

HEC MONTRÉAL

“Don’t forget to Tag me!”: The effect of metadata in open-domain story generation

par

Lourdes Crivelli

HEC Montréal

Directeur/Directrice de recherche

Laurent Charlin

Sciences de la gestion

Data Science and Business Analytics

*Mémoire présenté en vue de l’obtention
du grade de maîtrise ès sciences en gestion
(M. Sc.)*

12,2020

© Lourdes Crivelli, 2020

Résumé

En permettant aux ordinateurs d'écrire des textes à la manière des humains, les modèles de langage basés sur les Transformers ont révolutionné le domaine de la génération du langage naturel. Par contre, ces modèles produisent du texte séquentiellement, sans possibilité de conditionner la génération autre qu'avec une phrase initiale (*prompt*). Dans ce travail, nous étudions comment générer le texte à produire en conditionnant le générateur avec des métadonnées (tags). Sur la base de travaux précédents, nous utilisons le récent modèle de langage GPT-2 pour comparer l'efficacité du conditionnement sur différentes métadonnées pour générer des créations littéraires (creative stories). Nous étudions l'effet des types de métadonnées suivants : le titre de l'histoire, l'élément à produire, le genre à produire, le plan de la trame à produire et les paragraphes précédents de l'histoire. Nous utilisons l'ensemble de données WikiPlots auquel nous ajoutons le genre, à partir du site IMDB, et nous traitons les données pour minimiser la contamination entre les ensembles de test et train. Nous proposons une analyse quantitative de l'impact des types de métadonnées seul ou en combinaison. Notre travail est le premier à inclure également une analyse qualitative sur la manière dont les différentes métadonnées peuvent influencer le texte généré. Nous montrons que, sur la base d'une analyse quantitative, le genre est la balise qui a le plus d'impact sur la perplexité de l'ensemble de test d'un modèle mais son influence diminue au fur et à mesure qu'elle est combinée avec des métadonnées plus spécifiques comme le plan de la trame ou le paragraphe précédent. Nous montrons également que l'élément à produire et le paragraphe précédent sont des balises faciles à obtenir qui ont une grande influence sur le conditionnement de la génération.

Mots clés : Génération du langage naturel ; Modèles de langage ; GPT-2 ; Conditionnement de la génération de texte ; Métadonnées ; Génération de récits ; NLG ; Apprentissage automatique ; Apprentissage approfondi

Méthodes de recherche : Recherche quantitative, recherche qualitative

Abstract

Transformer-based Language Models have revolutionized the field of Natural Language Generation, enabling computers to write human-like text. Currently, the text is produced in a sequential manner, with no option to condition the generation other than an initial sentence (prompt). In our work, we study how to condition the produced text by adding metadata (tags) to the training corpus and use this metadata as part of the input during the generation step. Building on previous work, we use the recent GPT-2 language model to compare the efficacy of conditioning on different metadata for creative-story generation. We study the effect of the following metadata: title, discourse, genre, plot outline and previous paragraphs of the story. We use the WikiPlots dataset, adding the genre tag from IMDB and curating it to minimize contamination between the test and train sets. We propose a quantitative analysis of their impact, both at an individual level and in combination. Our work is the first to also include a qualitative analysis on how different metadata can influence the generated text. We show that, on a quantitative analysis, genre is the tag that has the most impact on a model's perplexity but its influence decreases as it is combined with more specific metadata such as the plot outline or the previous paragraph. We also show that discourse and the previous paragraph are easy to obtain tags that have a great influence in conditioning the generation.

Keywords: Natural Language Generation; Language Models; GPT-2; Conditioning Text Generation; Metadata; Story-Generation; NLG; Machine Learning; Deep Learning

Research methods: Quantitative Research, Qualitative Research

Table of contents

Résumé	v
Abstract	vii
Table of contents	ix
List of tables and figures	xi
List of abbreviations and acronyms	xiii
Preface	1
Introduction	3
I.1 Study Context	3
I.2 Problem & Research Question	5
I.3 Contributions to the field	5
I.4 Dissertation Flow	6
Chapter 1 Literature Review	7
1.1 History of NLG	7
1.1.1 Rule-based Methods and Statistical Language Generation	7
1.1.2 Deep Neural Networks	9
1.1.3 Transformers	12
1.1.4 Transformer-Based Language Models	13
1.2 Evaluation Methods	16
1.2.1 Untrained Automatic Evaluation Metrics	16
1.2.2 Machine-Learned Evaluation Metrics	18
1.2.3 Human-Centric Evaluation Methods	19
1.3 Ethics	19

