of data augmentation, but simultaneously highlighted the need for a careful curation process, so that duplicated records can be avoided where possible, and as much variety as possible can be introduced.

Following this project, future works could focus their research in different directions. First, this project has highlighted that CACAPO contains many records that lack contextual input data. An attempt was made to improve this, but there is room to enhance the dataset further and thus develop a fully contextual dataset with reverse-engineered qualities. Another dataset-focused follow-up work could be to take the creation methods described by van der Lee et al. (2) and reverse engineer a new dataset. Whilst this will be more costly, it would prove beneficial for the field of NLG to detail the difficulties of this process, from the collection of data to the detailed choices that are required to be made when creating annotations.

Another direction is a focus on data augmentation. Previous works such as Riberio et al. (72) attempted to create an easy-to-use script to quickly augment English datasets. However, this method was shown to have only limited applicability to languages other than English, such as the Dutch language used in this project, but also proved limited in the types of augmentation that could be conducted. Future works could focus on enhancing the checklist method for the English language, whilst also making it more applicable to other languages. Especially the latter would be beneficial to the field, as several languages currently lack representation and are costly to gather.

Moreover, researchers could also focus on the standardization of manual reviews. This project took inspiration from Thomson and Reiter's paper (69) and whilst insightful, no

method was found to distinguish between the severity of mistakes. This distinction would be insightful for the field, where big reoccurring errors could take priority in research.

A fourth direction would be to delve deeper into the performance of the previously discussed models to gain a better understanding of why the results of the discussed metrics are as they are. For example, the PARENT recall metrics for the models used during this research were lower than in previous works, however, a definitive cause for this discrepancy was not identified in this project. Such understanding could benefit the field in the application and the understanding of these metrics when comparing different works with each other.

Moreover, future works could focus on the project observations regarding omissions, especially the observation where an increase in contextual data in the input led to an increase in omissions. Initial hypotheses were made where the length of the input could have impacted the chance of omissions, however, no evidence for this was found in this project. Nonetheless, whilst this was not observed in the limited sample sets, further research is required to definitively conclude whether or not the length of an input can affect the chance of omissions. Moreover, the witnessed increase in omissions raised the question of whether the omitted data was required to generate good generations, or whether it was omittable. Future works could highlight when input data is required contextually, which could support research in content selection.

Looking back to the research directions discussed in section 2.5, several previous works have adapted Transformers with additional modules. The results of this project indicate that this is a good direction to focus on. Observations such as the underspecification of input data, omission of context-relevant Named Entities, and incorrect generation of relations between entities indicate that end-to-end models struggle with content selection and content determination. Additional modules and models could be added to enhance this aspect. One possible method is to train a model to create an input from target texts, the reverse of what this project focused on. Once this is trained, inputs could be automatically created, after which a content selection module could tailor the created input to remove redundant information and then a content determination module could ensure that each required value is in the generation. Moreover, observations in this project have highlighted that the end-to-end model has difficulties capturing the correct relationship between entities, as this relation is not shown in the input. Thus the model would often generate an incorrect relationship between entities. Such relational data could be captured by the model that creates input from reference texts, thus removing this burden from end-to-end models. The modules would then return a seemingly optimal input to the transformer to generate a text with.

REFERENCES

[1] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. **Attention is all you need**. *Advances in neural information processing systems*, **30**, 2017. vi, 5, 7, 21, 22

[2] Chris van der Lee, Chris Emmery, Sander Wubben, and Emiel Krahmer. **The CACAPO dataset: A multilingual, multi-domain dataset for neural pipeline and end-to-end data-to-text generation**. In *Proceedings of the 13th International Conference on Natural Language Generation*, pages 68–79, 2020. vi, viii, 12, 13, 14, 15, 16, 23, 24, 25, 52, 53

[3] Khofiz Shakhidi. **Council post: Information is the new gold**. *Forbes*, Jun 2020. 1

[4] Albert Gatt and Emiel Krahmer. **Survey of the state of the art in natural language generation: Core tasks, applications and evaluation**. *Journal of Artificial Intelligence Research*, **61**:65–170, 2018. 1, 3, 4, 12, 18, 19

[5] Chenhe Dong, Yinghui Li, Haifan Gong, Miaoxin Chen, Junxin Li, Ying Shen, and Min Yang. **A Survey of Natural Language Generation**. *arXiv preprint arXiv:2112.11739*, 2021. 1, 3, 4, 12

[6] Robert Dale. **Natural language generation: The commercial state of the art in 2020**. *Natural Language Engineering*, **26**(4):481–487, 2020. 1

[7] Ratish Puduppully and Mirella Lapata. **Data-to-text generation with macro planning**. *Transactions of the Association for Computational Linguistics*, **9**:510–527, 2021. 2, 9

[8] Ratish Puduppully, Li Dong, and Mirella Lapata. **Data-to-text generation with content selection and planning**. In *Proceedings of the AAAI conference on artificial intelligence*, **33**, pages 6908–6915, 2019. 2, 8

[9] Li Gong, Josep M Crego, and Jean Senellart. **Enhanced transformer model for data-to-text generation**. In *Proceedings of the 3rd Workshop on Neural Generation and Translation*, pages 148–156, 2019. 2

[10] Ehud Reiter and Robert Dale. **Building applied natural language generation systems**. *Natural Language Engineering*, **3**(1):57–87, 1997. 3

[11] Somayajulu Sripada, Neil Burnett, Ross Turner, John Mastin, and Dave Evans. **A case study: NLG meeting weather industry demand for quality and quantity of textual weather forecasts**. In *Proceedings of the 8th International Natural Language Generation Conference (INLG)*, pages 1–5, 2014. 3

[12] Yi Yang, Mark Christopher Siy Uy, and Allen Huang. **Finbert: A pretrained language model for financial communications**. *arXiv preprint arXiv:2006.08097*, 2020. 3

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

[14] João Pinto Barbosa Machado Aires. **Automatic generation of sports news**. 2016. 3

[15] Shuang Chen, Jinpeng Wang, Xiaocheng Feng, Feng Jiang, Bing Qin, and Chin-Yew Lin. **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)*, pages 3022–3032, 2019. 4

[16] Erkut Erdem, Menekse Kuyu, Semih Yagcioglu, Anette Frank, Letitia Parcalabescu, Barbara Plank, Andrii Babii, Oleksii Turuta, Aykut Erdem, Iacer Calixto, et al. **Neural Natural Language Generation: A Survey on Multilinguality, Multimodality, Controllability and Learning**. *Journal of Artificial Intelligence Research*, **73**:1131–1207, 2022. 4

[17] Junyi Li, Tianyi Tang, Wayne Xin Zhao, Jian-Yun Nie, and Ji-Rong Wen. **A survey of pretrained language models based text generation**. *arXiv preprint arXiv:2201.05273*, 2022. 4

[18] Ruslan Yermakov, Nicholas Drago, and Angelo Ziletti. **Biomedical Data-to-Text Generation via Fine-Tuning Transformers**. *arXiv preprint arXiv:2109.01518*, 2021. 4, 19

[19] Katikapalli Subramanyam Kalyan, Ajit Rajasekharan, and Sivanesan Sangeetha. **Ammus: A survey of transformer-based pretrained models in natural language processing**. *arXiv preprint arXiv:2108.05542*, 2021. 4

[20] Zdeněk Kasner and Ondřej Dušek. **Data-to-text generation with iterative text editing**. *arXiv preprint arXiv:2011.01694*, 2020. 4, 19

[21] Abhishek Singh. **PoinT-5: Pointer Network and T-5 based Financial NarrativeSummarisation**. *arXiv preprint arXiv:2010.04191*, 2020. 4, 19

[22] Ramakanth Pasunuru, Mengwen Liu, Mohit Bansal, Sujith Ravi, and Markus Dreyer. **Efficiently summarizing text and graph encodings of multi-document clusters**. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4768–4779, 2021. 4, 19

[23] Hong Chen, Hiroya Takamura, and Hideki Nakayama. **SciXGen: A Scientific Paper Dataset for Context-Aware Text Generation**. *arXiv preprint arXiv:2110.10774*, 2021. 4, 19

[24] Chin-Yew Lin. **Rouge: A package for automatic evaluation of summaries**. In *Text summarization branches out*, pages 74–81, 2004. 5, 19

