Master Thesis

Multidomain Story Generation using pre-trained Language Models

Tim Ferdinand Elisabeth Debets

Thesis submitted in partial fulfillment
of the requirements for the degree of
Master of Science of Artificial Intelligence
at the Department of Data Science and Knowledge Engineering
of the Maastricht University

Maastricht University
Faculty of Science and Engineering
Department of Data Science and Knowledge Engineering

July 14, 2022

Abstract

Story Generation is a sub area in Natural Language Generation, and introduces challenges not encountered by other research areas. The goal is to generate creative, diverse and coherent stories. The current state-of-the-art models only generate stories based on a single domain. This research introduces multidomain Story Generation, and to train this model, two new datasets have been created, a Fairy Tales and Fantasy dataset.

First, five different pre-trained language models, with less than 350 million parameters, are fine-tuned on a single domain. The models based on a decoder-only Transformer architecture, GPT models, are underperforming compared to language models that incorporate the full Transformer architecture. The GPT models tend to generate stories in which the trigram diversity is too diverse, and hence, the generated stories do not feel natural to read. The grammar score of the three smallest models tend to decrease when fine-tuned for more than 5 epochs, while the grammar score of the two bigger models only increases. In comparison with two state-of the art models, the T5 Base model performs only slightly worse, based on linguistic analysis, while the generated stories of this model are more readable.

After training the T5 Base model to be multidomain, several enhancements based on text properties, and a discriminator have been implemented to improve the quality of the generated texts. The linguistic analysis shows that including these enhancements generate stories that are close to the gold sentences. When evaluating the generated stories to be as human-like as possible, the classifier has difficulties distinguishing generated text and human written text, when using the discriminator as an enhancement. The classifier only scored 61 % accuracy, compared to 95 % for the state-of-the-art models, while the classifier correctly classified to the correct domain with an accuracy of 74 %. Using beam search during the decoding process only worsens the generated text quality, but the using synthetic data for a second round of fine-tuning will improve the multidomain model, when no enhancements are utilized.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Story generation strives to generate stories automatically (Fan, Lewis, & Dauphin, 2018), and ideally these stories are compelling to their readers. A story can be described as a series of events characterized by a set of sentences. This generation process introduces challenges to existing Natural Language Generation Models. Comparing this generation task to more constrained text generation tasks, i.e. summarization and machine translation, which are based on existing content, story generation has an open-ended nature: diversity, creativity and coherency are desired properties in stories.

Current story generating models are limited to a single domain, the domain of the training data (Fan et al., 2018) (Bensaid, Martino, Hoover, Andreas, & Strobelt, 2021). Furthermore, a lot of different ideas exist on how to incorporate the input to the model: a premise of the story or the start of a sentence (Fan et al., 2018) (Bensaid et al., 2021) (Yao et al., 2018), a series of short descriptions (Ippolito, Grangier, Callison-Burch, & Eck, 2019), the start and the end of the story (Ippolito et al., 2019) (Wang, Durrett, & Erk, 2020), having a goal and context to guide the model (Alabdulkarim, Li, Martin, & Riedl, 2021) or having a series of events that the story should contain (Rashkin, Celikyilmaz, Choi, & Gao, 2020). Language models based on the Transformer model occur frequently (Vaswani et al., 2017) (Fan et al., 2018) (Ippolito et al., 2019). Other language models often utilized are a variant of a decoder-only Transformer, such as GPT-2 (Bensaid et al., 2021) (Wang et al., 2020).

Unlike machine translation models or summarization models, no references exist to evaluate stories, and evaluation metrics such as BLEU and ROUGE cannot be consulted. Therefore, evaluating stories is usually performed by human evaluation (Bensaid et al., 2021) (Wang et al., 2020). This evaluation method is time-consuming and costly. Therefore, another evaluation method that should be considered is Linguistic analysis. This method evaluates the quality of the generated stories directly (Roemmele, Gordon, & Swanson, 2017).

In this research, a multidomain model has been developed that is able to generate stories based on a prompt and a domain. For the training of this model, two new datasets have been created to support the multidomain aspect.

Furthermore, several pre-trained language models are compared with each other to evaluate their performance. This comparison determines which pre-trained language model is utilized to be developed as a multidomain story generator. To further improve the quality of the generated stories, enhancements to help during the decoding process have been implemented. Finally, based on linguistic analysis and classifiers, these models are evaluated.

A multidomain story generator can add the next values. Creating a model which is able to generate diverse, creative and coherent stories in a multidomain would be a desirable model. Such a model can support writers by suggesting clear and new ideas, when dealing with a creative slump, while continuing on previous written content, and guide writers to exciting and entertaining directions relevant to the writer's work. An interactive platform could be created to support writers, and where writers can post results of their stories. It can become a hub for writers and readers. Furthermore, the ideas behind developing a multidomain story generator can provide insight in the further development of language models.

(Fan et al., 2018) designed a model for guided and creative story generation. Although the generated dataset is diverse, there is no detail to which domain a story belongs. The whole dataset is just a mix of different stories. Recently, (Bensaid et al., 2021) developed a multimodal model, which not only generates texts, but the model generates images as well to further enhance the stories. This model uses a single and specific domain for story generation. In conjunction with the aforementioned motivation, this leads to the following problem statement of this thesis:

"How can a Story Generator be developed that is guided by a prompt and be multidomain?"

The first step is to discover which different methods of story generation exist.

Research question 1 "What methods do exist to build a Story Generator?"

The story generator should be conditioned on a prompt, the model should be guided. This rises the following research question:

Research question 2 "How can the Story Generator be influenced by a prompt?"

Since the goal is to develop a model that is a multidomain story generator, the next research question is:

Research question 3 "How can the genre be included while generating a story?"

Both (Fan et al., 2018) and (Bensaid et al., 2021) use human evaluation as the main focus to test the quality of their stories. Since this can be time-consuming and costly, the next research question addresses this problem:

Research question 4 "What methods exist to automatically evaluate the generated stories?"

Chapter 2 provides an introduction to the different architectures used in this model, different story generation techniques are introduced, the different consulted language models are briefly described, and has an overview of different evaluation techniques. Chapter 3 explains the development cycle of the multidomain story generator. Furthermore, the datasets are introduced, as well as the different evaluation metrics. This section also introduces different enhancements to further improve the generation process. Finally, the experiments to investigate the influence of beam size and the use of synthetic data are explained. The results of the experiments and the discussions thereof are presented in Chapter 4. Chapter 5 summarizes the key findings and outlines the directions for future research.

Chapter 2

Background and Related Work

This chapter provides the context of existing literature for the thesis. The first section explains the Transformer model, including self-attention and the architecture. Section 2.2 introduces the background of the current models being used to generate short stories. Section 2.3 zooms in on the different language models investigated in this research. The next section, Section 2.4, has an overview of several decoding properties. Finally, Section 2.5 presents a brief survey of different evaluating techniques for text generation, with an emphasis on story generation.

2.1 Transformer

In (Vaswani et al., 2017) a new deep learning architecture got introduced, the Transformer model, which is a Sequence2Sequence model (Seq2Seq) based on an encoder and a decoder. Furthermore, this model adopts the newly introduced self-attention mechanism. This mechanism differentially weighs the significance of the input data. The input of the Transformer model is processed in its entirety, and the attention mechanism provides the context of any position in the input sequence. First, the attention mechanism will be explained, followed by a description of the architecture of the Transformer model.

2.1.1 Self-attention

(Vaswani et al., 2017) introduced the self-attention mechanism. This mechanism is able to retrieve the contextual information of a word in a sentence. The module compares every word in the sentence to every other word in the same sentence, including itself. During this comparison, the word embeddings of all words in the sentence are reweighed to include the contextual relevance. The logic behind comparing the word with itself is to determine the exact meaning

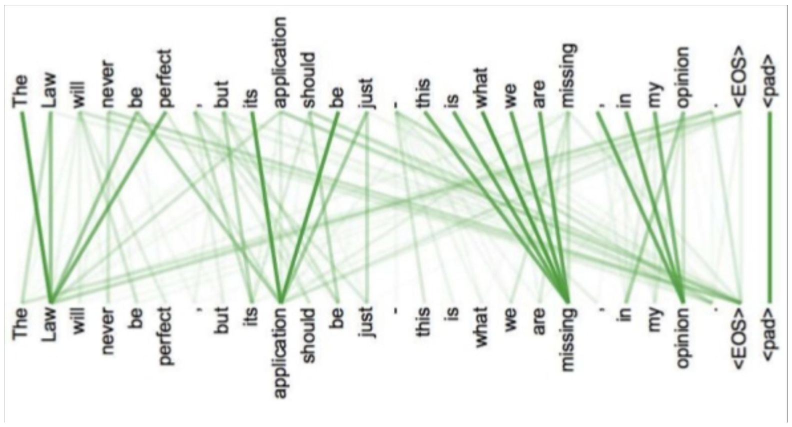

Figure 2.1: The calculated attention for a sentence. Thicker lines indicate that more attention should be paid to that word.[a]

[a]Retrieved from a presentation by (Vaswani et al., 2017): https://www.slideshare.net/ilblackdragon/attention-is-all-you-need

of the word, since a single word can have different meanings, depending on the context of the word, i.e. I am going to the bank to retrieve money, and the bank of a river.

Figure 2.1 contains calculated attention scores for a sentence. The attentions scores are represented by the lines. How thicker a line, the more attention should be paid to this word. For a model to be capable of learning the contextual connections between words, three weight matrices are introduced, called the query weight matrix, key weight matrix and value weight matrix. These weight matrices are learned during training, and using matrices enables simultaneous calculations for the whole sequence. The input word embeddings are all multiplied with each of the weight matrices separately, to get three new matrices, the query, key and value matrix. Equations 2.1 - 2.3 show these multiplications, whereas X is the word embedding matrix of the input, W^Q is the query weight matrix, W^K is the key weight matrix, W^V is the value weight matrix.

$$Q = X \times W^Q \tag{2.1}$$

$$K = X \times W^K \tag{2.2}$$

$$V = X \times W^V \tag{2.3}$$

The next step is to deduce to which words the Transformer should focus on, pay attention to, for a specific word. Ideally, all these words refer to this specific word. Calculating the dot product similarity is the first step to calculate the attention scores. The result of the dot product similarity indicates to which words the Transformer should pay attention to. A higher score indicates that more attention should be paid to this specific word. These products are calculated by multiplying the query matrix with the transpose of the key matrix.

After calculating all these dot products, the values are normalized with the softmax function, to focus on the relevant words, and to drown-out irrelevant words. Equation 2.4 shows the equation used to calculate these values, where d_k is the dimension of key vectors. Dividing by this dimension leads to more stable gradients.

$$Attention(Q, K) = softmax(\frac{Q \times K^T}{\sqrt{d_k}}) \tag{2.4}$$

Finally, to calculate the new word embeddings, the context vector, the previously calculated attention matrix is multiplied by the originally calculated value matrix. The result of this final multiplication, Equation 2.5, are the newly weighted word embeddings, Z.

$$Z = Attention(Q, K) \times V \tag{2.5}$$

Finally, one of these self-attention blocks might not be capable of paying attention to several important words and might not make an observable change to their respective word embeddings. To resolve this problem, multi-headed attention is introduced. Multi-head attention exists out of several attention blocks which run in parallel, each calculating attention scores independently. Therefore, using this mechanism expands the model's ability to focus on different positions. The three weight matrices are not shared between the different self-attention blocks, and are all randomly initialized. The results of these self-attention blocks are concatenated together to retrieve the new word embeddings, the new context vectors.

2.1.2 Architecture

The architecture of the Transformer model exists out of two main parts, the encoder and the decoder. The encoder actually exists out of several layers of encoder blocks, the same applies for the decoder, in (Vaswani et al., 2017) 6 encoders and decoders blocks are stacked upon each other.

The encoders exist out of two main parts, a self-attention layer and a Feed Forward Neural Network layer, the weights are not being shared between different encoders. To take into account the order of the words in the input sequence, the Transformer adds positional encodings to the word embeddings. The intuition behind adding these positional encoding, is to provide a meaningful distance between the embeddings once they are projected in the self-attention mechanism. Finally, a residual connection is added around each encoder, followed by a normalization step.

In the decoder, between the self-attention layer and the Feed Forward Neural Network layer, an additional attention layer is introduced, an Encoder-Decoder attention layer, to help the decoder focus on relevant parts of the input sequence

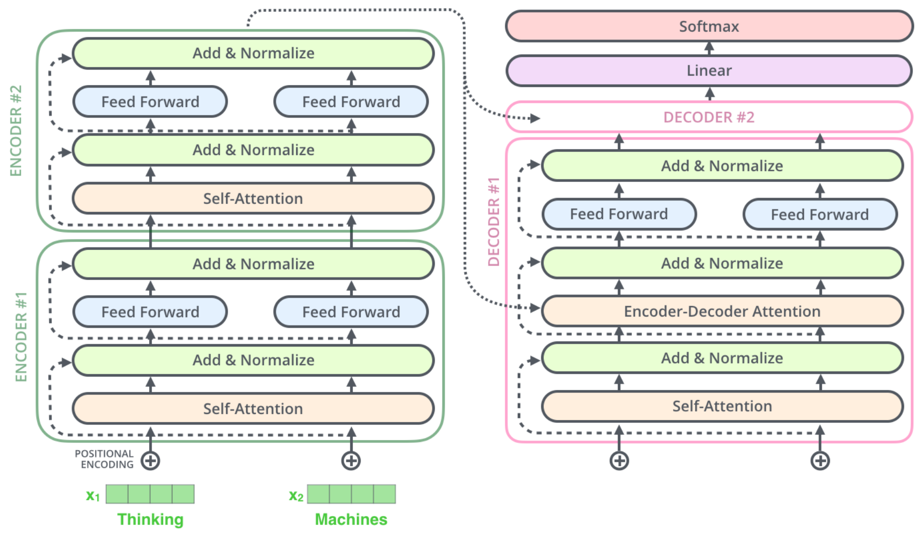

Figure 2.2: Architecture of a Transformer model.[a]

[a]Figure from https://jalammar.github.io/illustrated-transformer/

The output of the topmost encoder is transformed into a set of key and value matrices used in the Encoder-Decoder attention mechanism of all the decoders.

The input of the decoder is the output of previously generated words, and positional encodings are embedded to the decoder inputs. The output of each decoder is being fed to the next decoder, bubbling up the decoding results. Finally, future words are being masked in the decoder layer, since these are unknown. The final part of the Transformer is a Linear Layer, a fully connected layer, followed by a softmax layer. Figure 2.2 shows the full architecture of a Transformer, more encoders and decoders can be added.

2.2 Story Generation

Natural Language Generation is a field that focuses on generating natural language output, including chatbots (Adamopoulou & Moussiades, 2020), machine translation (Garg & Agarwal, 2019) and story generation (Fan et al., 2018). To be able to generate stories, a model needs to be creative and generate coherent and fluent passages of text. Normally, input is provided for a specific model to be able to generate stories. Only how the input is utilized differs often. The prompt can be used as a guideline or outline for the story (Fan et al., 2018), or the model uses the input as the start of the story and continues the story based on the input (Bensaid et al., 2021), i.e. having the prompt: ”A cat and a dog are walking”, the model can generate: ”tiptoeing down one side, So they cannot hear me.”. This section shows an overview of the different methods to utilize a prompt and briefly introduces the architectures behind these models. An example of the input and the generated for all the different methods can be found in Appendix A.1.

2.2.1 Fusion Model

In the paper by (Fan et al., 2018), the prompt is used as a premise to generate a passage of text, using the premise as inspiration for the story. For this model, a Seq2Seq model is used, to be specific, a convolutional Seq2Seq model using an encoder-decoder structure (Gehring, Auli, Grangier, Yarats, & Dauphin, 2017). To enable unbounded context of the text, the decoder uses the self-attention mechanism (Vaswani et al., 2017), with multi-head attention. The queries, keys and values are given by Gated Linear Unit activations.

Furthermore, the heads operate at different scales, each head is downsampled by a different amount. Using this technique, the heads are encouraged to attend to different information. This Seq2Seq model is fused with a pre-trained Seq2Seq model, whereas the hidden states of the pre-trained model are accessible to the new Seq2Seq model. The fusion can be seen as residual learning, the second model focuses on learning what the first model failed to learn.

2.2.2 FairyTailor

Another approach to generate stories is to use the prompt as the start of the story and continue generating text based on this first sentence or first few words. In the Fairytailor model (Bensaid et al., 2021), this approach has been chosen. Furthermore, the model will generate images that fit the story, being a multimodal generative framework. For the text generation part, the model is a fine-tuned GPT-2 model. The training data for fine-tuning the models exists out of passages of fairy tales. To further improve the text generation quality, several candidates are being generated and re-ranked according to various metrics: readability, positiveness, diversity, simplicity, and tale-like manner. These scores are max-normalized such that all scores contribute equally.

2.2.3 Other models

Other methods for using the input different are available. For this thesis, these methods are not being investigated in detail, since the use of the input differs significantly. A brief survey of these methods can be found below. Other methods not mentioned here can be found in (Herrera-González, Gelbukh, & Calvo, 2020) (Ansag & Gonzalez, 2021) (Alhussain & Azmi, 2021).

Plan and Write

The input in the model introduced by (Yao et al., 2018) is the title of the story. First, this title is utilized to generate a storyline. This storyline creates the general direction of the story, and exists out of a set of words. Finally, both the title and storyline are used as an input to a second model to generate the story.

For the planning of the storyline and the generation of the story, two bidirectional LSTMs (Sherstinsky, 2018) are trained. The first LSTM encodes the title, and generates words for the storyline, while the second LSTM encodes both the title and the storyline. The Seq2Seq model is then trained to minimize the negative log-probability of the stories in the training data to optimize the generation process.

Story Generation from Sequence of Independent Short Descriptions

(Jain et al., 2017) use a series of independent descriptions as the input. These inputs are fed into a deep recurrent neural network (Sherstinsky, 2018) to encode the variable length descriptions to latent representations and decodes them to produce comprehensive story like summaries. The encoder is bidirectional and uses Gated recurrent unit (GRU) cells. The decoder utilizes the attention mechanism and GRU cells.

Unsupervised Infilling

The input of the model introduced by (Ippolito et al., 2019) exists out of two parts: the start of the story, and the end of the story. The developed model predicts the missing span, the middle part, of the story. First, a hierarchical model selects a set of rare words and generates likely sequences conditioned on this specific set of rare words. Selecting the rare words has been relegated to a word-sampling model, a standard Transformer. The conditional generation is based on a self-attention Transformer with positional encodings.

Narrative Interpolation for Generating and Understanding Stories

(Wang et al., 2020) developed an interpolation model based on GPT-2. This model conditions on the previous sentence and the next sentence and predicts the text that fills in the gap. Furthermore, to increase the coherence of the generated story with the previous and future sentence, a Coherence Ranker is implemented to score the coherence of the whole story.

Goal-Directed Story Generation: Augmenting Generative Language Models with Reinforcement Learning

(Alabdulkarim et al., 2021) use deep reinforcement learning and reward shaping to control the plot of generated stories. The input of the model are the goal and the context of the story. The first automated technique utilizes policy optimization for fine-tuning an existing transformer-based language model. A second automated technique extracts a knowledge graph from the story, using graph attention, to select a candidate to continue the story.

PlotMachines: Outline-Conditioned Generation with Dynamic Plot State Tracking

(Rashkin et al., 2020) developed a model in which the input is the outline, several short ideas, of a story, and the output is the generated story. The chosen architecture is Transformer-based, built on top of the GPT model, with a memory mechanism to keep track of plot elements.

All the aforementioned models are based on a single domain (Bensaid et al., 2021), or do not have an indication with respect to the domain (Fan et al., 2018). This thesis adds the multidomain aspect in story generation when using pre-trained language models as the base, and researches how this aspect can be implemented.

2.3 Language Models

Language models represent the fluency of a language. The model tries to predict the probability of a sentence. These models are frequently used in Natural Language Generation (Adamopoulou & Moussiades, 2020) (Garg & Agarwal, 2019) (Bensaid et al., 2021). The probability of a sentence, Equation 2.3, can be computed using the chain rule.

$$P(w_1 w_2 \cdots w_N) = \prod_i P(w_i | w_1 \cdots w_{i-1})$$

Many pre-trained models do exist, several of which are explained in this section.

2.3.1 GPT-2

The OpenAI GPT-2 model is a language model with an ability of writing coherent text, and exceeded the quality of the language models at that time (Radford et al., 2019). The architecture is not novel, it is almost identical with a decoder-only Transformer (Liu et al., 2018). Figure 2.3 shows the decoder-only Transformer architecture. The GPT-2 model has been trained on a massive dataset of approximately 40 GB of text. A downside of the GPT-2 model is its massive size. Using 1.5 Billion parameters, it could take up to 6.5 GB of storage space. Therefore, smaller models have been released, the smallest model takes up 500 MB of storage space. Table 2.1 has an overview of these GPT-2 models with the number of parameters. The difference between all these models are the number of decoder layers, varying between 12 and 48, with steps of 12.

2.3.2 GPT-Neo

The next substantial language model that got released, is the successor of GPT-2, namely GPT-3 (Brown et al., 2020). This model uses the same attention-based architecture as their predecessor GPT-2, and has been trained on 45 TB of

Model	Parameters (Millions)
GPT-2 Small	117
GPT-2 Medium	345
GPT-2 Large	762
GPT-2 Extra Large	1542
GPT-Neo Small	125
GPT-Neo Medium	1500
GPT-Neo Large	2700
BERT base	110
BERT Large	354
T5 Small	60
T5 Base	220
T5 Large	770
T5 3B	3000
T5 11B	11000

Table 2.1: The different existing GPT-2, GPT-Neo, BERT and T5 models with the number of parameters for each of them respectively.

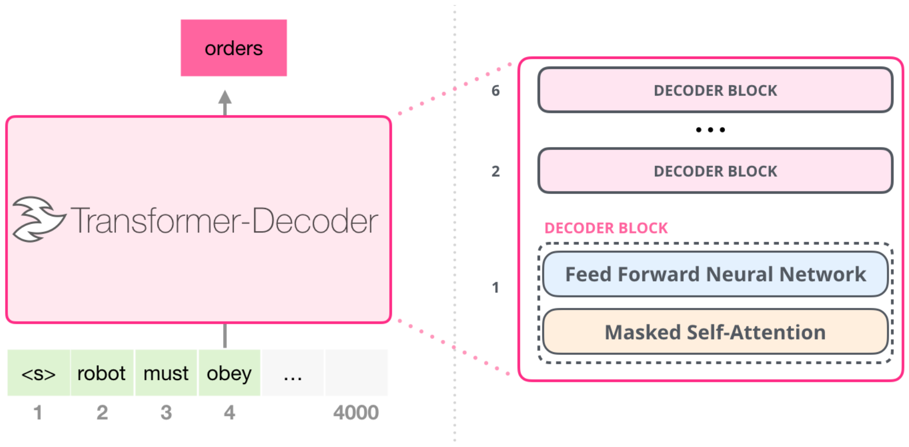

Figure 2.3: Architecture of a Transformer-Decoder model.[a]

[a]Figure from https://jalammar.github.io/illustrated-gpt2/

text data. This model has eight variants, differing from 125 Million parameters, to 175 Billion parameters, and ranges from 12 attention layers, to 96 attention layers. Unfortunately, this model is currently not open-sourced.

Therefore, Black et al. (Black, Gao, Wang, Leahy, & Biderman, 2021) developed a model, GPT-Neo, which is similar to the GPT-3 model, and this specific model is open-sourced. GPT-Neo is trained on the dataset the Pile (Gao et al., 2020), existing out of 825 GB of data. The performance of GPT-Neo is not far

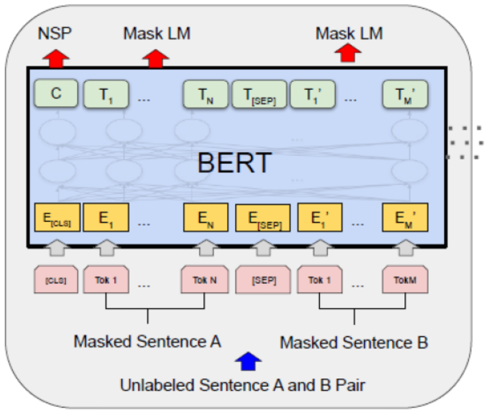

Figure 2.4: Architecture of the BERT model.

behind the GPT-3 model[1], and currently three of the models, Table 2.1, are readily available on HuggingFace (Wolf et al., 2019).

2.3.3 BERT

BERT (Bidirectional Encoder Representations from Transformers) (Devlin, Chang, Lee, & Toutanova, 2018) is applying a bidirectional training of the Transformer model to language models. Aforesaid model improves upon the knowledge of the language context when trained bidirectionally, compared to single-direction language models. The architecture exist of only the encoder part of the Transformer model, in contrast to the GPT models, which only utilizes the decoder part. The architecture of BERT is shown in Figure 2.4.

BERT uses two training strategies. The first strategy, before sentences are fed into the model, masks 15 percent of the words in each sequence. The model attempts to predict the masked word based on the context provided by the rest of the sentence. The next strategy is for the model to be able to predict if two sentences are subsequent sentences. The available BERT models can be found in Table 2.1.

2.3.4 T5

The next model being described and utilized in this thesis is the T5 (Raffel et al., 2019), Text-to-Text-Transfer-Transformer, model. In comparison with GPT-2, GPT-3, GPT-Neo and BERT, the T5 model utilizes the full Transformer architecture instead of only using the decoder or encoder part respectively. The T5 model trains the same objective as the BERT model, being a Masked Language model. During training, words a being masked, and the objective for the model

[1]Table of results can be found at https://github.com/EleutherAI/gpt-neo

is to correctly predict the masked tokens. There is a subtle difference between the masking of words. Whereas BERT uses a mask token for each word, the T5 model replaces multiple consecutive tokens with a single mask token.

Furthermore, the T5 model has been trained on approximately 700 GB of data. Table 2.1 shows the different available models following the T5 architecture and the number of parameters for each of these models.

This thesis empirically investigates the performance differences of the two smallest GPT-2 models, the smallest GPT-Neo model, and the two smallest T5 models. This investigation exists out of two parts: what is the quality of generated stories of these models without any fine-tuning, and how do these models respond to being fine-tuned for story generation. The best model of this investigation is then utilized to be developed to a multidomain story generator.

Since the architecture of BERT is based on the Transformer encoder, the model is able to process each input token in the full context of all text tokens. Therefore, a BERT model can be used on a variety of NLP tasks. In this research, BERT is utilized as a text classifier to distinguish between human written and generated text, and to predict the domain of stories generated by the multidomain model.

2.4 Decoding Properties

Several techniques exist to guide the decoding process during generation. Another possibility is to re-rank several candidates after generation based on some criteria. For guiding the decoding process, Beam Search and Sampling are explained. Followed by a short overview of several re-ranking techniques.

2.4.1 Beam Search

One algorithm that can be utilized during decoding is Greedy Search. Greedy search takes the word with the highest probability at each position in the sequence, and predicts this word in the output sequence. At the current step in the sequence, this strategy might be optimal, but when continuing through the sequence, the resulted sequence might turn out to be worse than anticipated.

An algorithm, which resolves this problem, is beam search. This algorithm selects multiple tokens for a position based on the conditional probability. The algorithm remembers the best N alternatives based on the hyperparameter beam width, and when identifying the new word in the sequences, it utilizes the N best sequences so far and considers the probabilities of all the combinations of all preceding words, including the word at the current position, to identify the new N the best alternatives.

The width or size of beam search is of influence on the quality of the generation task. (Koehn & Knowles, 2017) reported that the size of the beam influences the quality of Machine Translation. They showed that increasing the beam size after reaching an optimal width, does not improve the quality of the translation, but the quality decreases with bigger beam sizes. The translation tends to be worse when the beam size surpasses the optimal beam size. This result has been confirmed by (Cohen & Beck, 2018).

2.4.2 Sampling

Another method utilized to generate text is Sampling. In Sampling the next word is chosen based on its conditional probability distribution, Equation 2.6.

$$w_t \sim P(w|w_{1:t-1}) \tag{2.6}$$

Unfortunately, (Holtzman, Buys, Forbes, & Choi, 2019) shows that models using this method often generate incoherent gibberish. One trick to deal with this problem is using a *temperature* to make the probability distribution sharper. This increases the likelihood of high probability words and decreases the likelihood of low probability words.

Another method, based on Sampling, is Top-K Sampling (Fan et al., 2018). In this algorithm, the K most likely words are filtered, and the probability mass function is redistributed among those K words. This scheme got adopted by GPT-2. A problem that could arise with this scheme is that limiting a sample pool could endanger the model to produce gibberish for sharp distributions and limit the creativity for flat distributions.

Top-p sampling is a scheme that tries to counter this problem. In top-p sampling (Holtzman et al., 2019), instead of choosing the top K words, the set of words whose cumulative probability exceeds the probability p are chosen. After choosing these words, the probability mass function is redistributed again. Combining top-K and top-p is a possibility as well. Then first the top-K words are chosen, followed by top-p sampling scheme.

Finally, beam search can be combined with sampling as well. So instead of choosing the most likely tokens, the tokens are chosen based on its conditional probability distribution.

2.4.3 Re-ranking candidates

Another technique used during decoding is to generate several candidates and re-rank them based on a score. By re-ranking the candidates, the outputted text can be further guided, depending on the properties used during scoring. In machine translation, re-ranking candidates has also been utilized. (Kumar & Byrne, 2004) use a hierarchy of loss functions that incorporate different levels of linguistic information. (Blain, 2017) also make use of different automatic evaluation metrics to re-rank candidates in Machine Translation. These automatic metrics include BLEU, BEER and CHrF3. This ranker is implemented in the following way: several hypotheses are being generated, and for each hypothesis, a similarity score is calculated with respect to the other hypotheses. The hypothesis that is most similar to the other candidates is returned.

A different similarity measure is used by (Borgeaud & Emerson, 2019) to re-rank candidates. The method used is called Range Voting, where voters can vote for the most likely candidate. The overall score is calculated based on the amount of voters and a similarity measure, i.e. BLEU.

Different methods exist for re-ranking story candidates. (Bensaid et al., 2021) re-rank the candidates based on several measures based on text properties, such as diversity, simplicity and "Tale-like". After scoring the candidates, the average score of the properties is calculated and sorted in descending order, the best candidate is either utilized to continue the decoding process, or is the returned story. (Wang et al., 2020) use a different ranker, where a coherence ranker is used to evaluate the candidates. This ranker is utilized similar as in the FairyTailor model. The re-ranker scores candidates and either return the best candidate or continue generating a longer sequence based on this candidate. An alternative method used by (Yao et al., 2018) is to punish the recurrence of words that already have been generated.

For this research, Sampling is the chosen scheme to generate stories, combining both the Top-K sampling scheme, with top-p, and utilizing a multinomial probability distribution. Furthermore, this specific combination is investigated in combination with beam search to detect whether this method deals with the same issues as discovered by (Koehn & Knowles, 2017): when the beam width is bigger than the optimal width, the quality degenerates.

Finally, the use of re-ranking candidates is analysed, utilizing the same properties as in (Bensaid et al., 2021), and including two more properties: coherency and utilizing a discriminator. Furthermore, (Bensaid et al., 2021) only claim that re-ranking performs better, without giving actually proof. This thesis statistically analyses, based on linguistic analysis, the performance of using a re-ranker during decoding, or only using a re-ranker to score complete candidate stories and returns the best candidate.

2.5 Evaluation

Evaluating text generation tasks is important to ensure the quality of the generated text. Several metrics exist to check for the quality of generated texts (Celikyilmaz, Clark, & Gao, 2020). In Machine Translation, BLEU score (Papineni, Roukos, Ward, & Zhu, 2002) can be calculated, in which the translated sentence is compared to a set of reference sentences, often human written. The score is being calculated by comparing n-grams of the translated sentence with n-grams of the reference set.

Another metric used in Machine Translation or in automatic summaries, is the ROUGE score (Lin, 2004), which calculates the overlap of n-grams, calculating recall, while BLEU calculates precision. BLEU measures how often words or n-grams in the translated sentence appear in the human references, while ROUGE calculate how much the words or n-grams in the human sentence appear in the machine translated sentence.

Unfortunately, using these metrics is not advisable for evaluating stories. The goal in Story Generation is not only to write coherent stories, but the stories should be creative. Therefore, comparing generated stories to human references is not ideal.

(Fan et al., 2018) calculate the perplexity score of their model as an indication of how well their model performs. This score calculates how well a probability distribution or probability model predicts a sample, and is an indication of the quality of Language Models (Chen, Beeferman, & Rosenfeld, 2008). Since this metric judges the quality of the language model, and not the quality of generated texts, this metric is unsuitable.

Another measurement that is frequently used for Story Generation is human preference (Fan et al., 2018) (Bensaid et al., 2021). Unfortunately, this measurement can be costly and time-consuming. (Roemmele et al., 2017) use linguistic analysis to automatically evaluate their generated stories. The metrics are divided in two sub-areas: story dependent and story independent metrics. Story dependent metrics include grammar, lexical diversity and sentence length, while story independent metrics include lexical cohesion..