[25] Moniba Keymanesh, Adrian Benton, and Mark Dredze. **What Makes Data-to-Text Generation Hard for Pretrained Language Models?** *arXiv preprint arXiv:2205.11505*, 2022. 7

[26] Ethan Joseph, Julian Lioanag, and Mei Si. **Improving Data-to-Text Generation via Preserving High-Frequency Phrases and Fact-Checking**. *IJCoL. Italian Journal of Computational Linguistics*, **7**(7-1, 2):223–244, 2021. 7

[27] Thiago Castro Ferreira, Chris van der Lee, Emiel Van Miltenburg, and Emiel Krahmer. **Neural data-to-text generation: A comparison between pipeline and end-to-end architectures**. *arXiv preprint arXiv:1908.09022*, 2019. 8

[28] Claire Gardent, Anastasia Shimorina, Shashi Narayan, and Laura Perez-Beltrachini. **The WebNLG challenge: Generating text from RDF data**. In *Proceedings of the 10th International Conference on Natural Language Generation*, pages 124–133, 2017. 8

[29] Juliette Faille, Albert Gatt, and Claire Gardent. **The Natural Language Generation Pipeline, Neural Text Generation and Explainability**. In *2nd Workshop on Interactive Natural Language Technology for Explainable Artificial Intelligence*, 2020. 8

[30] Hamza Harkous, Isabel Groves, and Amir Saffari. **Have your text and use it too! end-to-end neural data-to-text generation with semantic fidelity**. *arXiv preprint arXiv:2004.06577*, 2020. 8, 18

[31] Zdeněk Kasner and Ondřej Dušek. **Neural Pipeline for Zero-Shot Data-to-Text Generation**. *arXiv preprint arXiv:2203.16279*, 2022. 9

[32] Yang Yang, Juan Cao, Yujun Wen, and Pengzhou Zhang. **Table to text generation with accurate content copying**. *Scientific reports*, **11**(1):1–12, 2021. 9

[33] Clément Rebuffel, Laure Soulier, Geoffrey Scoutheeten, and Patrick Gallinari. **A hierarchical model for data-to-text generation**. In *European Conference on Information Retrieval*, pages 65–80. Springer, 2020. 9

[34] Gian Wiher, Clara Meister, and Ryan Cotterell. **On decoding strategies for neural text generators**. *arXiv preprint arXiv:2203.15721*, 2022. 9, 10, 11

[35] Sina Zarriess, Henrik Voigt, and Simeon Schüz. **Decoding methods in neural language generation: a survey**. *Information*, **12**(9):355, 2021. 9, 10, 11

[36] Luca Massarelli, Fabio Petroni, Aleksandra Piktus, Myle Ott, Tim Rocktäschel, Vassilis Plachouras, Fabrizio Silvestri, and Sebastian Riedel. **How decoding strategies affect the verifiability of generated text**. *arXiv preprint arXiv:1911.03587*, 2019. 10, 11

[37] Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. **The curious case of neural text degeneration**. *arXiv preprint arXiv:1904.09751*, 2019. 10

[38] Sean Welleck, Ilia Kulikov, Stephen Roller, Emily Dinan, Kyunghyun Cho, and Jason Weston. **Neural text generation with unlikelihood training**. *arXiv preprint arXiv:1908.04319*, 2019. 11

[39] BOT_DEVELOPER. **Daily Financial News for 6000+ stocks**. *Kaggle*, Jul 2020. 12, 15

[40] GENNADIYR. **Historical Financial News Archive**. *Kaggle*, Feb 2020. 12, 16

[41] NEVIL DSOUZA. **Financial Markets dataset- prices & news**. *Kaggle*, May 2019. 12, 16

[42] ZHIYU CHEN, WENHU CHEN, CHARESE SMILEY, SAMEENA SHAH, IANA BOROVA, DYLAN LANGDON, REEMA MOUSSA, MATT BEANE, TING-HAO HUANG, BRYAN ROUTLEDGE, ET AL. **Finqa: A dataset of numerical reasoning over financial data**. *arXiv preprint arXiv:2109.00122*, 2021. 12, 16

[43] SAMIR ABDALJALIL AND HOUDA BOUAMOR. **An Exploration of Automatic Text Summarization of Financial Reports**. In *Proceedings of the Third Workshop on Financial Technology and Natural Language Processing*, pages 1–7, 2021. 12, 16

[44] MANUEL R VARGAS, BEATRIZ SLP DE LIMA, AND ALEXANDRE G EVSUKOFF. **Deep learning for stock market prediction from financial news articles**. In *2017 IEEE international conference on computational intelligence and virtual environments for measurement systems and applications (CIVEMSA)*, pages 60–65. IEEE, 2017. 12

[45] ANKUR P PARIKH, XUEZHI WANG, SEBASTIAN GEHRMANN, MANAAL FARUQUI, BHUWAN DHINGRA, DIYI YANG, AND DIPANJAN DAS. **ToTTo: A controlled table-to-text generation dataset**. *arXiv preprint arXiv:2004.14373*, 2020. 13, 17, 43

[46] THIAGO CASTRO FERREIRA, DIEGO MOUSSALLEM, EMIEL KRAHMER, AND SANDER WUBBEN. **Enriching the WebNLG corpus**. In *Proceedings of the 11th International Conference on Natural Language Generation*, pages 171–176, 2018. 13, 17

[47] CLAIRE GARDENT, ANASTASIA SHIMORINA, SHASHI NARAYAN, AND LAURA PEREZ-BELTRACHINI. **Creating training corpora for nlg micro-planning**. In *55th annual meeting of the Association for Computational Linguistics (ACL)*, 2017. 13

[48] SHANKAR KANTHARA, RIXIE TIFFANY KO LEONG, XIANG LIN, AHMED MASRY, MEGH THAKKAR, ENAMUL HOQUE, AND SHAFIQ JOTY. **Chart-to-Text: A Large-Scale Benchmark for Chart Summarization**. *arXiv preprint arXiv:2203.06486*, 2022. 13, 17

[49] Zhoujun Cheng, Haoyu Dong, Zhiruo Wang, Ran Jia, Jiaqi Guo, Yan Gao, Shi Han, Jian-Guang Lou, and Dongmei Zhang. **Hitab: A hierarchical table dataset for question answering and natural language generation**. *arXiv preprint arXiv:2108.06712*, 2021. 13, 17

[50] Shereen Oraby, Vrindavan Harrison, Abteen Ebrahimi, and Marilyn Walker. **Curate and generate: A corpus and method for joint control of semantics and style in neural nlg**. *arXiv preprint arXiv:1906.01334*, 2019. 13, 17

[51] Jekaterina Novikova, Ondřej Dušek, and Verena Rieser. **The E2E dataset: New challenges for end-to-end generation**. *arXiv preprint arXiv:1706.09254*, 2017. 13, 17

[52] Sam Wiseman, Stuart M Shieber, and Alexander M Rush. **Challenges in data-to-document generation**. *arXiv preprint arXiv:1707.08052*, 2017. 13, 16

[53] Ratish Puduppully, Li Dong, and Mirella Lapata. **Data-to-text generation with entity modeling**. *arXiv preprint arXiv:1906.03221*, 2019. 13, 16

[54] Linyong Nan, Dragomir Radev, Rui Zhang, Amrit Rau, Abhinand Sivaprasad, Chiachun Hsieh, Xiangru Tang, Aadit Vyas, Neha Verma, Pranav Krishna, et al. **Dart: Open-domain structured data record to text generation**. *arXiv preprint arXiv:2007.02871*, 2020. 14, 17, 44

[55] Sebastian Ruder, Ivan Vulić, and Anders Søgaard. **Square One Bias in NLP: Towards a Multi-Dimensional Exploration of the Research Manifold**. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 2340–2354, Dublin, Ireland, May 2022. Association for Computational Linguistics. 15

[56] Asli Celikyilmaz, Elizabeth Clark, and Jianfeng Gao. **Evaluation of text generation: A survey**. *arXiv preprint arXiv:2006.14799*, 2020. 18, 19

[57] Wei Li, Wenhao Wu, Moye Chen, Jiachen Liu, Xinyan Xiao, and Hua Wu. **Faithfulness in Natural Language Generation: A Systematic Survey of Analysis, Evaluation and Optimization Methods**. *arXiv preprint arXiv:2203.05227*, 2022. 18

[58] Tatsunori B Hashimoto, Hugh Zhang, and Percy Liang. **Unifying human and statistical evaluation for natural language generation**. *arXiv preprint arXiv:1904.02792*, 2019. 19