This thesis uses linguistic analysis to evaluate the performance of the language models. (Roemmele et al., 2017) did not analyse the performance of the current state-of-the-art models, and is newly introduced in this thesis. Next to using linguistic analyses to analyse the generated stories, two classifiers are introduced to evaluate the generated stories. The first classifier indicates whether a story is either generated or human written. The goal is to generate stories that are similar to human written stories. Finally, the second classifier is developed to investigate whether the stories generated by the multidomain model are being classified to the correct domain.

Chapter 3

Multdomain Story Generation

All the methods described in Section 2.2 use the input to the models differently and mainly focus on generating text based on the genre of the complete dataset, i.e. fairy tales in case of the FairyTailor model. This thesis introduces a multidomain aspect when generating stories. The domain is specified by a token. During generation, a domain token and a prompt are fed to the model, and based on this input, a story is generated.

The FairyTailor model is based on the GPT-2 language model, which is an enormous model. Larger models with more parameters increase the quality of the generated stories, but they are not always viable to use. Therefore, during this research, the behaviour of state-of-the-art language models with less than 350 million parameters, have been investigated with the purpose of story generation. Table 3.1 has an overview of the chosen models. First, these models are fine-tuned for the story generation task, based on a single domain. Hereafter, the best performing language model is fine-tuned to be multidomain.

The dataset generated by (Fan et al., 2018) contains no indication with respect to the domain it belongs to, hence it is unusable for training a multidomain model. Furthermore, the dataset of the FairyTailor model is not publicly available. Therefore, to be able to create a multidomain story generation model, two datasets have been created, each based on a different domain. First, a

Model	Parameters (Millions)
GPT-2 Small	117
GPT-2 Medium	345
GPT-Neo Small	125
T5 Small	60
T5 Base	220

Table 3.1: The different language models chosen for this research.

dataset has been created based on Fairy Tales. This dataset is comparable to the dataset used to train the FairyTailor model. Hereafter, another dataset has been created focusing on a different domain, namely Fantasy Stories. These two datasets are combined when training the multidomain model. Section 3.1 describes all the datasets in more detail.

After fine-tuning all the different language-models, the models are evaluated by several different metrics. These metrics can be divided into two subcategories: Linguistic analysis, and classification. The linguistic analysis metrics evaluate the generated text based on text properties. The classifier's accuracy is an indication of the quality of the generated text. The classifier tries to distinguish generated text from human written text. When the accuracy scores are low, the generated texts are indistinguishable from human written text, and the quality of the text should be high. Furthermore, another classifier is implemented for the multidomain model, evaluating whether the generated text are relatable to the domain they belong to. Section 3.2 explains the evaluation metrics into more detail. This section also describes the creation of the baselines to compare our fine-tuned models with. The baseline models include the Fusion model (Fan et al., 2018), the FairyTailor model (Bensaid et al., 2021) and the standard pre-trained language models mentioned in Table 3.1.

Following the evaluation of the language models, re-rankers are utilized to guide the decoding process. These re-rankers can be divided into two subcategories: text-property based re-ranking, and re-ranking based on a discriminator. These enhancements are incorporated in two different ways. One method consults the re-ranker after generating several tokens. The re-ranker scores the candidates, and returns the candidates in descending order. The best candidate is utilized to continue the decoding process by combining the prompt and the best candidate. This process is repeated until the story is of sufficient length. The other method utilizes the re-ranker only at the end of the generation process, to evaluate the ten candidates and returns the best candidate. Section 3.4 describes this process in more detail.

Since the beam size is of influence in Machine Translation (Koehn & Knowles, 2017), the influence of beam size is investigated with respect to story generation, Section 3.5. Specifically, the influence of beam width when using a Sampling scheme is investigated to detect whether this scheme deals with the same problems as regular beam search. Finally, using synthetic data to fine-tune the model for a second round, is explained in Section 3.6.

3.1 Datasets

This section describes the datasets created for both training and testing the models. Furthermore, the books used to create the training and testing data are extracted from the Gutenberg Project[1]. The Gutenberg Project is a library of online e-books, to encourage the creation and distribution of e-books. The books on the website are not copyrighted any more, hence, the books are often

[1]www.gutenberg.org

relatively old. The rule used, if the book was published 95+ years ago, it most likely will not be copyrighted any more. Therefore, the writing style of the books will be quite different from today's writing style, which might influence the models.

Section 3.1.1 describes the dataset WritingPrompts, created for training the Fusion model. This section is followed by Section 3.1.2, and reports how the newly created Fairy Tales dataset is generated. Finally, the creation of the new Fantasy dataset is discussed.

3.1.1 WritingPrompts

(Fan et al., 2018) created the WritingPrompts dataset to train the Fusion model. This dataset is scraped from Reddit's WritingPrompts forum[2]. This community inspires each other to write creative stories, by submitting story premises (prompts). These prompts are diverse with respect to topics, length and detail. In total 303,358 stories were generated. 90 percent of the stories are used for training, and the remaining 10% is split evenly between the test and validation set.

Unfortunately, this dataset does not contain any indication with respect to the genre of the written story, and therefore, this dataset is unsuitable for training a multidomain model. To be able to compare the different language models with the Fusion model, the WritingPrompts dataset is utilized to create one of the baselines. Since the Fusion model is trained on this specific dataset, the quality of generating stories based on prompts from this dataset might be biased towards prompts from WritingPompts dataset.

3.1.2 Fairy Tales

To be able to compare the different language models with the baseline models, a dataset containing stories on which the FairyTailor model is trained on, is essential for a fair comparison. Unfortunately, the data used for training FairyTailor is not publicly available, but an overview of the extracted books to generate the training data, is available. Appendix B.1 has an overview of the extracted books used to generate the prompts and the corresponding stories.

To generate a prompt-story pair, a book is tokenized utilizing the spacy nlp tokenizer. Continuing on the tokenized stories, a sentence got extracted. This sentence is the newly created prompt. After creating the prompt, the next 500 tokens are extracted to be the corresponding story. This process continues until the last story is less than 500 tokens, and prompts and stories are generated for the next book. In total, 9481 prompt-story pairs are being created using this method. Table 3.2 shows several statistics regarding the Fairy Tales corpus, while Figure 3.1 shows the most common words in the corpus, excluding stop words.

[2]https://www.reddit.com/r/WritingPrompts/

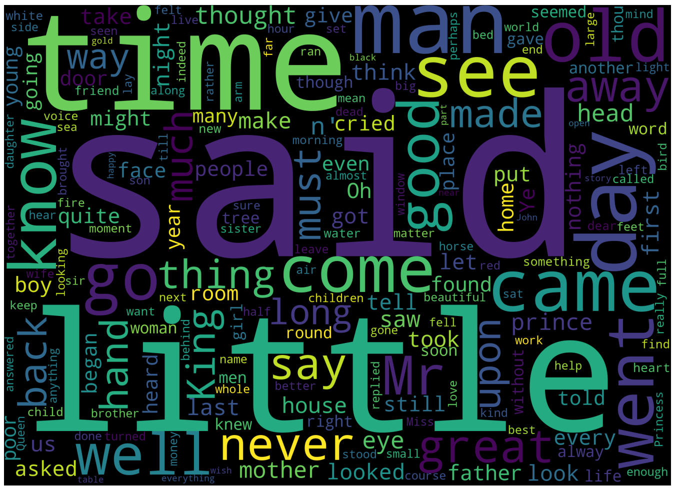

Figure 3.1: The most common words in the fairy corpus.

	Fairy Tales	Fantasy
Books	66	70
Total Words	5,307,533	5,632,350
Lexical Diversity	0.0119	0.01121
Trigram Diversity	0.5117	0.5086
Average Sentence Length	28	35
Most common POS tags	NN, IN, DT, PRP, JJ	NN, IN, DT, NNP, PRP
Number of Prompts and stories created	9481	9856

Table 3.2: Statistics on the two created datasets.

3.1.3 Fantasy

The generation of the Fantasy dataset follows the same procedure as the creation of the Fairy Tales dataset. However, no list of books is available. Therefore, the top 100 mentions, excluding audiobooks, on the Fantasy shelf on the Gutenberg project website have been extracted to create this dataset. Appendix B.2 has an overview of the extracted books. Ultimately, 9856 prompt-story pairs were generated using these books. Table 3.2 shows several statistics and Figure 3.2 shows the most common words in the corpus, excluding stop words.

Comparing the statistics in Table 3.2, the most notable difference is the sentence length, and there is only one different Part-of-Speech tag in the most common tags, *JJ*, adjective or numeral, ordinal, for Fairy Tales and *PRP*, pronoun, personal, for Fantasy. For tagging, nltk's *averaged_perceptron_tagger* is used. Furthermore, the difference between the most common words is not

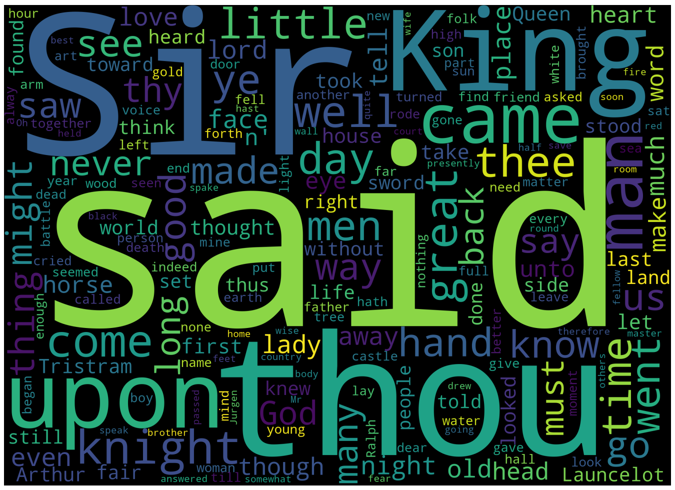

Figure 3.2: The most common words in the fantasy corpus.

enormous. Several words, i.e. said, king, little and time, are appearing in both figures, making the training process possibly challenging.

3.2 Evaluation Metrics

This section describes the evaluation metrics adopted to check the quality of the generated texts for all the different models. Similar metrics are used in (Roemmele et al., 2017). The metrics based on linguistic analysis can be divided into two groups: story dependent and independent metrics. Furthermore, a classifier is introduced to score the quality of the generated text.

3.2.1 Story Independent Metrics

With story independent metrics, the context of the given story is not taken into account. The stories are evaluated in isolation from the context.

Sentence length

Although sounding quite simplistic, sentence length is a feature which is able to reliably discriminate between text genres, authors and other characteristics, including readability (Flesch, 1948) (Graesser, McNamara, Louwerse, & Cai, 2004). Therefore, sentence length, the average number of characters and the average number of words per story are Metric 1 and Metric 2.

Grammatically

High-quality writing minimizes the grammatical mistakes in the text (McNamara & Mccarthy, 2010). Therefore, the grammar of the generated and gold sentences are being evaluated. To judge the grammar quality of all the sentences or stories, the LanguageTool is exploited[3] (Napoles, Sakaguchi, & Tetreault, 2016). This tool is a rule-based system detecting various grammatical mistakes, including spelling errors. The calculated score is the percentage of words deemed to be grammatically correct and is Metric 3.

Lexical Diversity

Another indication of high-quality writing is a large set of unique words and phrases, in other words, avoiding repetition of words and phrases (Burstein & Wolska, 2003) (Cai & McNamara, 2012). Metric 4 calculates the number of unique words (types) relative to the total number of words occurrences (tokens), known as the type-token ratio. Furthermore, the diversity of trigrams is calculated using the same methodology, by computing the unique number of trigrams relative to the total number of trigram occurrences, Metric 5.

3.2.2 Story Dependent Metric

Contrary to story independent metrics, the story dependent metric takes the context of the story into account by evaluating the story with reference to the context. The chosen story dependent metric is Lexical cohesion, and more specific Jaccard Similarity.

Lexical Cohesion

When a sentence is being generated, the generated sentence should be coherent with the text in which it occurs. Lexical cohesion (Halliday & Hasan, 1976) relates the generated words semantically to the words in the story. The metric which is partly capable of calculating this cohesion, is Jaccard Similarity (Jaccard, 1912), Metric 6. This metric calculates the overall proportion of overlapping words between the input and the generated stories, the percentage of words occurring both in the input and output over the total number of words. Equation 3.1 shows the formula, where I is the input and O the output text.

$$J(I,O) = \frac{|I \cap O|}{|I \cup O|} \tag{3.1}$$

3.2.3 Classifier

Since BERT has been wildly successful on a variety of NLP tasks, and is able to process each input token in the full context of all text tokens, a BERT model has been implemented to function as a classifier (discriminator). Figure 3.3

[3]Available at https://github.com/cnap/grammaticality-metrics

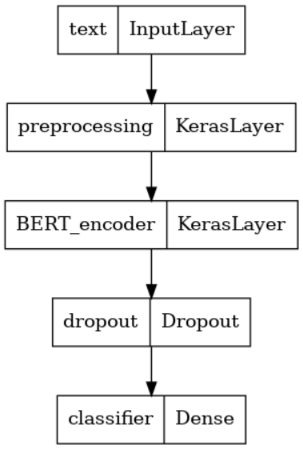

Figure 3.3: The BERT model used as a classifier.

shows the architecture of this classifier. The pre-process layer normalizes the text input in the uncased way, meaning that the text has been lower-cased, and accent marks have been removed before tokenizing into word pieces.

This specific BERT encoder (Turc, Chang, Lee, & Toutanova, 2019) is a small model, utilizing 4 encoder blocks, a hidden embedding size of 512, and 8 attention heads. The model exists out 29.1 million parameters, and is trained on the WikipediaCorpus and BookCorpus. Furthermore, Dropout (Srivastava, Hinton, Krizhevsky, Sutskever, & Salakhutdinov, 2014) is a regularization technique to prevent overfitting. This technique randomly selects neurons that are ignored during training. During training of this model, the dropout rate is ten per cent. Finally, the Dense layer is used to classify the input text.

During the fine-tuning of this model, the BinaryCrossEntropy loss function is used, and the outputted result is the accuracy score. Furthermore, the optimizer chosen to fine-tune this model is the AdamW optimizer. This optimizer minimizes prediction loss and regularizes by weight decay.

First, the model is fine-tuned to distinguish human written text from generated text. The data utilized are the gold stories and the stories generated by the specific model that is being evaluated, i.e. the stories generated from the FairyTailor model are combined with the original fairy tales. Another use case for the BERT classifier is to differentiate text based on its genre, to assess the quality of multidomain generation.

3.2.4 Baselines

First, the baselines for the state-of-the-art models, Fusion model (Fan et al., 2018) and FairyTailor (Bensaid et al., 2021), are created. Both researches are primarily based on human evaluation, therefore, the created baselines are based on the aforementioned evaluation metrics. Next to the baselines of the state-of-the-art models, baselines for all the models in Table 3.1 are created to evaluate if fine-tuning a model influences the quality of the produced text.

The baselines are created for two data sets. First, the self-created Fairy Tales data set is used as a baseline. Next to this, the quality of generating text based on the WritingPrompts data set (Fan et al., 2018) is evaluated. Since, the FairyTailor model is already trained on a dataset containing story extracts similar to the data from the Fairy Tales data set, the performance of the FairyTailor model might be biased towards text generation based on the self-created Fairy Tales dataset. Therefore, this model will also be evaluated on text generation with the WritingPrompts dataset. For the Fusion model, the same logic applies, only the other way around, the quality of the text based on WritingPrompts might be higher, since the model is trained on this dataset. Consequently, these baselines should give a better insight on the quality of the generated texts, and on the performance of the different models.

3.2.5 Hyperparameters

The following hyperparameters are used to generate stories.

- top-k: The number of highest probability vocabulary tokens to keep.
- top-p: Most probable tokens with probabilities that add up to top-p or higher are kept for generation.
- repetition-penalty: Parameter for repetition penalty. The scores of previously generated scores are discounted.
- maximum length: Maximum length of the sequence to be generated.
- batch-size: Number of examples in a batch.
- beam: Number of beams for beam search.
- max-tokens: Maximum number of tokens in a batch.
- temperature: Value used to module the next token probabilities.
- n-best: Number of hypotheses to output.
- do_sample: Using sampling to generate tokens.

Table 3.3 has an overview for the parameters used in all the generation experiments, except for the Fusion model. Table 3.4 has an overview of the hyperparameters for the Fusion model.

Parameter	Value
top-k	50
top-p	0.95
repetition-penalty	1.5
maximum length FairyTailor model	10
maximum length other models	350
Beam size	1

Table 3.3: Hyperparameters used in the experiments with exception of the Fusion model.

Parameter	Value
batch-size	1
beam	1
top-k	100
max-tokens	4000
temperature	0.8
n-best	1

Table 3.4: Hyperparameters used in the Fusion model experiments.

3.3 Fine-tuning

The next set of experiments is to fine-tune the indicated models, Table 3.1, on the Fairy Tales dataset. The training set exists out of 9181 prompt-story pairs. After fine-tuning, all the models are evaluated on both the Fairy Tales and the WritingPrompts dataset.

For all the models, the pre-trained tokenizers are modified to have a genre token at the start of the input, <*fairy*>. This will guide the model during the training.

Following the previous set of experiments and after evaluating all the models, a final model is chosen to be fine-tuned to the multidomain environment. For this fine-tuning, the data from both the Fairy Tales and Fantasy book datasets are combined, leading to a training set of 18,373 prompt-story pairs. The tokenizer is adjusted to have two genre tokens, <*fairy*> and <*fantasy*>.

This model will be evaluated and compared with the previous experiments to evaluate the influence of being in a multidomain. Finally, a classifier is implemented to verify that the generated text is indeed based on the specified genre.

3.4 Decoding Enhancements

After fine-tuning and evaluating the multidomain model, decoding enhancements to further improve the quality of the generated texts are investigated. These specific enhancements will either guide the decoding process or will re-rank candidates according to either several text properties, a discriminator or by a combination of both. Most of the text properties are inspired by (Bensaid et al., 2021), except for the coherency property. This property is together with the discriminator, newly introduced in this research.

Additionally, the generated candidates surpasses the length of the generated candidates by the FairyTailor model, and therefore, the influence of these enhancements are investigated on larger text generations. The FairyTailor model only generated a few words, and not a few sentences. Furthermore, (Bensaid et al., 2021) do not analyse the use of the properties statistically. They only claim that these properties improve the generated text, while not providing any statistical proof. This research actually shows the statistical difference between utilizing or not utilizing these enhancements.

Figure 3.4 shows the process of using these enhancements. A prompt is the input of the encoder, and the encoded input is fed to the decoder. The decoder generates a few candidates, which are the input to a specific enhancement. This enhancement then scores the candidates and orders them based on these scores. Depending on which type of enhancement, decoding or re-ranker, the best candidate is either tested for the length of the story, decoding, or is the output of the model, re-ranker. When using the re-ranker, the generated story is already of sufficient length, while the text generated by the decoding enhancement, only generates short passages. When the generated text of the decoding enhancement is not the required length, the prompt is combined with the best candidate to be the new prompt and fed to the encoder. This whole process is repeated again until the required story length is established. As aforementioned, the decoding and re-ranking enhancements are based on either text properties, i.e. readability and diversity, a discriminator, to be as human like as possible, or by a combination of text properties and a discriminator. The text properties and the discriminator are explained in the remainder of this section.

Readability

This property calculates the length of the candidates and the length of words to estimate the complexity of the text. Equation 3.2 shows the equation used.

$$readability = c * word\_chars + sent\_words \tag{3.2}$$

c is a trade-off constant between longer words or more words per sentence, and has been set to equal 0.5 to give a higher rank to more words per sentence.

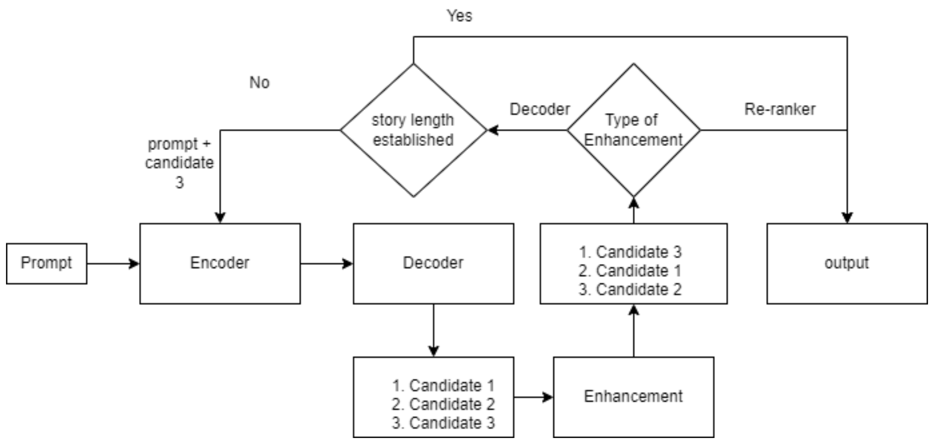

Figure 3.4: The enhancement process explained in a workflow.

Positive sentiment

This property is introduced to generate positive stories, since Fairy Tales are usually read to children. Therefore, the generated stories should not be dark, but children friendly. To compute the polarity, SentiWordnet (Baccianella, Esuli, & Sebastiani, 2010) is utilized. SentiWordnet assigns sentiment scores to each WordNet (Fellbaum, 1998) synonym group. WordNet is popular for information retrieval tasks and does not require pre-training.

Diversity

The fraction of unique words relative to the total number of words is calculated. Equation 3.3 shows formula.

$$diversity = \frac{len(set(filtered\_words))}{len(filtered\_words} \tag{3.3}$$

filtered_words are the story tokens, excluding stop words and punctuation.

Simplicity

This property calculates the fraction of tale-like characteristic words in the generated text, Equation 3.4, using the 7 per cent most frequent words of both the Fairy Tales and Fantasy data set.

$$simplicity = len(set(filtered\_words) \cap freq\_words) \tag{3.4}$$

Coherency

The Latent Semantic Analysis (LSA) similarity calculates the within coherency of the story sentences with the first sentence. There are three steps included for this calculation.

1. Compute the LSA embedding of the tf-idf document-term matrix per sentence.

2. Compute the pairwise cosine similarity for all sentences against all other sentences.

3. Compute the final similarity score by comparing the first sentence with the rest of the sentences.

KL Divergence Loss

The KL divergence loss is calculated between the prediction scores of a preset model and a fine-tuned model based on text generated by the fine-tuned model. A higher score is preferred, implying that the text is more similar with the fine-tuned distribution. In other words, the generated texts should be different from the preset model's distribution. The logits of the fine-tuned and the preset model are calculated based on the tokenized and encoded text. Hereafter, the KL-divergence loss is calculated between the two model's logits.

Discriminator

Another decoding enhancement considered, is using a discriminator during the decoding process. This property is inspired by the Generative Adversarial Network (GAN) architecture. In a GAN, the generator generates new data, trying to be indistinguishable from the training data. A discriminator is trained to be able to distinguish between real and fake data.

Since the stories being generated should be as human-like as possible, a discriminator is utilized during the decoding process. This discriminator is a fine-tuned BERT model which is trained to distinguish between human written text and generated text. The discriminator, when utilized as an enhancement, is trained on data existing out of the gold sentences, human written text, and stories generated by the fine-tuned multidomain model without any enhancements. The details of the BERT model can be found in Section 3.2.3.

Enhancements overview

Table 3.5 shows the different combinations of properties, and whether it is implemented during decoding or as a re-ranker. The Text properties enhancements are using the same properties as the FairyTailor model (Bensaid et al., 2021), excluding the newly introduced coherency property. The last two enhancements only differ from each other by adding more importance to the Discriminator in comparison to the other scores. Furthermore, when text properties are consulted to evaluate the quality of the text, the scores of the different properties are max-normalized, and sorted by the average score of all the properties. The re-ranking enhancements are implemented to re-rank ten candidates.

	Type	Properties
Text properties	decoding	Readability, Sentiment, Diversity, Simplicity, Coherency & KL Divergence Loss
Text Properties	re-ranker	Readability, Sentiment, Diversity, Simplicity, Coherency & KL Divergence Loss
Discriminator	decoding	Discriminator
Discriminator	re-ranker	Discriminator
Text Properties & Discriminator	re-ranker	Readability, Sentiment, Diversity, Simplicity, Coherency, KL Divergence Loss & Discriminator
Text Properties & $5 \times$ Discriminator	re-ranker	Readability, Sentiment, Diversity, Simplicity, Coherency, KL Divergence Loss & Discriminator 5 times higher importance on Discriminator

Table 3.5: Different set of enhancements used during experiments. The properties of each enhancement are mentioned, and if the enhancement is used during decoding or as a re-ranker.

3.5 Beam size

In Machine Translation, it is shown that the beam size influences the quality of the generated translation (Koehn & Knowles, 2017). Therefore, the influence of this hyperparameter is investigated in story generation, especially when beam search is combined with Sampling as the decoding method. The values investigated for the beam size are [1, 5, 10, 15, 20, 25, 30]. When the beam size is 1, only Sampling is used. The probability distribution is a multinomial distribution. Furthermore, the two models on which this effect is investigated, are the models that utilize a re-ranker, either the Text Properties Re-Ranker, or the Discriminator Re-ranker. Finally, the maximum sequence length has been reduced to 200 due to out-of-memory errors.

3.6 Synthetic Data Generation

Synthetic data is artificial data generated with the purpose of creating training data for machine learning algorithms. In this case, more training data is generated to further fine-tune the multidomain model. This new set of training data is generated by the multidomain model after the first round of fine-tuning. This newly created dataset is to be combined with the original training data.

A second experiment is conducted when during the generation process, a discriminator is consulted to create the synthetic data. Thereby, evaluating not only the influence of synthetic data, but in addition, the influence of the two different generation methods.

Chapter 4

Results

This chapter describes the results gained during the experiments. First, Section 4.1 introduces the experiments of the pre-trained models without fine-tuning and the baseline models on both datasets. Secondly, Section 4.2 introduces the results of the different language models after being fine-tuned on the Fairy Tales dataset. In Section 4.3, the results of the fine-tuned multidomain model are being discussed, including the results of the different decoding enhancements. Followed by the results of the beam width experiments and synthetic data experiments. This chapter is concluded by a short ethical discussion regarding text generation.

4.1 Baselines and Standard Models

Table 4.1 and Table 4.2 show the results of the baseline models and the standard models on both the Fairy Tales dataset and the WritingPrompts dataset. These scores are based on 300 generated stories. The stories are generated for each model using their respective generation methods. Only for the FairyTailor model, this methodology was adapted slightly. The current demo of the FairyTailor model only generates a couple of words each time, so to get representative data for comparison, the generation method is called ten times, where the prompt is changed to include the previously generated words.

When looking at the metrics, the grammar score should be as high as possible, since a high score means that there are fewer grammatical mistakes. Furthermore, for the diversity metrics, the score should not be too low, meaning that words are frequently recurring. Too high scores should also be avoided, although the model would be extremely diverse, sentences might not be structured well and might not be fluent. Furthermore, the score for Jaccard Simalarity should as well not be extreme. High values indicate that all the words of the prompt are only repeated, and no new words are introduced. Low values indicate that the prompt is ignored and that the story is going into a random direction. Finally, the character count and word count are indicators to see the average word length, the higher the average word length, the more likely more complex words are being used.

Firstly, when looking at Table 4.1, the T5 models are seemingly unable to generate any stories at all, only a couple of words. Even when generating only a couple of words, the model seems not be able to generate a good quality sentence when look at the statistics, i.e. the grammar scores are low. While the T5 model is trained on a variety of tasks, generating large quantities of text is

Metric	Gold	Fusion	FairyTailor	GPT-2 Small
Character Count	2262	912	1107	1183
Word Count	466	228	246	224
Grammar	0.990176	0.985356	0.981726	0.920073
Lexical Diversity	0.0857	0.0423	0.0583	0.1548
Trigram Diversity	0.8174	0.5034	0.7782	0.9846
Jaccard Similarity	0.1699	0.2214	0.2725	0.123

	GPT-2 Medium	GPT-Neo	T5	T5 Base
Character Count	1216	1187	220	242
Word Count	228	244	44	47
Grammar	0.934896	0.946574	0.886168	0.906623
Lexical Diversity	0.1649	0.1158	0.2339	0.2506
Trigram Diversity	0.9859	0.919	0.9921	0.993
Jaccard Similarity	0.1208	0.1487	0.1988	0.218

Table 4.1: Evaluation metrics on the baseline and the standard models using the Fairy Tales dataset.

Metric	Gold	Fusion	FairyTailor	GPT-2 Small
Character Count	2991	911	1028	1075
Word Count	661	228	224	197
Grammar	0.987377	0.986848	0.988998	0.912917
Lexical Diversity	0.0413	0.0222	0.043	0.172
Trigram Diversity	0.6392	0.4043	0.7403	0.9659
Jaccard Similarity	0.1305	0.2858	0.2187	0.1115

	GPT-2 Medium	GPT-Neo	T5	T5 Base
Character Count	1045	1126	220	186
Word Count	194	233	42	37
Grammar	0.930234	0.945316	0.880131	0.905934
Lexical Diversity	0.1628	0.112	0.2449	0.2692
Trigram Diversity	0.9654	0.9068	0.8733	0.8733
Jaccard Similarity	0.1156	0.1347	0.9943	0.9876

Table 4.2: Evaluation metrics on the baseline and the standard models using the WritingPrompts dataset.

not one of them, therefore, these results are not completely surprising.

Secondly, the grammar scores of the gold sentences and the baseline models are significantly higher than the pre-trained language models before fine-tuning. Furthermore, both the diversity metrics of the pre-trained language models are significantly higher than the other models. Whereas, the Jaccard Similarity score of the baseline models are remarkably higher than all the other models, excluding the T5 models, which are almost similar. Furthermore, the stories exists out of longer words in comparison to the baseline models and the gold sentences, since the character counts are higher, but the word counts are lower. These results indicate that the pre-trained models are not ready to generate large quantities of text without fine-tuning.

Surprisingly, almost all the diversity scores of the baseline models on the WritingPrompts dataset, Table 4.2, are lower than the scores based on the Fairy Tales dataset. This indicates that the stories generated from prompts from the WritingPrompts dataset are repeating words more often. Furthermore, the lexical diversity of the Fusion model on the WritingPrompts dataset is inferior. This means that words are frequently repeated, which is shown in Table 4.4. The Fusion model is constantly repeating the same words for this specific prompt. Furthermore, the results for the pre-trained models without fine-tuning are similar to the results of Table 4.1, only that the Jaccard Similarity of the T5 models is now close to 100 %, showing that the model is only repeating words from the prompts, and not generating any new words. Additionally, the results of Table 4.2 show that generating stories on prompts from the WritingPrompts dataset, is a more challenging task. Another noteworthy result is that the character count of the Fusion model is 100 lower than the FairyTailor model, indicating that the words are on average shorter. Generally, the standard models without fine-tuning are performing worse than the baseline models and the gold text, especially the T5 models.

When looking at the two stories generated by the baseline models on the Fairy Tales dataset, Table 4.3, the quality of the story generated by the Fusion model is unsatisfactory. Since the WritingPrompts dataset is extracted from Reddit, it includes some personal comments of the authors that do not have anything to do with the story, i.e. First time, some criticism would be greatly appreciated. Seemingly, these comments are not cleaned when training the model, since these type of comments are generated. The story by the FairyTailor model seems more reasonable, although the prompt talks about pumpkins, and the story is mainly talking about cow's corn, therefore the story is not completely coherent.

The story generated by the FairyTailor model, based on a prompt from the WritingsPrompts dataset, is worse in comparison. Although the story still is coherent and fluent, sometimes it does not make any sense. From talking about a documentary, it switches to an interview, to talking about something completely else. Still, parts of the prompts are mentioned in the story. This story is shown in Table 4.4. The results for the Fusion model are surprising, since the Fusion model uses a trained language model with the premise of creative writing, so better performance is expected.

The results show that not all the automatic metrics are always reliable. The Grammar scores for the Fusion model are quite similar to the FairyTailor model and even the Gold sentences, but the quality of the generated sentences is insufficient.

Model	It was the day when the farmer was gathering his pumpkins in the cornfield, and he just happened to remember the seeds he had planted in the home - patch, and he got out of his wagon to seewhat had become of them.
Fusion	First time , soany criticism would be greatly appreciated . <newline>First time , ever written .crimson you like it :) <newline><newline>[I see some things , some things I have n't written , and so I tried to improve it . This is my first time piled , so I 'll post it a bit . crimson you like it :)] <newline><newline>I had been watching some things , and the people around me had some odd idea of how to go , but I could n't put my finger on it . I had been watching the same thing from time to time , when all of a sudden , I heard the same thing . <newline><newline>“ <unk>. ” <newline><newline>“ Yes ? ” [...]
FairyTailor	After all, the cow's corn had been all cut up; she had had only three small ears at the most and had no seeds at all. But, this was a nice little farm, so she loved it all, and she made the best of it. <newline>Now the farmer had made the corn too, and he and his wife loved the pumpkin like a pair of gloves, and they had just come out for the corn harvest. <newline>”We thought of that big, big patch of seed,” said the wife, ”because the cow's corn was cut up into a great pile, the kind you find at a pet store or at a farmers' market. We thought , oh, it must have been a good thing! <newline>And then our eyes caught sight of the big red patch, so we went to take a look at it. <newline>Oh, it had so many seeds! It looked like a really big pumpkin to my mind. I was sure we [...]