[59] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. **Bleu: a method for automatic evaluation of machine translation**. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318, 2002. 19

[60] Jekaterina Novikova, Ondřej Dušek, Amanda Cercas Curry, and Verena Rieser. **Why we need new evaluation metrics for NLG**. *arXiv preprint arXiv:1707.06875*, 2017. 19

[61] **Transformers**. *Hugging Face*. 20

[62] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. **Exploring the limits of transfer learning with a unified text-to-text transformer**. *The Journal of Machine Learning Research*, **21**(1):5485–5551, 2020. 21

[63] Yeb Havinga. **Yhavinga/T5-V1.1-base-dutch-cased · hugging face**. *Hugging Face*. 21

[64] Linting Xue, Noah Constant, Adam Roberts, Mihir Kale, Rami Al-Rfou, Aditya Siddhant, Aditya Barua, and Colin Raffel. **mT5: A Massively Multilingual Pre-trained Text-to-Text Transformer**. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 483–498, Online, June 2021. Association for Computational Linguistics. 22

[65] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. **Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension**. *arXiv preprint arXiv:1910.13461*, 2019. 22

[66] Chris van der Lee, Chris Emmery, Sander Wubben, and Emiel Krahmer. **CACAPO dataset**, 2022. 25

[67] Yeb Havinga. **Pre-training Dutch T5 and UL2 models, evaluation and model lists - a hugging face space by Yhavinga**. *Hugging Face Space*. 27

[68] Weizhe Yuan, Graham Neubig, and Pengfei Liu. **Bartscore: Evaluating generated text as text generation**. *Advances in Neural Information Processing Systems*, **34**:27263–27277, 2021. 28

[69] Craig Thomson and Ehud Reiter. **A gold standard methodology for evaluating accuracy in data-to-text systems**. *arXiv preprint arXiv:2011.03992*, 2020. 28, 53

[70] J Alammar. **Ecco: An Open Source Library for the Explainability of Transformer Language Models**. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing: System Demonstrations*, pages 249–257, Online, August 2021. Association for Computational Linguistics. 30

[71] Jesse Vig. **BertViz: A tool for visualizing multihead self-attention in the BERT model**. In *ICLR workshop: Debugging machine learning models*, 2019. 30

[72] Marco Tulio Ribeiro, Tongshuang Wu, Carlos Guestrin, and Sameer Singh. **Beyond accuracy: Behavioral testing of NLP models with CheckList**. *arXiv preprint arXiv:2005.04118*, 2020. 34, 49, 53

[73] Google-Research. **Language/language/table_text_eval at master · google-research/language**. *GitHub*. 41

[74] Mihir Kale and Abhinav Rastogi. **Text-to-text pre-training for data-to-text tasks**. *arXiv preprint arXiv:2005.10433*, 2020. 43

[75] Zhenyi Wang, Xiaoyang Wang, Bang An, Dong Yu, and Changyou Chen. **Towards faithful neural table-to-text generation with content-matching constraints**. *arXiv preprint arXiv:2005.00969*, 2020. 44

[76] Tian Lan. **Generating human-level text with contrastive search in Transformers** . *Hugging Face*. 72

[77] Yixuan Su and Nigel Collier. **Contrastive search is what you need for neural text generation**. *arXiv preprint arXiv:2210.14140*, 2022. 73

APPENDIX A

MODEL TRAINING GRAPHS

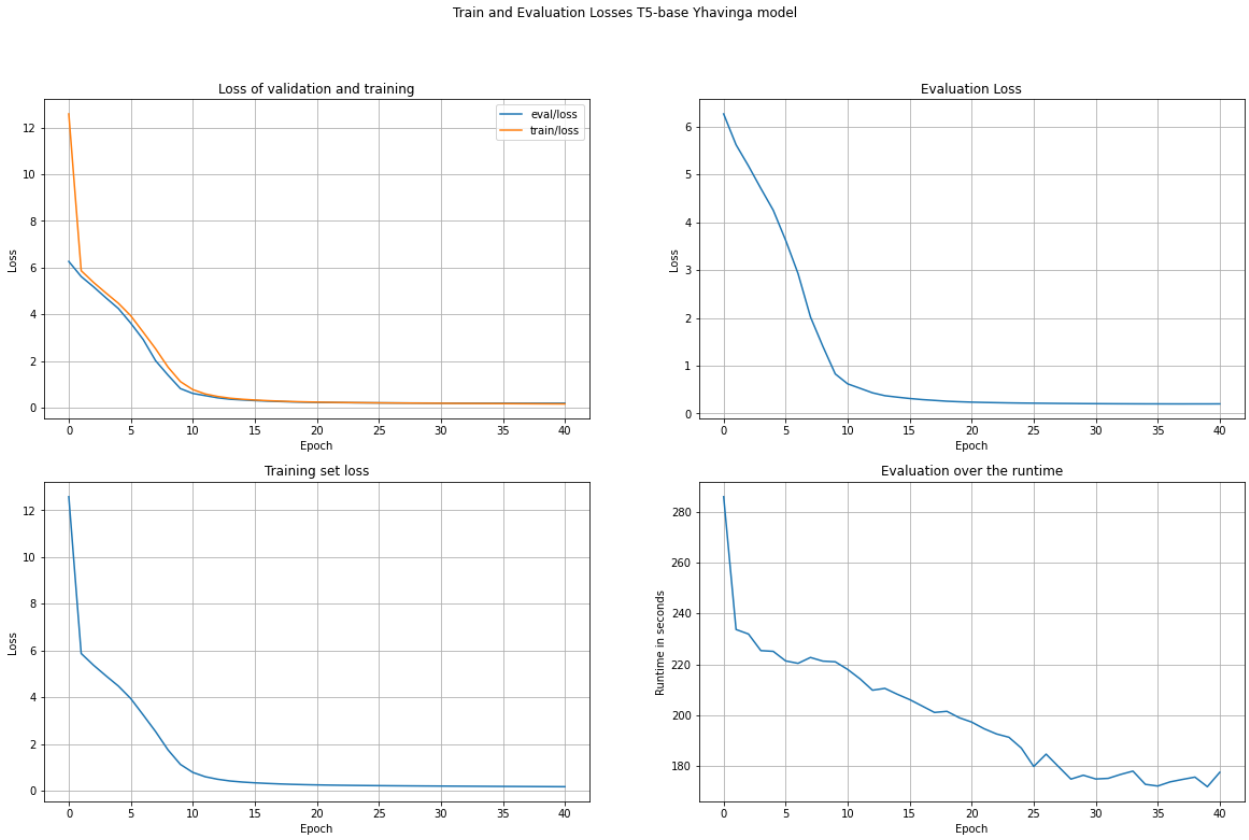

Figure A.1: Training overview of T5-dutch model of Yeb Havinga trained on the original CACAPO dataset