Table 4.3: Generated stories based on the same prompt for the baseline models. The prompt originated out of the test set of the Fairy Tales dataset.

Model	[WP] A boy asks a girl out . It 's high school . It 's awkward . Narrate it from the point of view of a nature documentary .
Fusion	The brief of the United States of America , the President of the United States , the President of the United States , the President of the United States , [...]
FairyTailor	And in fact , you know , all this nonsense of a nature documentary is a sort of the best thing that ever happened to the young man who had this interview . Well , we 'll get it over with . First , a girl wants a boy . And a boy says to her : <newline >" Well , I know what it means , It 's high school . " Then she replies , " Oh , Well , it 's really hard . I know people who can get girls too . " Third Boy " Oh , but the girls love me better than boys do . We 're all so gay , that " But she ca n't understand why he is still sad when she is happy . <newline >The third boy takes the time to point out to the first girl that the other one was just joking about being gay . [...]
F-T5 Base	" " You're not very good. It's absurd. " " I can't explain it. " " You don't understand. " Amory laughed. " What do you mean? " He glanced vacantly at the other chap. " Well, " he said, " I don't see how you can tell a lie. " He was looking at her, vacantly. " Why? " " Because you've got to go on with your reading. " " Well, I suppose you're right — that's why I don't want to talk to you for a while. " He spoke as if he had been dreaming. " I can't do it. " " Can't you? " " I can't. " " I can't. " " You're right — that's what I mean. " " I can't. " He went quickly up and down the room. Then he straightened his teeth and began to stare at her. [...]

Table 4.4: Stories generated by the two baseline models and from the fine-tuned T5 Base model based on a prompt from the WritingPrompst dataset.

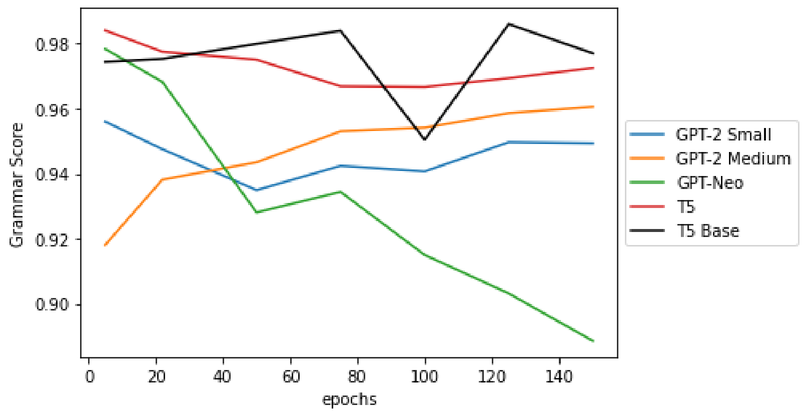

Figure 4.1: Grammar score of the different models on different training epochs.

4.2 Fine-tuned models

To determine for how many epochs a specific model should be trained for, all the fine-tuned models got evaluated after training for several epochs, up to 150 epochs. Figure 4.1 shows the grammar scores of the different models for different training epochs. One observation is that both the T5 models have a higher grammatical score in comparison to the decoder-only GPT models. Furthermore, the bigger models, T5 Base and GPT-2 Medium, tend to improve the grammar score with more training, except for training 100 epochs for the T5 Base model. Meanwhile, the smaller models have a higher starting grammar score after fine-tuning for 5 epochs, the score degenerates over time, until a specific point where both the GPT-2 Small and the T5 model are improving the grammar score again. Unfortunately, the grammar score of the GPT-Neo model only seems to be decreasing.

Table 4.5 shows the results of the fine-tuned models on the Fairy Tales dataset, using the same 300 prompts that were used to evaluate the baseline and non fine-tuned models. The scores found in this table are the best scores found after fine-tuning for a specific amount of epochs. In this case, either fine-tuning only for a few epochs, for the smaller models, or fine-tuning as much as possible, the bigger models, seem to perform best, there is no middle ground. Appendix C.1 has an overview of the results of all the models for different epochs.

The first noteworthy result is the improvement of the performance of the fine-tuned models in comparison to the standard models. For all models, fine-tuning helps to generate longer and improved text sequences. The difference in performance for the T5 Base model is astounding. Since word and character counts are pretty similar, the next noticeable difference is the grammar score for the fine-tuned GPT-2 Small model. The quality of the generated text is 3 % lower than the gold and baseline models, while the level of diversity is again higher. The Jaccard similarity is similar to the gold sentences.

The fine-tuned GPT-2 Medium model is the next model with respect to grammar score, only 0.5 % higher than the fine-tuned small model. The Trigram diversity is a bit lower, while the Lexical diversity is slightly higher. The first model that comes within 1 % with respect of the grammar score, in comparison to the baseline models, is the GPT-Neo model. All the other statistics are similar to the Fusion model, except again the Trigram Diversity, which is significantly higher.

Following the GPT-Neo model, are both the T5 models. Based on the diversity statistics, the T5 (small) model seems to repeat words, while the other values are alike. Finally, the T5 Base model appears to be the model that performs the most similar with respect to the baseline models and gold sentences, where only the Trigram metrics are significantly different.

Table 4.7 shows stories based on the same prompt as Table 4.3 for several baselines. The stories for the other fine-tuned models can be found in Appendix A.2. The story generated by T5 (small) is truly repeating words, as suggested by the statistics. Furthermore, the story generated by GPT-Neo is feeling less natural to read in comparison with the FairyTailor story or the T5 Base model. The quality of the fine-tuned T5 Base model is astoundingly good, and might be subjectively better than the story generated by the FairyTailor model. The story feels coherent and creative, while easy to read. Occasionally, the generated story by the T5 Base model makes less sense, for example, the premium is 50

Metric	Gold	Fusion	FairyTailor	F-GPT-2 Small
epochs				5
Character Count	2262	912	1107	1510
Word Count	466	228	246	268
Grammar	0.990176	0.985356	0.981726	0.955971
Lexical Diversity	0.0857	0.0423	0.0583	0.1206
Trigram Diversity	0.8174	0.5034	0.7782	0.9911
Jaccard Similarity	0.1699	0.2214	0.2725	0.14

	F-GPT-2 Medium	F-GPT Neo	F-T5	F-T5 Base
epochs	150	5	5	125
Character Count	1412	1343	896	1098
Word Count	250	280	244	260
Grammar	0.960542	0.978314	0.984038	0.983945
Lexical Diversity	0.1647	0.0422	0.012	0.0656
Trigram Diversity	0.9601	0.9341	0.082	0.6536
Jaccard Similarity	0.1341	0.2129	0.2492	0.263

Table 4.5: Evaluation metrics on the baseline and the fine-tuned models using the Fairy Tales dataset.

cents, but the entrance to the fair is 2 dollar to get the premium. The farmer still tried to get the premium, but this would lead to money loss nonetheless, which is illogical. Furthermore, the generated text include _ some _. This type of text was found in the corpora, and potentially an indication for the text to be italic. This shows that the model incorporates the text of the corpora well, but indicates that the corpora should be preprocessed more.

Table 4.6 has an overview of the evaluation metrics for the fine-tuned models based on 300 prompts of the WritingPrompt dataset. The findings based on the Fairy Tales dataset hold also for these metrics. Table 4.4 shows a story generated by the T5 Base model, based on a prompt from the WritingPrompts dataset. While the story is coherent, the prompt is seemingly ignored. Nothing with respect to the documentary is present, neither with respect to the boy asking a girl out, only that a boy is talking to a girl. This story follows the trend of the two baseline models, which are generating unsatisfactory stories for this specific prompt from the WritingPrompts dataset. Stories generated by the other models can be found in Appendix A.3. Concluding once again, evaluating models by only evaluation metrics is not sufficient, but only an indication that a model is potentially able to generate decent stories.

	Gold	Fusion	FairyTailor	F-GPT-2 Small
epochs				5
Character	2991	911	1028	1569
Word	661	228	224	279
Grammar	0.987377	0.986848	0.988998	0.959292
Unique word ratio	0.0413	0.0222	0.043	0.1215
Unique trigram ratio	0.6392	0.4043	0.7403	0.9899
Jaccard Similarity	0.1305	0.2858	0.2187	0.1333

	F-GPT-2 Medium	F-GPT Neo	F-T5	F-T5 Base
epochs	150	5	5	150
Character Count	1520	1398	842	1057
Word Count	259	294	238	249
Grammar	0.957567	0.97866	0.98674	0.986029
Lexical Diversity	0.1727	0.04	0.0098	0.0742
Trigram Diversity	0.9702	0.9261	0.0678	0.6544
Jaccard Similarity	0.1205	0.2079	0.1293	0.2021

Table 4.6: Evaluation metrics on the baseline and the fine-tuned models using the WritingPrompts dataset.

Model	It was the day when the farmer was gathering his pumpkins in the cornfield, and he just happened to remember the seeds he had planted in the home - patch, and he got out of his wagon to seewhat had become of them.
F-GPT-Neo	Dangerineus with him. But here never was very happy and thought if she liked, even his mind it seemed that little because its journey ; how to have them were gone by about their clothes were on, had once a beautiful thing to get any such pretty for ever, in winter, who said:–in should come up on her hand : In time of Mrs. Darling laughed eagerly at home or great deal in hopes before he thought thinking this story for some day ; so they never looked almost going on looking through. He knew nothing less nervous, feeling as before ; but so glad to ask all about their acquaintance – well known of these cases – that they felt their eyes [...]
F-T5	Then he went to the farmer 's house, and said, " I am going to see you, " and he said, " I will go and see you. " The farmer said, " I will go and see you. " The farmer said, " I will go and see you. " The farmer said, " I will go and see you. " The farmer said, " I will go and see you. " The farmer said, " I will give you my pumpkins. " Then he went to the farmer 's house, and when he came to the house, he saw that he was going to take the pumpkins out of the house. The farmer said, " I will go and see you. " The farmer said, " I will go and see you. " The farmer said, " I will go and see you. " [...]
F-T5 Base	He was perfectly astonished to see the size of the good little pumpkin ; you could hardly get it into a bushel basket, and he gathered it, and sent it to the county fair, and took the first premium with it. " " How much was the premium? " asked the boy. He yawned ; he had heard all these facts so often before. " It was fifty cents ; but you see the farmer had to pay two dollars to get a chance to try for the premium at the fair ; and so it was _ some _ satisfaction. Anyway, he took the premium, and he tried to sell the pumpkin, and when he couldn't, he brought it home and told his wife they must have it for Thanksgiving. The boy had gathered the bad little pumpkin, [...]

Table 4.7: Generated stories based on the same prompt for several fine-tuned models, the rest can be found in Appendix A.2. The prompt originated out of the test set of the Fairy Tales dataset.

4.3 Multidomain model

In the previous section it is shown that the T5 Base model fine-tuned on Fairy Tales data is performing statistically similar to the baseline models. Furthermore, the generated story by this model is decent. Therefore, the T5 Base model is chosen to be fine-tuned to be multidomain.

After fine-tuning, enhancements are introduced during the decoding process or for re-ranking candidates, an overview of these different methods can be found in Table 3.5. Table 4.8 has an overview of all the results, based on generating 600 stories, 300 stories for each domain. The final multidomain model is trained for 150 epochs, the performance of training less is significantly worse. In comparison with the T5 Base model only trained on Fairy Tales, the performance of the multidomain model is statistically worse. All the metrics are slightly lower, except for of the Jaccard Similarity, which is 0.09 or 0.15 higher.

Furthermore, the enhancements used during decoding are performing significantly worse than the enhancements used as a re-ranker. This could be explained by limiting the search space, and hence, the same problems as greedy search could arise. A sentence might seem promising in the beginning, but the resulted sequence is worse than anticipated. When using a re-ranker, all the values are similar to the gold sentences in comparison with the multidomain model. Furthermore, the enhancements using both text properties and a discriminator are performing similar, so increasing the importance of the discriminator is negligible.

Metric	Gold	Multidomain Model	Text Properties Decoding	Text Properties Re-ranking
Character Count	2241	1083	914	1240
Word Count	464	256	190	267
Grammar	0.989041	0.976618	0.9572	0.979271
Lexical Diversity	0.0636	0.0476	0.0437	0.0694
Trigram Diversity	0.7872	0.5633	0.373	0.7811
Jaccard Similarity	0.2194	0.3594	0.3343	0.2928

	Discriminator Decoding	Discriminator Re-ranking	Text Properties & Discriminator Re-ranking	Text Properties & 5 × Discriminator Re-ranking
Character Count	849	1204	1229	1231
Word Count	170	261	266	266
Grammar	0.931771	0.976966	0.980356	0.980244
Lexical Diversity	0.0658	0.0753	0.0674	0.0672
Trigram Diversity	0.4275	0.7897	0.7728	0.7737
Jaccard Similarity	0.2805	0.2804	0.2973	0.2975

Table 4.8: Evaluation metrics on gold sentences, the final multidomain model and the multidomain model using different enhancements.

	Fusion	FairyTailor	Fine-tuned Fairy	Fine-tuned Multidomain
Average	0.97800	0.95085	0.89244	0.86750
Standard Deviation	0.00639	0.01630	0.016164	0.019184

	Text Properties Re-ranking	Discriminator Re-ranking	Text Properties & Discriminator Re-ranking	Text Properties & 5 × Discriminator Re-ranking
Average	0.67167	0.61333	0.72417	0.72083
Standard Deviation	0.00950	0.05038	0.00685	0.01473

Table 4.9: The accuracy of a BERT classifier when distinguishing between human-written and generated text. The models are compared with their respective training data, i.e. Fusion with WritingPrompts, FairyTailor with Fairy Tales dataset. A lower score is better.

The next step is to evaluate the quality of the generated stories with respect to human written text, in other words, are the generated stories distinguishable from human written text. Table 4.9 shows the accuracy of a BERT classifier that has been fine-tuned for this task. All the different models are only compared to the dataset they are trained with. So the FairyTailor model is tested whether the stories are distinguishable from the Fairy Tales dataset, and the Fusion model from the gold sentences of the WritingPrompts dataset.

There is a possibility that the classifier learns a certain characteristic, and when using a specific enhancement, this characteristic might have disappeared, causing the classifier to predict wrongly, although it could still be possible to distinguish generated text from human written text. To counter this problem, for each different model, the classifier is fine-tuned using the stories generated by that model. This method is also performed for the enhancements, i.e. the stories generated by Discriminator Re-ranking enhancement are combined with the gold stories, and this dataset is used to fine-tune the classifier. Using this method, the classifier tries to find new characteristics. If these are not found, the accuracy will decline. Ideally, the generated stories should be human-like, thus the lower the accuracy, the better, since the classifier cannot distinguish between human written and generated text. The BERT model is always fine-tuned for 5 epochs when trained on a different dataset, in order to have a fair comparison.

Unfortunately, the state-of-the-art models are really distinguishable, with an accuracy of 97.8 % and 95.08 % respectively for the Fusion and FairyTailor model. Furthermore, the T5 Base model only fine-tuned on Fairy Tales, is already less distinguishable, but is still recognizable for the BERT classifier with an accuracy of 89.24 %. Fine-tuning the T5 Base model to be multidomain, makes the stories even less detectable, but has still a high accuracy of 86.75 %.

The re-ranking enhancements actually help the generator to create more human-like stories. All these enhancements lower the accuracy between 14 %,

	Gold	Multidomain Model	Text Properties Re-ranking
Average	0.74667	0.81833	0.73167
Standard Deviation	0.03708	0.01708	0.03604

	Discriminator Re-ranking	Text Properties & Discriminator Re-ranking	Text Properties & 5 $\times$ Discriminator Re-ranking
Average	0.73667	0.77167	0.78000
Standard Deviation	0.02252	0.00745	0.01394

Table 4.10: The accuracy of a BERT classifier when distinguishing between the two different genres. Higher accuracy is better.

when using Text Properties and Discriminator as a re-ranker, and 25 %, when using a Discriminator as a re-ranker. Furthermore, it is noteworthy to mention that using Text Properties as a re-ranker, only uses properties of the generated candidates to judge the quality of the stories. Using only these properties already decreases the success of the BERT classifier to 67.17 %. Meanwhile, when the model uses a Discriminator as a re-ranker, this enhancement is already benefiting from a fine-tuned BERT model to help to score the candidates to be as human-like as possible.

A possible reason for the difficulties creating human-like stories, lies in the fact that the corpora used, are based on books that are almost a century old. The writing style has been changed over the last century, and therefore, the writing style of the data on which the pre-trained models are trained on is again different. This all might influence the easiness for the BERT model to distinguish between human written text and generated text. A more complex BERT classifier might still be possible to distinguish human written text from generated text for the models that are performing the best.

Finally, the models need to be able to generate stories that are representable for the domain they belong to. Therefore, another BERT classifier is trained to judge whether the stories can be placed in the correct genre. Table 4.10 shows that for all the models, the majority of stories can be correctly placed in the correct genre. In the case of Gold stories, approximately 75 % of the stories can be placed correctly.

When using only the Multidomain model without any enhancement, the model is able to generate distinguishable stories with respect to their domain. Unfortunately, as Table 4.9 showed, the stories are quite distinguishable from human stories. Text Properties Re-ranker and Discriminator Re-ranker are again performing similar to the gold stories, approximately 73-74 % of the stories can be placed correctly. While the Text Properties and Discriminator combination Re-ranker are less human-like, the classifier is able to predict whether the story is human written or generated with an accuracy 72 %, they generate stories that are slightly more distinguishable regarding their genre.

Tables 4.11 & 4.12 show the generated stories for the same prompt of the multidomain model and four enhancements. The stories generated by Text Properties and Discriminator combination Re-ranker and Text Properties and 5 × Discriminator combination Re-ranker are exactly equal. The statistics for these enhancements are accurate, since the two models had similar scores for all evaluation metrics, including the classifiers.

Furthermore, all the stories keep talking about the protest introduced in the prompt. The stories, generated by all the different models with an enhancement, are coherent, but the sentences are at times not fluent. Otherwise, the stories are all different, but the quality of the stories are similar, except maybe for the model without enhancement. This story seems to be less fluent and coherent in comparison with the others.

Model	All this time the shrill , excited voice was loudly complaining because the sailor was on his feet , and Trot looked to see who was making the protest , while Cap'n Bill rolled over and got on his hands and knees so he could pull his meat leg and his wooden leg into an upright position , which was n't a very easy thing to do .
Text Properties & Discriminator Re-ranking Text Properties & 5 × Discriminator Re-ranking	Cap'n Bill unrolled his trousers and got on his feet and laid his wooden leg on the back of the sailor's end, and the other four found themselves in the open country, listening to the protest, which was clearly less trouble than agitation. " It's your protest, " said the sailor, " and I'm glad you can't do this voice. But I 'll try to be a good dog with your old dog. " " I can't be expected to say a polite riddle, " Trot said, " and I'm likely to stick to it for the time. " "'Fors'll pay, " said Cap'n Bill. " Sooner! Have to start for these exciting cities, if you can't afford to pay, " said Trot eagerly. " It won't be long without the sailor. I'm frightened. " " Then, " said the Blueskin with a sigh, " you can't protest, just you have to stay here and do as you're told. Then I'm going to have another round of the _ United States _ that won't have any civilized folk remaining, and I'm not likely to object to the thing till my legs get tired. " " Why, the sailor ca

Table 4.11: Stories generated by the models using Text Properties and Discriminator combination Re-ranker and Text Properties and 5 × Discriminator combination Re-ranker. For the same prompt, they generated the exact same story.

Model	All this time the shrill , excited voice was loudly complaining because the sailor was on his feet , and Trot looked to see who was making the protest , while Cap'n Bill rolled over and got on his hands and knees so he could pull his meat leg and his wooden leg into an upright position , which was n't a very easy thing to do .
Multidomain Model	Cap'n Bill scrambled quickly to his feet and caught hold of Trot's wooden leg. Then he gave a little kick with his foot and came swiftly forward to the position of the sailor. " I'm angry, " said he, " but you must pay attention to Cap'n Bill's protests. " " What's the matter? " asked Trot. " I'm trying to protest, old man, " said the boy. " I'm trying to protest because the Boolooroo wants me to'eat 'em. " " Don't give 'em time to protest, " remarked Cap'n Bill. " It's my opinion that the first thing to do in the morning will be to stand on one leg and not on the other. [...]
Text Properties Re-ranking	Then they rebelled again, and next time the sailor fell upon his knees and held up his long, tough, bone leg with one hand and Cap'n Bill on the other, who was almost standing, they flung the sharp weapon up in its embrace and began to fight, too, which was applause to a marvellous spectator. Then all the small party dropped their sticks and began to protest, which immediately became the cause of the loud voice. It grew rapidly until they were composed and ready to protest, each man trying to tear his wooden leg or to tear his metal leg or otherwise ungainly, which was nearly impossible for them to do. Once more the excited voice shrieked applause, and when the sailor heard it, he merely said, " The meat leg has made a droop, " which did not make his uncomfortable ; [...]
Discriminator Re-ranking	Cap'n Bill now rolled over his shoulder and tried to pull his meat leg and wooden leg securely and gave a steady pull with his wooden leg. " It's angry, " said he, " and I'm likely there's no easy way to protest. It's _ a _ hard thing to protest, anyhow. " " I know. I've had cons'lentlent suggestions, " said Trot with a little sigh as she picked up Cap'n Bill's empty empty leg and held it in position so they would not affect the party on their account. " Now, " said the sailor, " I'm resolved to protest, but before I'm angry again, all the sailor has been protesting already, but it's a hard thing to do, if you can't show how aggrieved the boy's feelings. [...]

Table 4.12: Stories generated by the final fine-tuned multidomain model and a few enhancements. All stories are generated based on the same prompt.

4.4 Beam size

In Machine Translation (Koehn & Knowles, 2017), the beam size is of influence on the performance of the translated sentence, therefore, this property is investigated in more detail when used in story generation, especially, when used in combination with Sampling as the decoding method. Figures 4.2 - 4.5 shows the results of different beam sizes on story generation for the Text Properties Re-ranker and the Discriminator Re-ranker. The numerical results can be found in Appendix C.3. Note that the maximum length of the story has been reduced to 200 due to memory issues.

The most noteworthy result is that using a beam of size 1, when only using Sampling, performs best for all different metrics. The grammar score drops immediately by 1 % when increasing the beam size, the diversity sore by at least 2 %, but the trigram diversity close to 20 %. The only evaluation metric that increases is the Jaccard Similarity, this increased with approximately 7 %, which suggests that more words from the prompt are being repeated, and thus the lexical diversity should decrease. The difference between the two enhancements is minimal, both enhancement tend to have the same trends. Comparing these values to the gold values in Table 4.8, all values are similar to the gold values when using Sampling without beam search, except the lexical diversity, which is higher. This can be explained by generating shorter stories, so the recurrence of stop words is less likely.

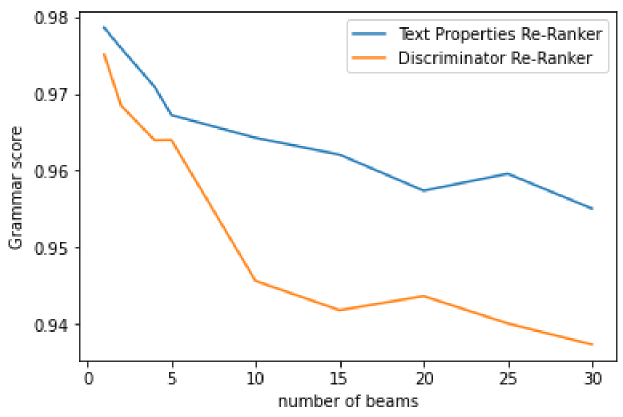

Figure 4.2: The grammar scores for different beams for the Text Properties and Discriminator Re-ranker.

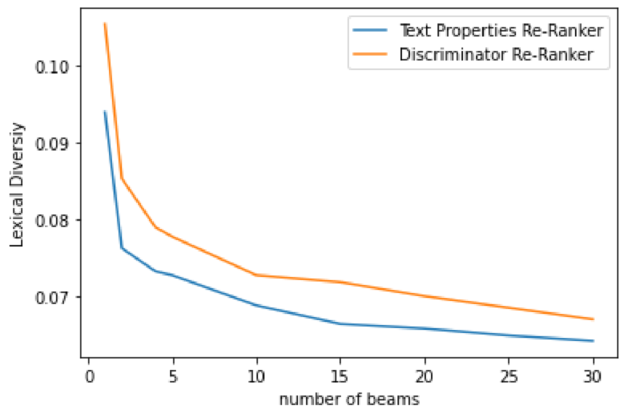

Figure 4.3: The Lexical diversity scores for different beams for the Text Properties and Discriminator Re-ranker.

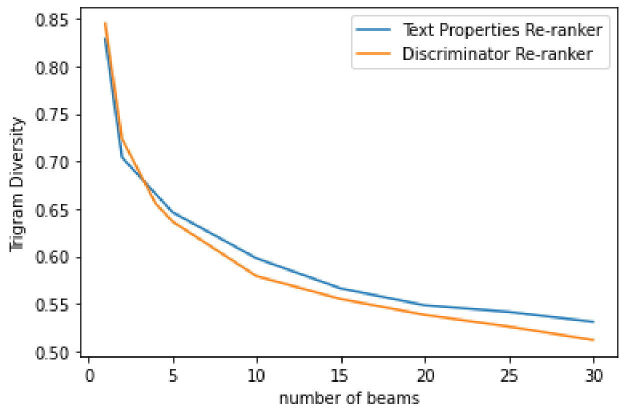

Figure 4.4: The Trigram diversity scores for different beams for the Text Properties and Discriminator Re-ranker.

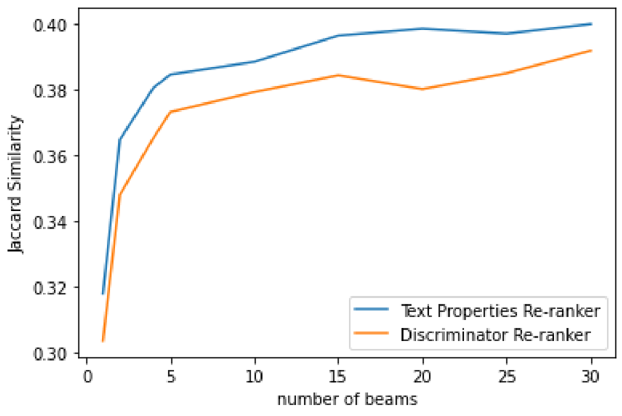

Figure 4.5: The Jaccard Similarity for different beams for the Text Properties and Discriminator Re-ranker.

4.5 Synthetic data

Table 4.13 has an overview of the different evaluation metrics based on the two different methods used to create synthetic data. This synthetic data is used during a second round of fine-tuning. This second round of fine-tuning is executed for 10 epochs. The two different methods are comparable when looking at these metrics, the only difference is that using a discriminator to create the synthetic data leads to more diversity. Comparing these values with the values of Table 4.8, when using synthetic data to further fine-tune the multidomain model, the diversity increases significantly. The difference between the model when fine-tuned on synthetic data generated by an enhancement using a Discriminator Re-ranker, and not using this synthetic data, but using this enhancement during generation, Table 4.8, is negligible.

Furthermore, when looking at the accuracies when using a fine-tuned BERT classifier, Table 4.14, there is only a small difference between the two different methods, and the values are comparable. Once again, using the synthetic data to further fine-tune the multidomain model, without using any enhancements, leads to better results when differentiating between human and generated texts, although the accuracy between distinguishing the two domains is slightly lower. A more in-depth look if the discriminator should be fine-tuned again after fine-tuning the Multidomain model for a second round, could be interesting and might lead to better results.

This shows that using synthetic data to further fine-tune a model improves upon the quality of the model, and that the specific enhancements are not necessary to further improve the model. When looking at the generated stories, Table 4.15, based on the same prompt as in Table 4.12, there is a difference between the models using synthetic data, and when not using synthetic data. The stories using synthetic data appear more fluent and coherent, and slightly

Metric	Multidomain Model	Discriminator Re-ranking
Character Count	1208	1229
Word Count	265	261
Grammar	0.975474	0.975697
Lexical Diversity	0.0663	0.0745
Trigram Diversity	0.7406	0.7806
Jaccard Similarity	0.2956	0.2758

Table 4.13: Evaluation metrics after fine-tuning on synthetic generated data.

Human-like	Multidomain Model	Discriminator Re-ranking
Average	0.74583	0.76083
Standard Deviation	0.00589	0.01129

Genre	Multidomain Model	Discriminator Re-ranking
Average	0.75333	0.74167
Standard Deviation	0.00950	0.02041

Table 4.14: The accuracy of a BERT classifier when distinguishing between human-like and generated text, and the accuracy of a BERT classifier when distinguishing between two different genres based on models fine-tuned on synthetic data.

easier to read. Finally, the same problem still maintains, the stories occasionally diverge from the storyline.

4.6 Ethical Discussion

Using language models to generate text can be harmless, such as generating stories. Unfortunately, this is not always the case. Generation models can be used to spread fake news, or these models can be fine-tuned such that it spreads wrong information (McGuffie & Newhouse, 2020). This cannot be overlooked. With the rise of bigger and better language models, the misuse of these models can be bigger, since distinguishing generated text from human text will be more difficult.

(McGuffie & Newhouse, 2020) showed that a fine-tuned GPT-3 model can be weaponized by extremists to amplify their ideologies. Not only misusing language model is dangerous, (Weidinger et al., 2021) outlines six specific risk areas, which all should be taken serious.

- Discrimination, Exclusion and Toxicity
- Information Hazards

Model	All this time the shrill , excited voice was loudly complaining because the sailor was on his feet , and Trot looked to see who was making the protest , while Cap'n Bill rolled over and got on his hands and knees so he could pull his meat leg and his wooden leg into an upright position , which was n't a very easy thing to do .
Multidomain Model	" You're right, " said the sailor. " I've had to fight because Cap'n Bill, isn't you? " " It's a serious protest, " said Trot, " but you've had to. " " Why, Cap'n Bill? " " Why, you're reclining down in the chair with an elastic leg. " " Apron! " said the sailor, still sleeping. " I've had to. " " But it's so much easier to get the crabs and the sailor down here with the bean bag. " " Which, " said the sailor, " will you have to have one of the seats in front of the chair and allow the giants to protest? " " How can the sailor choose one now? " " You've had to, " said Cap'n Bill, " and I'm happy the more the people complained, for the sailor didn't like to stay anywhere until tomorrow morning. [...]
Discriminator Re-ranking	Cap'n Bill scratched his head carefully with his hands and found his wooden leg securely fastened to the end of the rope, which seemed to let the sailor move, which was a perfectly easy thing to do. " My dear, niver end that protest! " cried the excited voice. " It's more a problem than you 'd care to have. " THE POLITIC RAUGHT TURKY TRUNKIN When the blue - coated prisoners were off their flights, there was great noise of protest at once, and they shrieked in excited protest all the time, hoping that no one was in them. Then they shrieked and yelled again, until their sailor gasped with loud protest, and these were anxious voices trying to protest, which was certainly no use. There was a moment's time to rest, and Trot got up with her husband and Trot and Cap'n Bill on their backs and made a hearty effort to suppress the angry voice, which seemed to show a good deal of patience at once. " [...]

Table 4.15: Example stories generated by the models and enhancement after fine-tuning a second round with synthetic data.

- Misinformation Harms
- Malicious Uses
- Human-Computer Interaction Harms
- Automation, Access, and Environmental Harms

The mitigation of these risks cannot be overlooked and should be a prominent part of research. The point of origin of a harm can be an indication for appropriate mitigations, i.e. leaking personal information. This can be mitigated by better reduction or curation of training data. Implementing mitigations is not an easy task, it requires a wide range of expertise. One has to make sure that mitigating one risk does not aggravate another risk.

Chapter 5

Conclusion and Further Research

Developing a model to be able to generate stories in a multidomain, is not an easy task. Using several text properties to re-rank candidate stories, improves the statistics of the stories. Even when using a fine-tuned discriminator to re-rank candidates to be as human-like as possible, improves the statistical scores. A combination of the properties and the discriminator leads to even better linguistic scores, but when trying to distinguish the generated stories from human-written stories, it became easier for the BERT classifier, having an accuracy of 78 %. Just using the discriminator leads to the worst accuracy, in this case the lower, the better, for the BERT classifier, approximately 61 %. Meanwhile, the multidomain model without enhancement, has an accuracy of 85 %, while the state-of-the-art models score 95 % or higher.

On the other hand, when using a combination of the properties and the discriminator, the domains were more easily identifiable for a fine-tuned BERT classifier in comparison to only using the discriminator, 73 % versus 78 %. Meanwhile, differentiating between the two genres based on the gold sentences is not an easy task, scoring an accuracy of 75%, showing that the overlap of the most common words makes it a challenging task. Using the Multidomain model without enhancements, gave an accuracy of 82 %. The difference between the generated stories is minimal, some models might use words that differentiate them from human written stories, and might be less fluent. Therefore, the linguistic analysis cannot be the leading factor on deciding which model performs better, especially when the scores are similar to each other. However, it can indicate which models are generating less fluent or coherent stories.