A. MODEL TRAINING GRAPHS

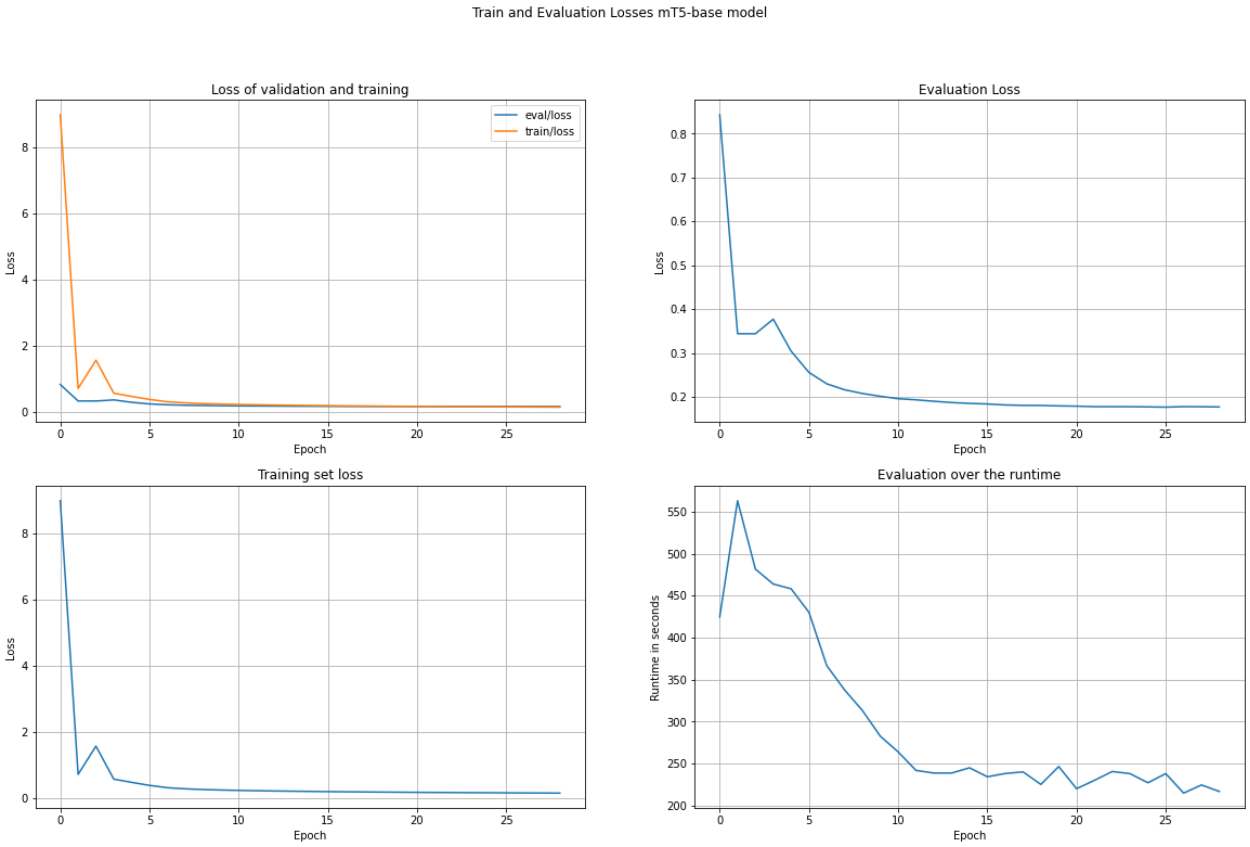

Figure A.2: Training overview of mT5-base model trained on the original CACAPO dataset

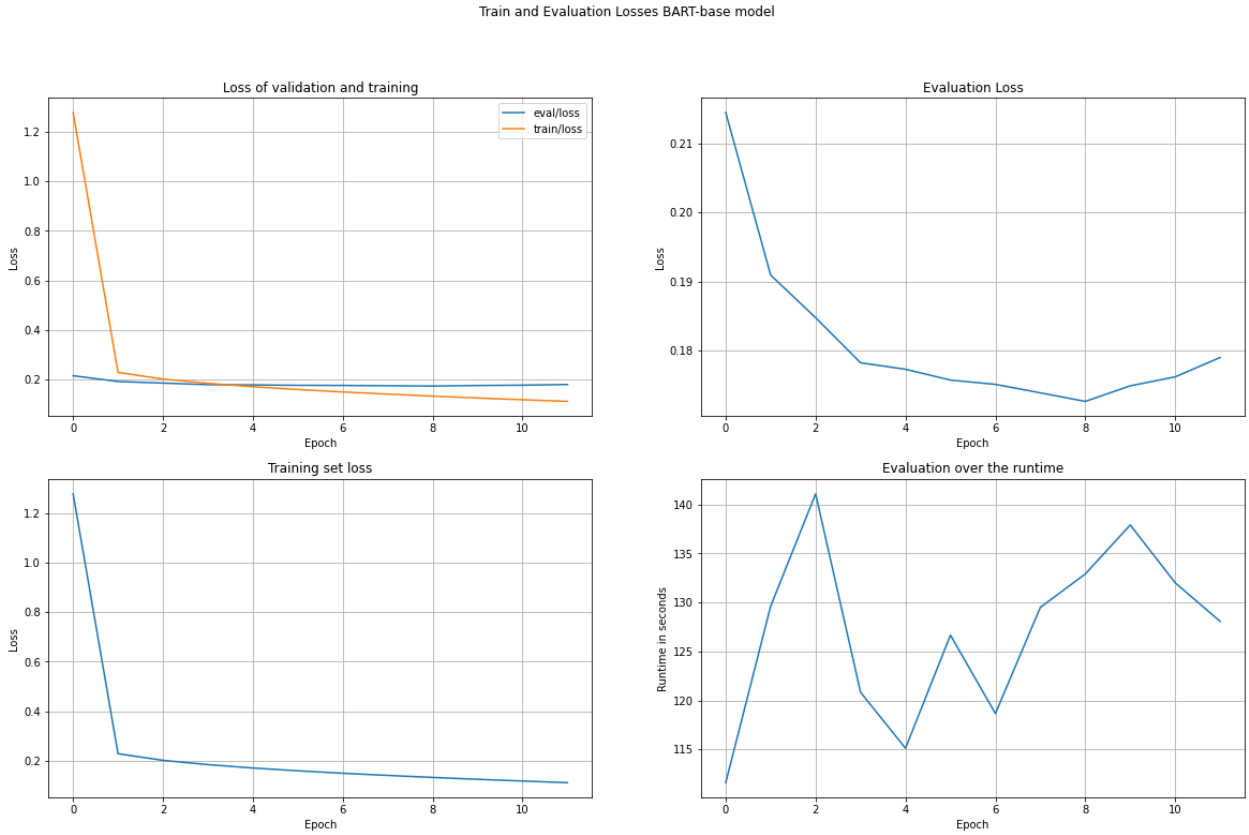

Figure A.3: Training overview of Bart-base model trained on the original CACAPO dataset

A. MODEL TRAINING GRAPHS

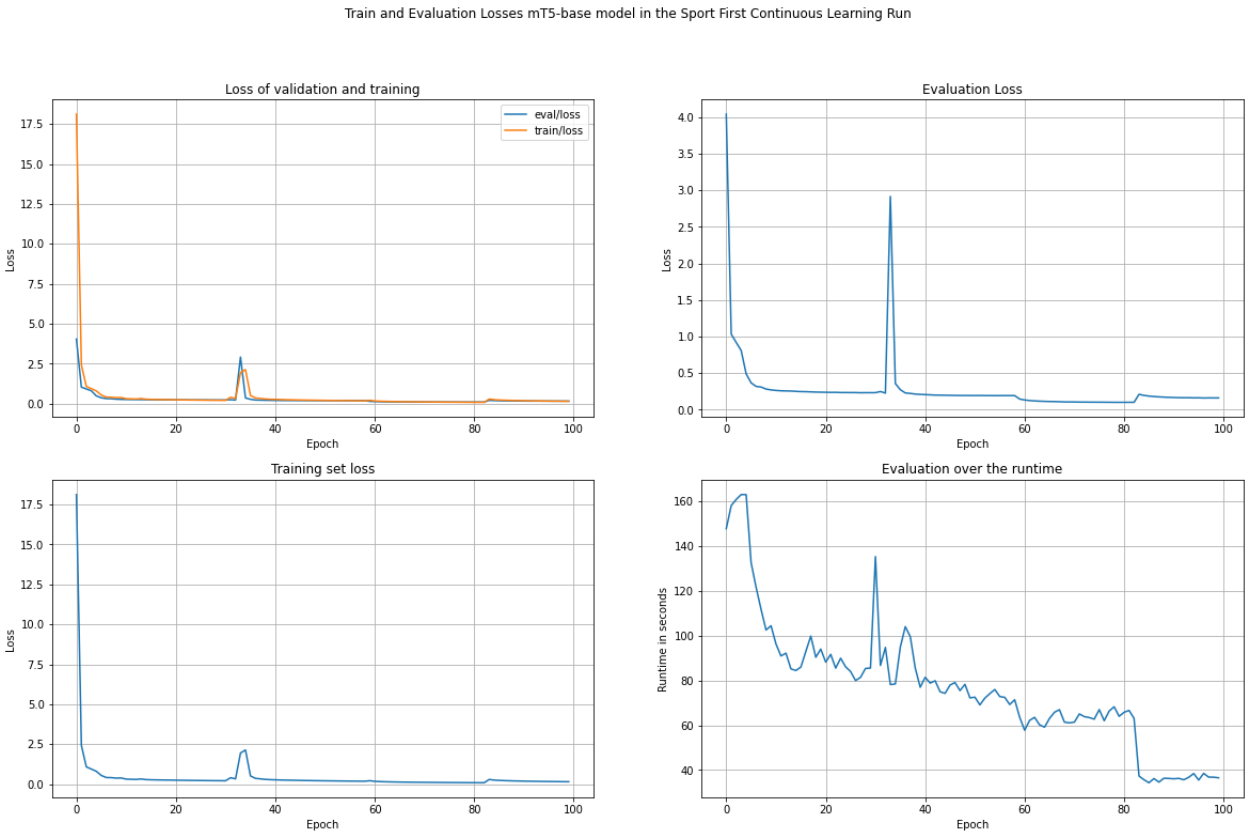

Figure A.4: Training overview of the mT5-base model trained on the original CACAPO dataset trained for the continuous learning experiment, in order of Sport, Stock, Weather, and then Incidents.

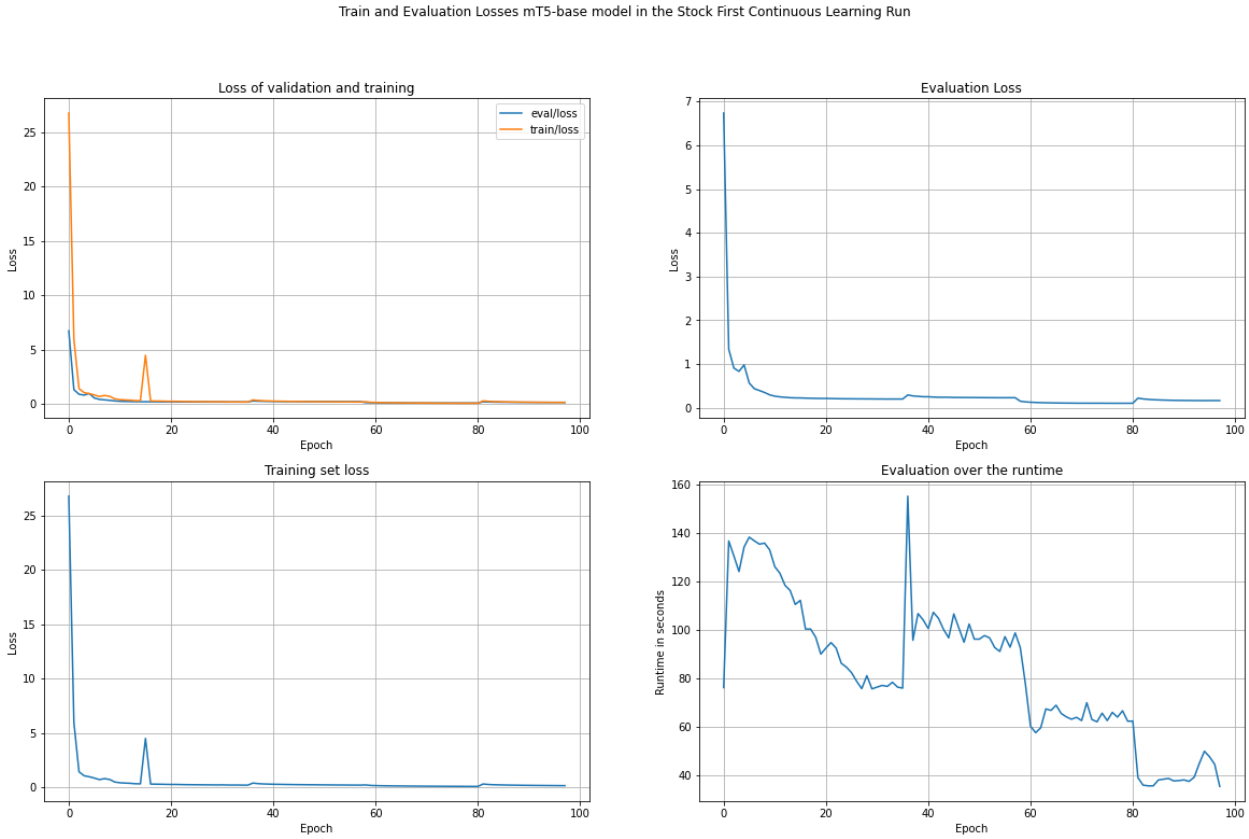

Figure A.5: Training overview of the mT5-base model trained on the original CACAPO dataset trained for the continuous learning experiment, in order of Stock, Sport, Weather, and then Incidents.

A. MODEL TRAINING GRAPHS

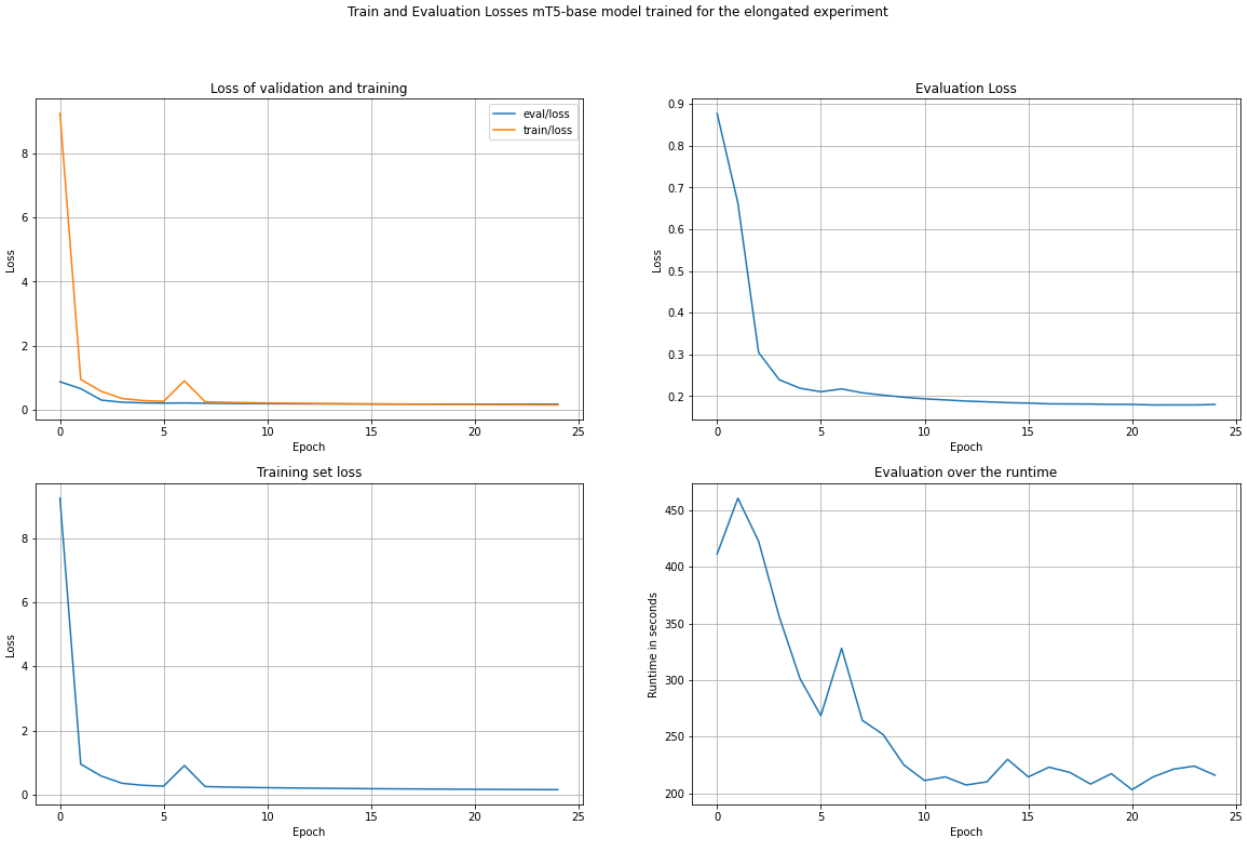

Figure A.6: Training overview of the mT5-base model trained on the elongated training set of CACAPO, used for the elongated experiment