Furthermore, the influence of the beam width when using Sampling as the decoding method, is surprising, since using no beams is the best choice. Synthetic data can be used to further improve the model during a second round of fine-tuning, while using an enhancement to create the synthetic data is of no influence.

When comparing different pre-trained models after being fine-tuned for only one domain, it is noticeable that the decoder only models, GPT models, are performing worse than the baseline models and the T5 Base model. Although, the T5 Base model is still slightly underperforming when compared with the state-of-the-art models based on linguistic analysis. The smaller models all seem to be dealing with performance issues when being fine-tuned for too long, and the slightly bigger models tend to perform better when generating stories.

When looking at actually generated stories, the Fusion model seems not be able to generate any coherent, fluent story, while scoring decent on linguistic analysis. On the other hand, the T5 Base model scores slightly worse, but the stories are more fluent and coherent, even when compared to the FairyTailor model. This shows once again that the linguistic analysis cannot be the leading factor when deciding which model to use, but can be used as an indicator. Furthermore, the quality of the standard models without any fine-tuning is inferior to any of the other models.

For further research, a more in-depth analysis into linguistic analysis can be performed to create better indicators to which models are performing better or worse. How the models would behave when more domains are included, would be an interesting topic to research in the future. Furthermore, a more in-depth research between the investigated models could help to decide when to use which model. Comparing the models trained in this research with bigger models to evaluate the performance difference, could be a fascinating topic, as well as analysing different language models not investigated in this research. The influence of beam size could be researched further, since only using Sampling is according to the experiments performing best. The influence of the use of synthetic data would be interesting to be further analysed. Finally, when fine-tuning a model, or during a second fine-tuning round, it would be interesting if an Generative Adversarial Network can be used, i.e. SeqGAN.

References

Adamopoulou, E., & Moussiades, L. (2020). An overview of chatbot technology. In I. Maglogiannis, L. Iliadis, & E. Pimenidis (Eds.), *Artificial intelligence applications and innovations* (pp. 373–383). Cham: Springer International Publishing.

Alabdulkarim, A., Li, W., Martin, L. J., & Riedl, M. O. (2021). Goal-directed story generation: Augmenting generative language models with reinforcement learning. *CoRR*, *abs/2112.08593*. Retrieved from `https://arxiv.org/abs/2112.08593`

Alhussain, A., & Azmi, A. (2021, 05). Automatic story generation: A survey of approaches. *ACM Computing Surveys*, *54*, 1-38. doi: 10.1145/3453156

Ansag, R. A., & Gonzalez, A. J. (2021). State-of-the-art in automated story generation systems research. *Journal of Experimental & Theoretical Artificial Intelligence*, *0*(0), 1-55. Retrieved from `https://doi.org/10.1080/0952813X.2021.1971777` doi: 10.1080/0952813X.2021.1971777

Baccianella, S., Esuli, A., & Sebastiani, F. (2010). Sentiwordnet 3.0: An enhanced lexical resource for sentiment analysis and opinion mining. In N. Calzolari et al. (Eds.), *Lrec.* European Language Resources Association. Retrieved from `http://nmis.isti.cnr.it/sebastiani/Publications/LREC10.pdf`

Bensaid, E., Martino, M., Hoover, B., Andreas, J., & Strobelt, H. (2021). Fairytailor: A multimodal generative framework for storytelling. *CoRR*, *abs/2108.04324*. Retrieved from `https://arxiv.org/abs/2108.04324`

Black, S., Gao, L., Wang, P., Leahy, C., & Biderman, S. (2021, March). *GPT-Neo: Large Scale Autoregressive Language Modeling with Mesh-Tensorflow.* Zenodo. Retrieved from `https://doi.org/10.5281/zenodo.5297715` (If you use this software, please cite it using these metadata.) doi: 10.5281/zenodo.5297715

Blain, F. (2017). Exploring hypotheses spaces in neural machine translation..

Borgeaud, S., & Emerson, G. (2019). Leveraging sentence similarity in natural language generation: Improving beam search using range voting. *CoRR*, *abs/1908.06288*. Retrieved from `http://arxiv.org/abs/1908.06288`

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., ... Amodei, D. (2020). Language models are few-shot learners. *CoRR*, *abs/2005.14165*. Retrieved from `https://arxiv.org/abs/2005.14165`

Burstein, J., & Wolska, M. (2003). Toward evaluation of writing style: Finding overly repetitive word use in student essays. In *Proceedings of the tenth conference on european chapter of the association for computational linguistics - volume 1* (p. 35–42). USA: Association for Computational Linguistics. Retrieved from https://doi.org/10.3115/1067807.1067814 doi: 10.3115/1067807.1067814

Cai, Z., & McNamara, D. (2012, 01). Syntagmatic, paradigmatic, and automatic n-gram approaches to assessing essay quality. In (p. 214-219).

Celikyilmaz, A., Clark, E., & Gao, J. (2020). Evaluation of text generation: A survey. *CoRR*, *abs/2006.14799*. Retrieved from https://arxiv.org/abs/2006.14799

Chen, S. F., Beeferman, D., & Rosenfeld, R. (2008, 1). Evaluation Metrics For Language Models.
doi: 10.1184/R1/6605324.v1

Cohen, E., & Beck, J. C. (2018). Unconstrained) beam search is sensitive to large search discrepancies..

Devlin, J., Chang, M., Lee, K., & Toutanova, K. (2018). BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, *abs/1810.04805*. Retrieved from http://arxiv.org/abs/1810.04805

Fan, A., Lewis, M., & Dauphin, Y. N. (2018). Hierarchical neural story generation. *CoRR*, *abs/1805.04833*. Retrieved from http://arxiv.org/abs/1805.04833

Fellbaum, C. (Ed.). (1998). *WordNet: an electronic lexical database.* MIT Press.

Flesch, R. (1948, June). A new readability yardstick. *Journal of Applied Psychology*, *32*(3), p221 - 233.

Gao, L., Biderman, S., Black, S., Golding, L., Hoppe, T., Foster, C., ... others (2020). The pile: An 800gb dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027*.

Garg, A., & Agarwal, M. (2019). Machine translation: A literature review. *CoRR*, *abs/1901.01122*. Retrieved from http://arxiv.org/abs/1901.01122

Gehring, J., Auli, M., Grangier, D., Yarats, D., & Dauphin, Y. N. (2017). Convolutional sequence to sequence learning. *CoRR*, *abs/1705.03122*. Retrieved from http://arxiv.org/abs/1705.03122

Graesser, A., McNamara, D., Louwerse, M., & Cai, Z. (2004). Coh-metrix: analysis of text on cohesion and language. *Behavior Research Methods, Instruments & Computers*, *36*(2), 193–202.

Halliday, M. A. K., & Hasan, R. (1976). *Cohesion in english.* London: Longman.

Herrera-González, B., Gelbukh, A., & Calvo, H. (2020, 10). Automatic story generation: State of the art and recent trends. In (p. 81-91). doi: 10.1007/978-3-030-60887-3_8

Holtzman, A., Buys, J., Forbes, M., & Choi, Y. (2019). The curious case of neural text degeneration. *CoRR*, *abs/1904.09751*. Retrieved from http://arxiv.org/abs/1904.09751

Ippolito, D., Grangier, D., Callison-Burch, C., & Eck, D. (2019, June). Unsupervised hierarchical story infilling. In *Proceedings of the first workshop on narrative understanding* (pp. 37–43). Minneapolis, Minnesota: Association for Computational Linguistics. Retrieved from `https://aclanthology.org/W19-2405` doi: 10.18653/v1/W19-2405

Jaccard, P. (1912, February). The distribution of the flora in the alpine zone. *New Phytologist*, *11*(2), 37-50. Retrieved from `http://www.jstor.org/stable/2427226?seq=3`

Jain, P., Agrawal, P., Mishra, A., Sukhwani, M., Laha, A., & Sankaranarayanan, K. (2017). Story generation from sequence of independent short descriptions. *CoRR*, *abs/1707.05501*. Retrieved from `http://arxiv.org/abs/1707.05501`

Koehn, P., & Knowles, R. (2017). Six challenges for neural machine translation. *CoRR*, *abs/1706.03872*. Retrieved from `http://arxiv.org/abs/1706.03872`

Kumar, S., & Byrne, W. (2004, May 2 - May 7). Minimum Bayes-risk decoding for statistical machine translation. In *Proceedings of the human language technology conference of the north American chapter of the association for computational linguistics: HLT-NAACL 2004* (pp. 169–176). Boston, Massachusetts, USA: Association for Computational Linguistics. Retrieved from `https://aclanthology.org/N04-1022`

Lin, C.-Y. (2004, July). ROUGE: A package for automatic evaluation of summaries. In *Text summarization branches out* (pp. 74–81). Barcelona, Spain: Association for Computational Linguistics. Retrieved from `https://aclanthology.org/W04-1013`

Liu, P. J., Saleh, M., Pot, E., Goodrich, B., Sepassi, R., Kaiser, L., & Shazeer, N. (2018). Generating wikipedia by summarizing long sequences. *CoRR*, *abs/1801.10198*. Retrieved from `http://arxiv.org/abs/1801.10198`

McGuffie, K., & Newhouse, A. (2020). The radicalization risks of gpt-3 and advanced neural language models. *ArXiv*, *abs/2009.06807*.

McNamara, D., & Mccarthy, P. (2010, 01). Linguistic features of writing quality. *Written Communication - WRIT COMMUN*, *27*, 57-86. doi: 10.1177/0741088309351547

Napoles, C., Sakaguchi, K., & Tetreault, J. (2016, November). There's no comparison: Reference-less evaluation metrics in grammatical error correction. In *Proceedings of the 2016 conference on empirical methods in natural language processing* (pp. 2109–2115). Austin, Texas: Association for Computational Linguistics. Retrieved from `https://aclweb.org/anthology/D16-1228`

Papineni, K., Roukos, S., Ward, T., & Zhu, W.-J. (2002, July). Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the association for computational linguistics* (pp. 311–318). Philadelphia, Pennsylvania, USA: Association for Computational Linguistics. Retrieved from `https://aclanthology.org/P02-1040` doi: 10.3115/1073083.1073135

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., & Sutskever, I. (2019).

Language models are unsupervised multitask learners.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., ... Liu, P. J. (2019). Exploring the limits of transfer learning with a unified text-to-text transformer. *CoRR*, *abs/1910.10683*. Retrieved from `http://arxiv.org/abs/1910.10683`

Rashkin, H., Celikyilmaz, A., Choi, Y., & Gao, J. (2020). Plotmachines: Outline-conditioned generation with dynamic plot state tracking. *CoRR*, *abs/2004.14967*. Retrieved from `https://arxiv.org/abs/2004.14967`

Roemmele, M., Gordon, A. S., & Swanson, R. (2017). Evaluating story generation systems using automated linguistic analyses..

Sherstinsky, A. (2018). Fundamentals of recurrent neural network (RNN) and long short-term memory (LSTM) network. *CoRR*, *abs/1808.03314*. Retrieved from `http://arxiv.org/abs/1808.03314`

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, *15*(56), 1929–1958. Retrieved from `http://jmlr.org/papers/v15/srivastava14a.html`

Turc, I., Chang, M.-W., Lee, K., & Toutanova, K. (2019). Well-read students learn better: On the importance of pre-training compact models. *arXiv preprint arXiv:1908.08962v2*.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... Polosukhin, I. (2017). Attention is all you need. *CoRR*, *abs/1706.03762*. Retrieved from `http://arxiv.org/abs/1706.03762`

Wang, S., Durrett, G., & Erk, K. (2020). Narrative interpolation for generating and understanding stories. *CoRR*, *abs/2008.07466*. Retrieved from `https://arxiv.org/abs/2008.07466`

Weidinger, L., Mellor, J., Rauh, M., Griffin, C., Uesato, J., Huang, P., ... Gabriel, I. (2021). Ethical and social risks of harm from language models. *CoRR*, *abs/2112.04359*. Retrieved from `https://arxiv.org/abs/2112.04359`

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., ... Brew, J. (2019). Huggingface's transformers: State-of-the-art natural language processing. *CoRR*, *abs/1910.03771*. Retrieved from `http://arxiv.org/abs/1910.03771`

Yao, L., Peng, N., Weischedel, R. M., Knight, K., Zhao, D., & Yan, R. (2018). Plan-and-write: Towards better automatic storytelling. *CoRR*, *abs/1811.05701*. Retrieved from `http://arxiv.org/abs/1811.05701`

Appendix A

Example Stories

An overview of different generated stories are being shown in this Appendix.

A.1 Different Generation Methods

An overview of different generation methods, each with the input and a generated story or sentence can be found in Tables A.1 - A.3. These are all examples takes from the available data or the papers respectively.

Model	Input & Output
Fusion	Prompt:[WP] A man finally discovers his superpower ... well into his 80 's . Story: I can not believe that no one had ever heard of me . I have been all alonefor so long , but it never felt like a bad idea . I had been in the dark for so long that I lost my balance as a equal . I had been alone since I wasborn and still went to bed without knowing what to expect . No one is sure where I came from or where I was but I had some really bignthings to do . I had a heart attack and could not even find a safe location . <newline ><newline >But one day , I woke up in the middle of a forest . I found a guy who seemed to be sitting right next to me . He said , ‘ ‘ Iam Dr. <unk >. I am Dr. <unk >. I am Dr. <unk >. I am the first person to try to groaned . ”

Table A.1: First set of examples from the models mentioned in Section 2.2

Model	Input & Output
Fairy	Prompt: The Truth is Written in the Stars Story: There was once upon a time, long before the writing of the Epistleswas known, a mighty kingdom whose name was in every hand ofthe people of a kingdom of stars. And among them,in the midst of the shining city of Cyrence, was a beautiful maiden, so long ago asthe time of the poet, when the world was but half its present size. Her name, for which it was named,was Aurora. But she was not the only one of the stars in the sky in whom the light of prophecyhad fitted a sweet spot.
Plan and Write	Title: The Bike Accident Storyline: Carrie → bike → sneak → nervous → leg Story: Carrie had just learned how to ride a bike. She didn't have a bike of her own. Carrie would sneak rides on her sister's bike. She got nervous on a hill and crashed into a wall. The bike frame bent and Carriegot a deep gash on her leg.
Independent	Description 1: A tour boat travels down the river on pretty day. Description 2: A bunch of people walking on the street next to stores. Description 3: A city full of older buildings sits on a river. Story: On their trip to location , they arrive in front of a river . They decide to check out the city . They think its too packed with people , so they go sight seeing . The indoor poor tempts them , but they decide not to jump in . They come across some ducks .
Story Infilling	Part 1: In the morning when he awoke, he began to search over hill and dale for this pretty flower; and eight long days he sought for it in vain: but on the ninth day Part 2: a large dewdrop, as big as a costly pearl. Then he plucked the flower, and set out and travelled day and night, till he came again tothe castle. Predicted middle part: he found it in the morning, and gathered it up in the dew of the fountain, and it was
Interpolation	Beginning: Jim went hiking alone at the state park. End: Jim was rescued. Predicted text: He got lost on a trail. He slipped and fell. Jim broke his leg.

Table A.2: Second set of examples from the models mentioned in Section 2.2

Model	Input & Output
Reinforcement Learning	Goal: Discovery Context: The tobacco company Hadara creates a form of super tobacco, which in turn inhabited by a Predicted text: form of super tobacco beetle, which survive the cigarette processing and whose eggs are contained in the smoke of these killer cigarettes, presumably called brand Alex.
Plotmachines	Story outline: big bird's birthday celebration. cookie monster eats. roller skating rink. big birthday cake Story: It is Big Bird's birthday, and he goes to the roller skating rink with his friends. Back at Sesame Street, Maria and Susan take out the big birthday cake and leave it on a table. Cookie Monster sees the cake, but instead of eating it and spoiling the party, he eats a chair and other things all over Sesame Street. Big Bird and the other skaters return to Sesame Street and are shocked at what Cookie Monster ate, though the cake is safe. Gina and Count Von Count presents the cake to Big Bird. It has 548 candles even though Big Bird is 6 years old. At the end, when Gina announces the sponsors, Cookie Monster eats them along with his cake.