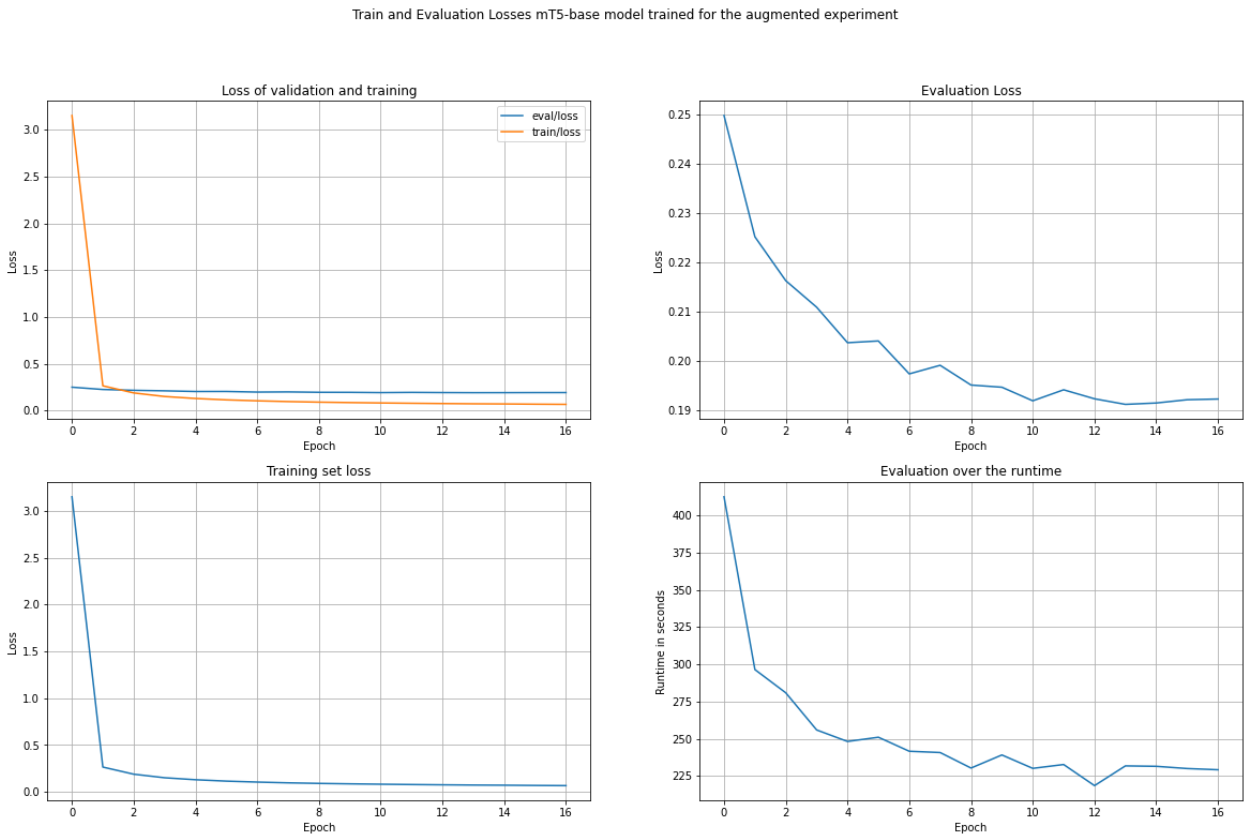

Figure A.7: Training overview of the mT5-base model trained on an augmented training set of CACAPO, used for the augmentation experiment

APPENDIX B

CACAPO ATTRIBUTE OVERVIEW

Subdomain	Attributes
Dutch Stocks	companyName, stockChange, locationName, stockChangePercentage, timePoint, exchangeName, moneyAmount, stockPoints
English Stocks	ORG, DATE, stockChange, PERCENT, exchangeName, LOC, stockPoints, MONEY, TICKER, ORDINAL
Dutch Weather	locationArea, weatherType, timePoint, windAmount, windDirection, cloudAmount, weatherIntensity, weatherFrequency, weatherArea, minimumTemperature, weatherOccurringChance, compassDirection, windChange, cloudType, cloudChange, weatherChange, windSpeedBft, gustAmount, gustVelocity, temperatureChange, temperatureHotCold, precipitationAmount, gustChange, windType, windTurning, snowAmount
English Weather	weatherType, timePoint, locationArea, weatherIntensity, temperatureCelsius, temperatureHotCold, windAmount, weatherOccurringChance, maximumTemperature, cloudAmount, weatherFrequency, temperatureChange, minimumTemperature, windDirection, weatherChange, windSpeedBft, compassDirection, snowAmount, windChange, gustAmount, precipitationAmount, cloudChange, weatherArea, gustVelocity, gustChange, cloudType, sunSetTime, sunRiseTime

Dutch sports	teamName, matchTime, goalName, goalScore, chanceForName, positionOfPlayer, goalType, goalkeeperName, hasWonTeam, finalScore, hasLostTeam, matchDate, assistName, chanceForType, coachName, teamStandings, assistType, playerName, stadiumPlayed, numberOfPoints, homeAway, matchStreakNumber, hasTiedTeam, matchStreakType, hasScored, playerNationality, numberOfSeasonGoals, numberOfMatchesPlayed, numberOfMatchGoals, tackleRecipientName, tackleGiverName, substituteName, chanceForNumber, twiceYellowName, nextMatchTeam, redCardName, refereeName, injuredName, playerAge, disallowedGoalType, injuryType, halfTimeScore, disallowedGoalName, nextMatchHomeAway, suspendedName, nextMatchDate, chanceForNationality, defendedName, formationTeam
English sports	batterName, teamName, pitchResult, pitcherName, inningNumber, runNumber, winLossType, matchDate, hitNumber, pitcherRecord, gameNumber, strikingType, fielderPosition, inningsPitched, strikeTrajectory, scoreTally, winLossRecord, fielderName, RBI, hasWonTeam, finalScore, outNumber, hasLostTeam, homeRunNumber, baseNumber, strikeOutNumber, startsNumber, pitchResultNumber, competitionName, walkNumber, locationPlayed, managerName, scoreNumber, onBaseNumber, pitchType, batterHitsTries, pitcherSaveRecord, ERA, teamStandings, injuryType, homeAway, pitchNumber, standingsGames, hasScored, battingAverage, teamRecord, earnedRunsNumber, throwDirection, pitchCount, battersFacedNumber, pitchesTotalThrown, atBatNumber, gameTally, matchStreakNumber, battingLineupNumber, umpireName, catchType, winningPercentage, matchStreakType, umpireType, unearnedRunsNumber, baseStolen, strikeNumber, retireNumber, stealNumber, baseReachedNumber, leftOnBase, basesRan, catcherName, isOut, errorNumber, numberOfStarts, presidentName, runAverage, inningScore, batterScoreNumber
Dutch incidents	victimStatus, victimGender, victimVehicle, incidentType, location, datetime, suspectVehicle, victimAge, victimAddress, cause, victimDescription, victimAmount, suspectGender, suspectStatus, suspectAge, suspectAddress, suspectDescription, suspectAmount, victimName
English incidents	victimNumber, victimStatus, accidentAddress, shootingType, accidentDate, victimGender, victimAge, takenToHospital, victimName, suspectName, hospitalName, suspectStatus, suspectGender, suspectAge, suspectNumber, victimBased, $victimAge_Group$, victimOccupation, numberOfRoundsFired, suspectWeapon, suspectVehicle, suspectBased, personnelArrivedTime, shootingNumber, prisonName, $suspectAge_Group$, victimRace, suspectRace, suspectDescription, suspectHeight, suspectOccupation, victimVehicle, suspectWeight

Table B.1: Overview of attributes in the CACAPO dataset

APPENDIX C

CONTRASTIVE SEARCH EXPERIMENT

During the creation of the baseline models for this project, models, datasets, and hyperparameters were convened from previous works. One such decision is the choice of decoding strategy for the model to use. Several previous works apply both Beam Search in combination with Nucleus Sampling, however, other decoding strategies have been proposed and argued for increased performance. One such decoding strategy is the use of Contrastive Search instead of Beam Search.

In short, Beam Search chooses the word with the highest likelihood to follow up on the previously generated words. This score is calculated by the model, however, certain words occur more frequently in natural language, thus these words will have a higher likelihood of being generated by the model. This results in model degeneration, which means that the generated text feels unnatural. To solve this issue, a sampling method is used to increase variety, however, in its nature, it also introduces randomness, which could increase the occurrence of out-of-context generations.

To combat the problem of text degeneration, the strategy of Contrastive Search introduces a degeneration penalty. First, the model calculates the probability of a candidate token to be predicted, similar to Beam Search. Then, the candidate token is compared to previously generated tokens and the cosine similarity between these tokens is calculated. The idea behind this additional calculation is that more similar tokens will increase the likelihood of model degeneration, and thus the likelihood the model will choose said candidate token will decrease with a higher cosine similarity. The user can change the impact of this co-similarity by changing the value of α (76).

Su and Collier found that Contrastive Search could be applied to pre-trained models during inference and the users could then achieve better performance on at least four

different tasks compared to Beam Search and Nucleus Sampling, those being open-ended text generation, summarization, machine translation, and code generation. (77) To test the authors' claim on the task of data-to-text, an experiment was conducted using the already fine-tuned T5-base and mT5 models. These models were saved before seeing the test set, and thus by loading them they will have not seen the test set before, thus avoiding contamination. The code was slightly adjusted to change the decoder strategy from Nucleus Sampling and Beam Search to Contrastive Search. After generating the test set, the previously used automatic metrics are used to evaluate the performance, after which the results are compared with the results of the baseline.

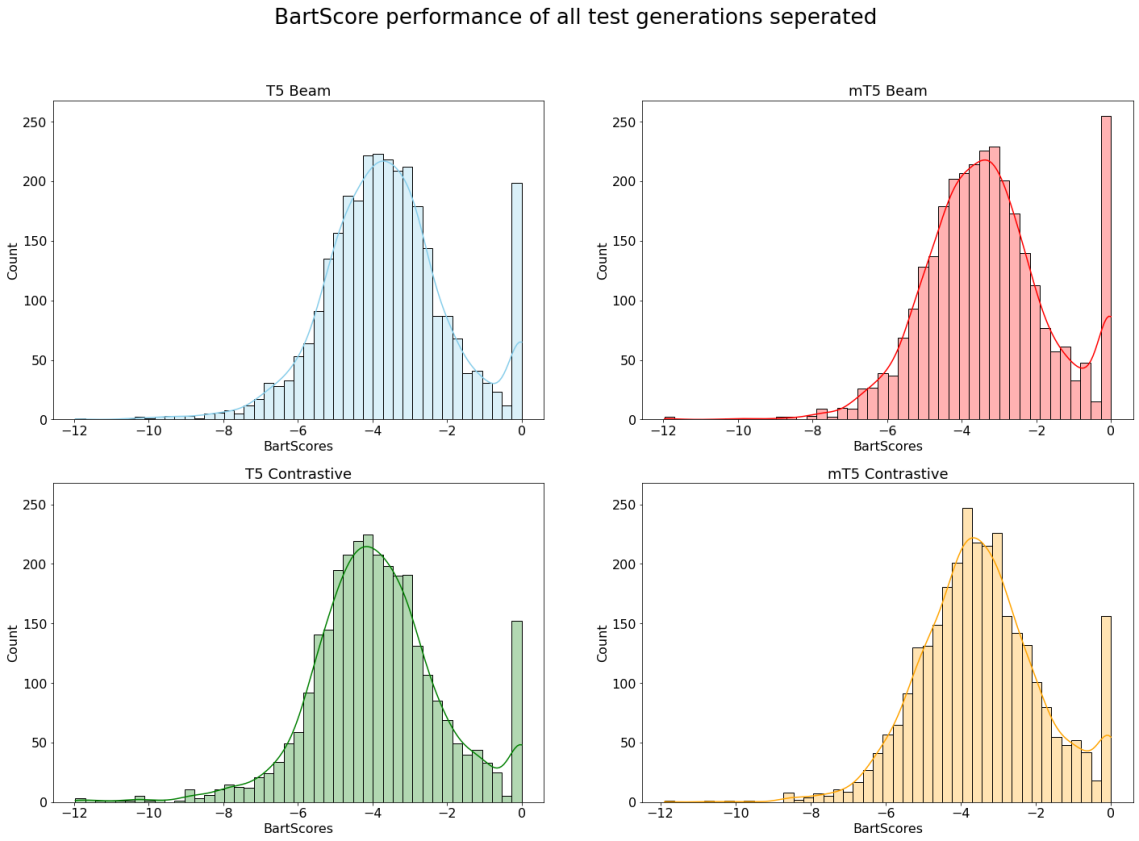

Figure C.1: Overview of BartScore performances for the Contrastive Search experiment

As figure C.1 highlights, the Contrastive Search model's performance was worse than the performance of the baseline. These results highlight that, based on the automatic metrics, using Contrastive Search decoding methods after the model has already been fine-tuned using Beam Search and Nucleus Sampling does not result in better performance in this case. Thus further experiments were implemented on the baseline models using Beam Search in combination with Nucleus Sampling.

APPENDIX D

CONTINUOUS LEARNING EXPERIMENT

During the manual review of the baseline model's generations, an observation was made that attributes were changed for other attributes, as described in section 3.4. A majority of these cases were found in the Dutch subject of Sports and after analysis, it was identified that this subject had the highest amount of unique attributes whilst also having close to the fewest records in the dataset. This sparked the hypothesis that the model might not see enough Dutch sports records at the same time during training, as other subjects and languages could dominate the batches during training.

To test this hypothesis, an experiment was conducted where a clean mT5-base model would be trained on one subject at a time. To set this experiment up, the dataset splits were split once more, such that each subject would have its own training, validation, and test splits. These splits would contain both Dutch and English records, thus splitting the original dataset into four smaller datasets. The overview of these datasets can be found in table D.1.

Domains	Train (Dutch/English)	Validation (Dutch/English)	Test (Dutch/English)
Sports	5360 (1945, 3415)	611 (250, 361)	999 (364, 635)
Stocks	3941 (2292, 1649)	488 (271, 217)	790 (459, 331)
Weather	3848 (2014, 1834)	471 (244, 227)	792 (410, 382)
Incidents	2141 (1116, 1025)	261 (123, 138)	447 (229, 218)

Table D.1: Subject-specific dataset sizes

To test this hypothesis, two different runs were created. First, a clean mT5 base model would be trained on each subject separately, starting with the largest subject and gradually moving to the smallest subject. This meant that the model would be first trained on Sports, then Stocks, then Weather, and lastly Incidents. The second run would have a different order of subjects, that being first Stocks, then Sports, Weather, and finally Incidents. Each model would be trained on one subject at a time, after which the best checkpoint would be used to train on the next subject. This results in four different checkpoints per run, thus eight different models. The results of these runs are visible in figures D.1 and D.2.

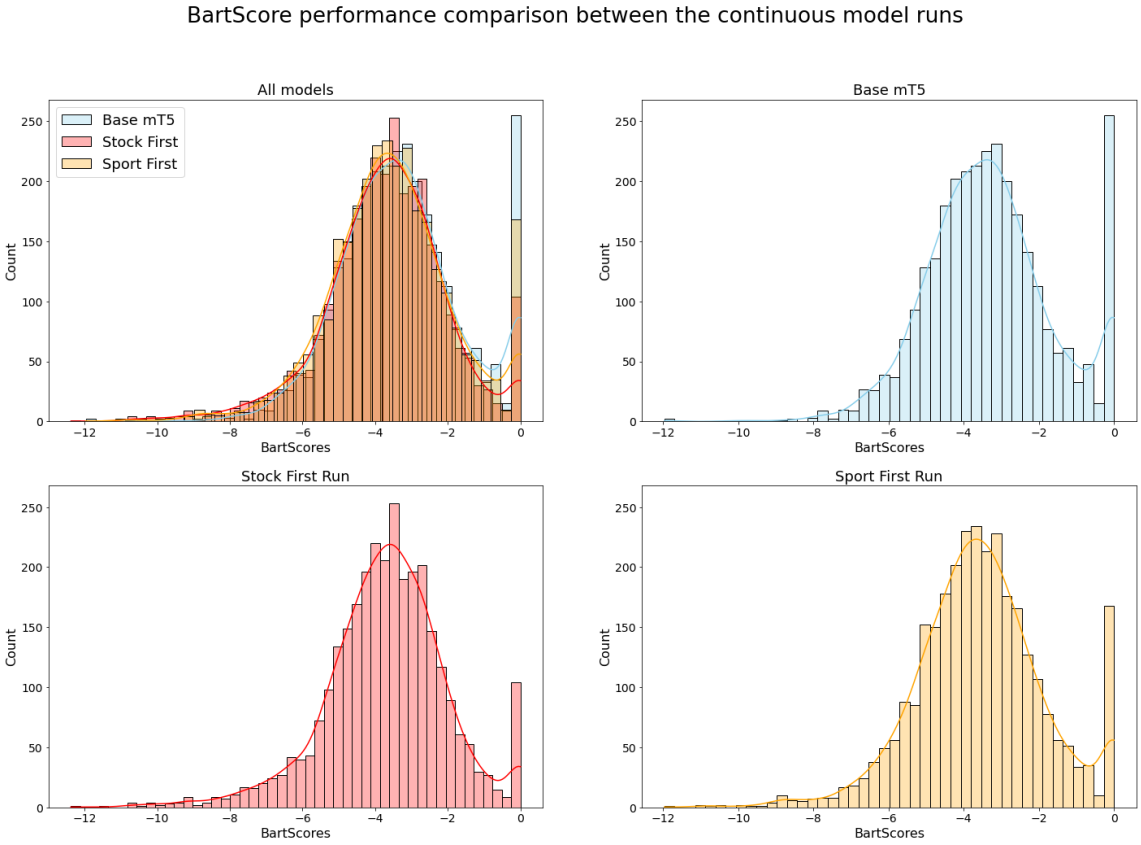

Figure D.1: Overview of BartScore performances for the Continuous Learning experiment