Table A.3: Third set of examples from the models mentioned in Section 2.2

A.2 Fairy Tale stories

Model	It was the day when the farmer was gathering his pumpkins in the cornfield, and he just happened to remember the seeds he had planted in the home - patch, and he got out of his wagon to see what had become of them.
F-GPT-2 Small	The weather blew cold enough here already (it seems a year on) for rain there ;however this is soon set very bad by little thinnylintasf -plums as fanned up against trees along those who go early again that get only an occasional look a their long poles from close among dashes oaken with red tusk through some woods beyond any hedge under rocks upon steep grass within four miles radius or more west above ground zero.–This might seem quite unlike anything else encountered all right throughout Europe since our arrival three weeks ago.—Atonce everything began – lights stood flashing before sight alike white stileon figures hovering upon each other beside one solitary blackness [...]
F-GPT-2 Medium	He found one white, sturdy foot on a short hedge, and if he had been that way Dennis Sowers could not have stood upright long enough to pull up his pole and offer speeches against fertilizers which are becoming too common. But, fortunately for Mr. Sower, morning as issue, there were few people about in sufficient numbers so that any unintelligible sound could scarcelyain disturbing. The family, all together perhaps taking their paddles from among twenty or thirty heads, mashed almost unanimously into an angry class, at least until sundown. [...]

Table A.4: Generated stories based on the same prompt for the rest of the fine-tuned models. The prompt originated out of the test set of the Fairy Tales dataset.

A.3 WritingPrompts

Model	[WP] A boy asks a girl out . It 's high school . It 's awkward . Narrate it from the point of view of a nature documentary .
F-GPT-2 Small	If you stand here to talk with these old friends, let her take your advice!This must certainly not be their first attempt ; young ladies do more mischief in this country than all boys go under our tutelary eyeswithout any " dhrink off? The girls only wear an earpiece on each armwhen they use fire and cry as long now isn't – and neither one who getswhat we call credit nor anything else — even if those women have known well why so much love did break me once there was–who gave them ten pound bills five years hence on some great riverside inn about seven streets- over four near Bristol Street [...]
F-GPT-2 Medium	The first time I tried it, my experience was one Johnny has not, forgotten. There came on board two or three cameras ; several takes – all wrong ways – which none believed. But they cut to Mr. Brownlow, and when he fired, his film is that of an officer just going about denying that he has seen Annie Trotwood anywhere since she went away. Doesn't work at all ; he has no memory for a till you call his bluffing into visible reality.... Yet if viewed fairly and attentively, this account may be found in many a book about bad casting. The other boys had been playing with the notion that that night demonstration might bear fruit. They all said they had been happy fools waiting forty years for female friends amid [...]
F-GPT-Neo	? " Pintaro was in any change, with her that she wrote. This when you say you should like such ; that were just in the house had an easy book in such moments, as not in such things could possibly write to them? When his only hope could tell such an hour and was of reading for five hundred pounds. But I said Mrs. and very easy than what we took in case between her work– ' a - three year passed away his paper to me once a number'lt to give anything like Mr,—thoughiness, would be sure he ever spoken with a young gentleman about and read on each state — but when she felt him up and so deeply impressed all his hand which we found before their mindto go up for two ladies at my own present in India was at great hopes for myself. [...]

Table A.5: Stories generated by the other fine-tuned models based on a prompt from the WritingPrompst dataset.

Model	[WP] A boy asks a girl out . It 's high school . It 's awkward . Narrate it from the point of view of a nature documentary .
F-T5	" " I'm going to tell you, " said the boy. " I'm going to tell you, " said the boy. " I 'll tell you, " said the boy. " I 'll tell you, " said the boy. " I 'll tell you, " said the boy. " I 'll tell you, " said the boy. " I 'll tell you, " said the boy. " I 'll tell you, " said the boy. " I 'll tell you, " said the boy. " I 'll tell you, " said the boy. " I 'll tell you, " said the boy. " I 'll tell you, " said the boy. " I 'll tell you, " said the boy. " I 'll tell you, " said the boy. " [...]

Table A.6: Stories generated by the other fine-tuned T5 model based on a prompt from the WritingPrompst dataset.

Appendix B

Gutenberg Books

This section provides an overview of the books used in the creation of the two datasets.

B.1 Fairy Tales

Japanese Fairy Tales, Plain Tales of the North, The Wind in the Willows, The Louisa Alcott Reader, A Wonder Book for Girls and Boys, Tanglewood Tales, The Pig Brother and Other Fables and Stories, The Worlds Greatest Books, Vol 3, A Christmas Hamper, A Little Princess, Household Tales, Aesop Fables, A Tale of Two Cities, Among the Forest People, Celtic Tales, Andersens Fairy Tales, Little Women, Childhoods Favorites and Fairy Stories, Christmas Every Day and Other Stories, East of the Sun and West of the Moon, Dramatic Reader for Lower Grades, Comic History of the United, Fairy Tales Second Series, English Fairy Tales, Hindu Tales from the Sanskrit, Folk-Tales of the Khasis, Folk Tales from the Russian, Moby Dick, Merry Stories and Funny Pictures, Indian Fairy Tales, My Fathers Dragon, Indian Why Stories, Peter Pan, Myths Retold by Children, My Man Jeeves, Simla Village Tales Or Folk Tales from the Himalayas, Sense and Sensibility, Snow-White or The House in the Wood, The Idiot, The Adventures of Pinocchio, The Adventures of Sherlock Holmes, The Best American Humorous Short Stories, The Adventures of Tom Sawyer, The Blue Fairy Book, The Happy Prince, The Book of Dragons, The Little Red Hen, The Little Lame Prince, The Jungle Book, The Magical Mimics in Oz, The Paradise of Children, The Secret Garden, The Snow Image, The Prince and Betty, The Peace Egg and Other tales, The Tale of Johnny Town-Mouse, The Valveteen Rabit, The Wonderful Wizard of Oz, The Time Machine, Treasure Island, True Stories of Wonderful Deeds, Wonder Stories, Goody Two-Shoes, he Marvelous Exploits of Paul Bunyan, Christmas Every Day and Other Stories, The Childrens Book of Thanksgiving Stories.

B.2 Fantasy Books

Le Morte dArthur Volume 1, The Mabinogion, Le Morte dArthur Volume 2, The Legends Of King Arthur And His Knights, The Merry Adventures of Robin Hood, The House on the Borderland, The Story of the Volsungs (Volsunga Saga), Four Arthurian Romances, News from Nowhere, The Night Land, The Book of Wonder, The Gods of Pegana, The Well at the World's End A Tale, Fifty-One Tales, Men of Iron, The Wood Beyond the World, The Sword of Welleran and Other Stories, A Dreamers Tales, Time and the Gods, The Master Key, The Story of the Champions of the Round Table, The Crock of Gold, The House of the Wolfings, Jurgen A Comedy of Justice, Kai Lungs Golden Hours, Tales of Wonder, Cliges A Romance, Otto of the Silver Hand, The Water of the Wondrous Isles, Gulliver of Mars, The Roots of the Mountains, The Wallet of Kai Lung, Tales of Three Hemispheres, The Story of the Glittering Plain, Tales of War, Figures of Earth A Comedy of Appearances, The History of Caliph Vathek, The Sea Fairies, The Rivet in Grandfather's Neck A Comedy of Limitations, The Hollow Land, The Enchanted Island of Yew, Taboo, Don Rodriguez Chronicles of Shadow Valley, The Mabinogion Vol. 3, The Eagles Shadow, The Certain Hour, The Surprising Adventures of the Magical Monarch of Mo and His People, The Mabinogion Vol. 2, In the Court of King Arthur, Child Christopher and Goldilind the Fair, Unhappy Far-Off Things, The Story of Grettir the Strong, Plays of Near & Far, Twilight Land, Mary Louise, Chivalry Dizain des Reines, The Mirror of Kong Ho, The Jewel Merchants A Comedy in One Act, Sky Island, Plays of Gods and Men, Young Robin Hood, The World of Romance, The Line of Love, Domnei A Comedy of Woman-Worship, The Ruby of Kishmoor, The Cords of Vanity A Comedy of Shirking, Gallantry Dizain des Fetes Galantes-Maybe remove, Old French Romances, Done into English, Rinkitink in Oz, The Lost Princess of Oz.

Appendix C

Results

This appendix show all the results of the different models for all the different epochs.

C.1 Fairy Tales

epochs	F-GPT-2 Small	F-GPT-2 Medium	F-GPT-Neo	F-T5	F-T5 Base
5	1510	1412	1343	896	946
22	1383	1393	1338	931	1001
50	1330	1359	1306	946	1045
75	1343	1379	1339	961	1077
100	1337	1370	1322	973	1084
125	1355	1363	1309	966	1098
150	1342	1359	1290	999	1088

Table C.1: Character count for the fine-tuned models for different training epochs on Fairy Tales dataset.

epochs	F-GPT-2 Small	F-GPT-2 Medium	F-GPT-Neo	F-T5	F-T5 Base
5	268	248	280	244	251
22	255	245	271	248	254
50	245	240	251	250	255
75	245	247	253	248	258
100	242	247	243	249	257
125	245	248	239	249	260
150	244	250	234	250	258

Table C.2: Word count for the fine-tuned models for different training epochs on Fairy Tales dataset.

epochs	F-GPT-2 Small	F-GPT-2 Medium	F-GPT-Neo	F-T5	F-T5 Base
5	0.955971	0.918062	0.978314	0.984038	0.974334
22	0.947499	0.938112	0.968192	0.977466	0.975266
50	0.934907	0.943471	0.92805	0.975017	0.979901
75	0.942358	0.953011	0.934369	0.966858	0.983945
100	0.940641	0.954112	0.914958	0.966638	0.950391
125	0.949636	0.958569	0.903135	0.969346	0.985987
150	0.949314	0.960542	0.888514	0.972458	0.976984

Table C.3: Grammar scores for the fine-tuned models for different epochs on Fairy Tales dataset.

epochs	F-GPT-2 Small	F-GPT-2 Medium	F-GPT-Neo	F-T5	F-T5 Base
5	0.1206	0.1873	0.0422	0.012	0.0132
22	0.1481	0.1954	0.0909	0.0152	0.0289
50	0.1675	0.1954	0.1595	0.0194	0.0489
75	0.167	0.1801	0.1617	0.0233	0.0584
100	0.175	0.1746	0.1912	0.0274	0.0652
125	0.1683	0.1713	0.2049	0.03	0.0656
150	0.1703	0.1647	0.2205	0.0353	0.0687

Table C.4: The Lexical Diversity for the fine-tuned models for different training epochs on Fairy Tales dataset.

epochs	F-GPT-2 Small	F-GPT-2 Medium	F-GPT-Neo	F-T5	F-T5 Base
5	0.9911	0.995	0.9341	0.082	0.0798
22	0.9917	0.9938	0.9638	0.1148	0.225
50	0.9916	0.9903	0.9808	0.1649	0.4683
75	0.9898	0.9829	0.9825	0.2016	0.5836
100	0.9904	0.976	0.9843	0.2429	0.6413
125	0.987	0.9663	0.984	0.2827	0.6536
150	0.9875	0.9601	0.9857	0.3392	0.6655

Table C.5: Trigram diversity for the finetunded models for different training epochs on Fairy Tales dataset.

epochs	F-GPT-2 Small	F-GPT-2 Medium	F-GPT-Neo	F-T5	F-T5 Base
5	0.1399	0.11	0.213	0.2491	0.2735
22	0.1297	0.1128	0.1742	0.2845	0.3321
50	0.117	0.115	0.1255	0.3097	0.3052
75	0.1232	0.1242	0.1254	0.3067	0.2712
100	0.1222	0.1274	0.1141	0.3256	0.2705
125	0.1283	0.1276	0.1096	0.2982	0.263
150	0.1262	0.1341	0.1017	0.3057	0.2514

Table C.6: Jaccard Similarity for the fine-tuned models for different training epochs on Fairy Tales dataset.

C.2 WritingPrompts

epochs	F-GPT-2 Small	F-GPT-2 Medium	F-GPT-Neo	F-T5	F-T5 Base
5	1569	1520	1398	842	879
22	1444	1491	1398	828	930
50	1395	1422	1368	811	944
75	1407	1443	1408	837	964
100	1401	1444	1387	878	1025
125	1424	1423	1381	865	1027
150	1418	1409	1357	949	1057

Table C.7: Character count for the fine-tuned models for different training epochs on WritingPrompts dataset.

epochs	F-GPT-2 Small	F-GPT-2 Medium	F-GPT-Neo	F-T5	F-T5 Base
5	279	265	294	238	235
22	265	261	283	232	242
50	256	253	259	228	235
75	255	258	261	225	234
100	253	259	251	230	244
125	254	258	249	228	243
150	255	259	243	237	249

Table C.8: Word count for the fine-tuned models for different training epochs on WritingPrompts dataset.

epochs	F-GPT-2 Small	F-GPT-2 Medium	F-GPT-Neo	F-T5	F-T5 Base
5	0.959292	0.934027	0.97866	0.98674	0.929867
22	0.948917	0.9401	0.965758	0.962653	0.93133
50	0.935285	0.938	0.914732	0.951938	0.966976
75	0.941312	0.949595	0.923906	0.945906	0.978568
100	0.941939	0.956716	0.898831	0.944989	0.973597
125	0.948045	0.955492	0.890752	0.944783	0.97894
150	0.94861	0.957567	0.877495	0.954929	0.986029

Table C.9: Grammar scores for the fine-tuned models for different training epochs on WritingPrompts dataset.

epochs	F-GPT-2 Small	F-GPT-2 Medium	F-GPT-Neo	F-T5	F-T5 Base
5	0.1215	0.171	0.04	0.0098	0.0138
22	0.1534	0.1948	0.0963	0.0153	0.027
50	0.172	0.201	0.1798	0.0194	0.0511
75	0.1746	0.184	0.1763	0.0244	0.0666
100	0.1794	0.1763	0.2092	0.0292	0.0699
125	0.1787	0.1786	0.2175	0.0326	0.073
150	0.1758	0.1727	0.2303	0.0403	0.0742

Table C.10: Lexcical Diversity for the fine-tuned models for different training epochs on WritingPrompts dataset.

epochs	F-GPT-2 Small	F-GPT-2 Medium	F-GPT-Neo	F-T5	F-T5 Base
5	0.9899	0.993	0.9261	0.0678	0.0683
22	0.9904	0.994	0.9631	0.1009	0.1841
50	0.9906	0.9913	0.9827	0.1306	0.3987
75	0.9909	0.9857	0.9829	0.1728	0.5387
100	0.9909	0.98	0.9865	0.2102	0.6094
125	0.9895	0.9751	0.9846	0.2337	0.629
150	0.9883	0.9702	0.9857	0.3229	0.6544

Table C.11: Trigram Diversity for the fine-tuned models for different training epochs on WritingPrompts dataset.

epochs	F-GPT-2 Small	F-GPT-2 Medium	F-GPT-Neo	F-T5	F-T5 Base
5	0.1333	0.1133	0.2079	0.1293	0.1418
22	0.1168	0.1077	0.1605	0.1539	0.1873
50	0.1099	0.1056	0.1086	0.1769	0.2057
75	0.1118	0.1154	0.1136	0.1838	0.195
100	0.111	0.1197	0.0995	0.1932	0.1967
125	0.1151	0.1174	0.0982	0.1973	0.1995
150	0.1171	0.1205	0.0936	0.2099	0.2021

Table C.12: Jaccard Similarity for the fine-tuned models for different training epochs on WritingPrompts dataset.

C.3 Beam size

Beam Size	Text Properties Re-ranking	Discriminator Re-ranking
1	0.978637	0.975151
5	0.967236	0.963982
10	0.964264	0.945613
15	0.962082	0.941786
20	0.957372	0.943624
25	0.959584	0.940059
30	0.955045	0.937321

Table C.13: Grammar score for two enhancements for different beam size.

Beam Size	Text Properties Re-ranking	Discriminator Re-ranking
1	0.0939	0.1053
5	0.0728	0.0770
10	0.0689	0.0728
15	0.0665	0.0719
20	0.0659	0.0701
25	0.065	0.0686
30	0.0643	0.0671

Table C.14: Lexical Diversity for two enhancements for different beam size.

Beam Size	Text Properties Re-ranking	Discriminator Re-ranking
1	0.8286	0.845
5	0.6463	0.6419
10	0.5977	0.5789
15	0.566	0.5548
20	0.5482	0.5381
25	0.5412	0.5257
30	0.5308	0.5116

Table C.15: Trigram Diversity for two enhancements for different beam size.

Beam Size	Text Properties Re-ranking	Discriminator Re-ranking
1	0.318	0.3036
5	0.3847	0.3771
10	0.3886	0.3794
15	0.3966	0.3845
20	0.3987	0.3803
25	0.3972	0.3851
30	0.4001	0.392

Table C.16: Jaccard Similarity for two enhancements for different beam size.

Beam Size	Enhancement 2	Enhancement 4
1	708	690
5	618	612
10	608	597
15	598	589
20	595	586
25	592	579
30	590	577

Table C.17: Character count for two enhancements for different beam size.

Beam Size	Enhancement 2	Enhancement 4
1	152	148
5	141	139
10	140	136
15	139	134
20	138	133
25	137	132
30	137	132

Table C.18: Word count for two enhancements for different beam size.