The performance of these models was slightly worse than the baseline run, even though the model of this run trained for significantly more epochs, that being 97 compared to the 29 of the baseline model. Interestingly, the baseline model's performance had more close to perfect BartScores, whilst the models from this experiment has more occurrences across the distribution.

After analysing the BartScore and Rouge performance of the two continuous learning runs it can be concluded that this experiment has not resolved the issue of too many unique attributes and too high a variety of records. These models still make mistakes by forgetting

or switching out attributes with different attributes.

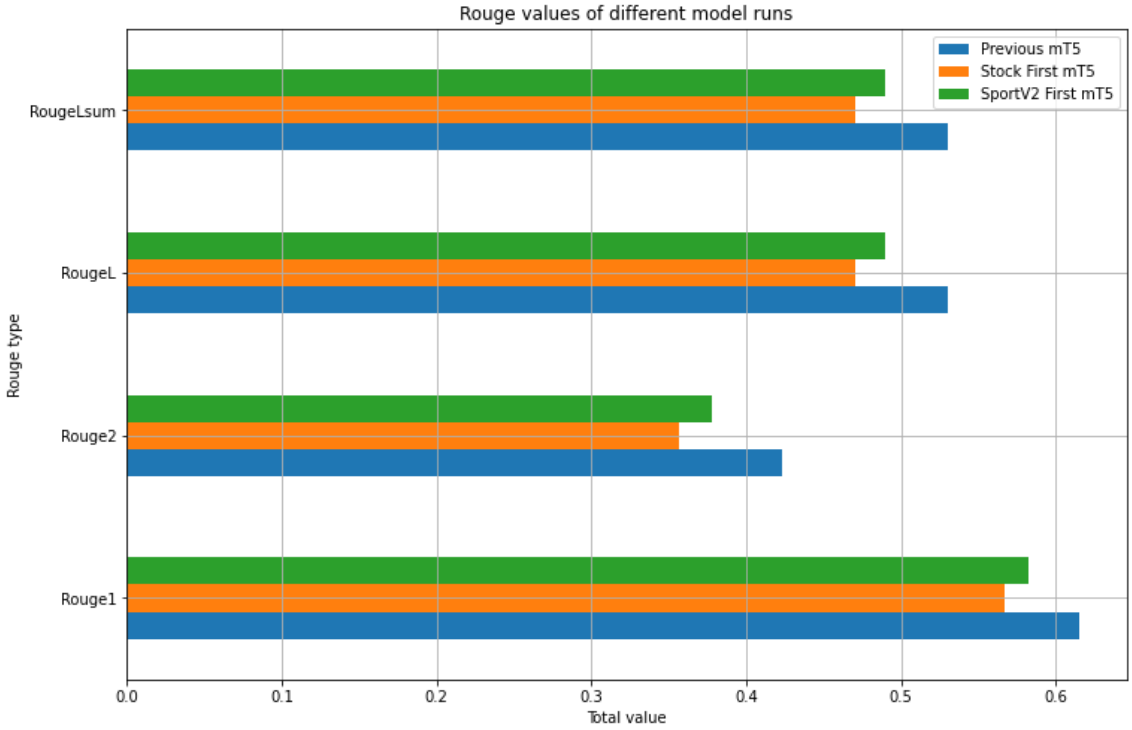

Figure D.2: Rouge score comparisons for the Continuous Learning experiment

APPENDIX E

MODEL PERFORMANCE OVERVIEW

Model	BLEU-4	Meteor	Rouge1	Rouge2	RougeL	RougeLsum	Mean BERTScore
T5-base English	0.173	0.474	0.544	0.349	0.456	0.457	0.911
T5-base Dutch	0.280	0.522	0.592	0.412	0.535	0.536	0.911
Yhavinga T5-base English	0.133	0.421	0.500	0.298	0.421	0.421	0.899
Yhavinga T5-base Dutch	0.285	0.541	0.602	0.429	0.545	0.545	0.911
mT5-base English	0.189	0.512	0.591	0.383	0.488	0.488	0.920
mT5-base Dutch	0.331	0.574	0.641	0.466	0.574	0.575	0.920
mT5-base Augmented English	0.186	0.504	0.574	0.363	0.472	0.472	0.917
mT5-base Augmented Dutch	0.332	0.576	0.640	0.466	0.575	0.576	0.919

E. MODEL PERFORMANCE OVERVIEW

mT5-base Elongated English	0.181	0.502	0.595	0.387	0.494	0.494	0.921
mT5-base Elongated Dutch	0.334	0.577	0.651	0.478	0.591	0.592	0.923
mT5-base StockFirst Continuous English	0.151	0.475	0.548	0.321	0.433	0.433	0.907
mT5-base StockFirst Continuous Dutch	0.232	0.518	0.587	0.392	0.507	0.508	0.907
mT5-base SportFirst Continuous English	0.154	0.473	0.554	0.332	0.442	0.441	0.909
mT5-base SportFirst Continuous Dutch	0.273	0.543	0.611	0.428	0.540	0.541	0.913
Bart-base English	0.213	0.544	0.620	0.415	0.516	0.516	0.928
Bart-base Dutch	0.320	0.577	0.656	0.479	0.590	0.590	0.924

Table E.1: First Half Results of the different models trained in this project on the original CACAPO test set, which has been split per language ($N_{English} = 1566$, $N_{Dutch} = 1462$). A higher value is best.

Model	Mean BARTScore	Sum Prec	Sum Recall	Sum F score	PARENT Precision	PARENT Recall	PARENT F score
T5-base English	-7.462	1215.213	310.520	381.660	0.776	0.198	0.244
T5-base Dutch	-5.685	1235.074	378.086	460.124	0.845	0.259	0.342
Yhavinga T5-base English	-7.301	1236.474	224.860	291.444	0.790	0.144	0.186
Yhavinga T5-base Dutch	-5.759	1189.972	421.723	500.847	0.814	0.289	0.343
mT5-base English	-7.324	1231.549	340.077	417.288	0.786	0.217	0.267
mT5-base Dutch	-5.588	1208.850	471.472	551.924	0.827	0.326	0.378
mT5-base Augmented English	-7.378	1216.670	323.365	404.035	0.777	0.207	0.259
mT5-base Augmented Dutch	-5.570	1216.636	471.115	554.068	0.832	0.322	0.379
mT5-base Elongated English	-7.303	1249.526	339.635	419.855	0.798	0.217	0.268
mT5-base Elongated Dutch	-5.481	1220.293	447.416	557.824	0.835	0.327	0.382
mT5-base StockFirst Continuous English	-7.460	1236.815	252.014	327.414	0.790	0.161	0.209
mT5-base StockFirst Continuous Dutch	-5.984	1218.906	352.713	434.511	0.834	0.241	0.297

mT5-base SportFirst Continuous English	-7.441	1245.640	266.608	336.771	0.795	0.170	0.215
mT5-base SportFirst Continuous Dutch	-5.759	1225.431	404.259	486.373	0.838	0.277	0.333
Bart-base English	-7.288	1202.468	388.164	462.340	0.768	0.248	0.295
Bart-base Dutch	-5.512	1237.911	459.551	541.423	0.847	0.314	0.370

Table E.2: Second Half Results of the different models trained in this project on the original CACAPO test set, which has been split per language ($N_{English} = 1566$, $N_{Dutch} = 1462$). A higher value is best.