Chapter 2 Transformer-Based Language Models	23
2.1. The Transformer Architecture	23
2.2. Training and Fine-Tuning Transformer-Based Language Models	26
2.3. Conditional Text Generation with Transformer-Based LM	29
2.3.1 Language Models for Conditional Generation	29
2.3.2 Modifications to existing Language Models	31
2.4. Problems with Conditional Text Generation	34
Chapter 3 Methodology	41
3.1 Wikiplots Dataset	41
3.2. Methods	50
3.3 Evaluating the Models	57
Chapter 4 Experiment Results	59
4.1 Evaluation Metrics	59
4.2 Generated Samples	63
4.3 Analysis of the Samples	67
Chapter 5 Conclusion	73
Bibliography	75
Appendix A - Wikiplots	83
Appendix B - Additional Datasets	86
Appendix C - Extractive Summarization	88
Appendix D - Full Wikipedia Plots	94
Appendix E - Samples of Decoding Strategies	99

List of tables and figures

Figure 1. History of Natural Language Generation	7
Figure 2. Sample of Seq2Seq architecture	10
Figure 3. Base Transformer architecture	23
Figure 4. Encoder and Decoder layers	24
Figure 5. Multi-Head Attention	26
Figure 6. GPT-2 models	27
Figure 7. GROVER context example	31
Figure 8. 'Who is Lionel Messi?' Degeneration examples	34
Figure 9. Example of Greedy and Beam Search	35
Figure 10. Nucleus sampling example	36
Figure 11. Number of plots per word group for Wikiplots	42
Figure 12. Information on all datasets	42
Figure 13. Most popular tags in Wikiplots+	45
Figure 14. Number of plots per genre tags for Wikiplots+	46
Figure 15. Number of plots per word group for Wikiplots+	47
Figure 16. Number of tags per book for Wikiplots 2	49
Figure 17. Number of plots per word group for Wikiplots 2	50
Figure 18. Comparison between the different models	52
Figure 19. Results from the experiments	59
Figure 20. ROUGE results	61
Figure 21. Results using different decoding strategies	62

List of abbreviations and acronyms

BSD	Book Summary Dataset
CTRL	Conditional Transformer Language
DNN	Deep Neural Networks
FFN	Feed-Forward Network
GPT	Generative Pre-Training
IMDB	Internet Movie Database
LM	Language Model
LSTM	Long-Short Term Memory
NL	Natural Language
NLG	Natural Language Generation
NLP	Natural Language Processing
NLU	Natural Language Understanding
POINTER	Progressive Insertion based Transformer
PPLM	Plug and Play Language Model
ProGeT	Progressive Generation of Text
RNN	Recurrent Neural Networks
ROUGE	Recall-Oriented Understudy for Gisting Evaluation
Seq2Seq	Sequence-to-sequence
WMP	Wikipedia Movie Plots

Preface

Ever since I was a child, the words “Once upon a time...” would cast a magic on me. There was nothing I enjoyed more than racing home to explore new worlds through the pages of a book. Fantasy, crime, sci-fi, horror, my desire for new adventures knew no bounds, and I would raid the bookstore at every given chance. But the more I read, the more I realized that worlds could collide: the murderer was (once again) the butler, the villain was the protagonist’s long-lost father, the lover had never died. Originality dried fast, and few books were able to satisfy my hunger.

I was still wondering about my thesis subject when Professor Laurent Charlin showed us in class current methods for computer-generated text. I became awestruck, to the point of disbelief. A machine had been able to write an article far more original than some stories I had read. What if it could be modified, to write original and creative plots? How could it be transformed into a plot generator, a muse to inspire authors on writer's block? I did not have to look further for my thesis topic: it had found me.

This work was written as part of my requirements for obtaining my master’s degrees at HEC Montreal, and it has been one of the most enjoyable — yet challenging — assignments I have done. Throughout my work, I hope you will be able to notice the passion of someone who loves books and is on a quest to improve the capacities of computer-based story-generation.

As every protagonist on a long journey, I have not been alone in this challenge. First and foremost, I would like to thank Professor Laurent Charlin, not only for guiding me and supervising my work, but most importantly for believing in me and giving me the confidence to embark on this adventure. Special thanks to the people at MarketMuse, who helped me develop my programming skills and were always up for a chat on current academic research. To all the people that supported me with the funds for my studies: HEC Montreal, Fondation Boucaro, MITACS. Without their help, I would have not been able to give this project the focus and commitment that I did. To the friends I made along the way, who made a global pandemic more enjoyable.

Finally, I would like to thank everyone back home, who supported this selfish dream of mine of going to study abroad. 9,032 km away, with their own problems and a collapsing economy, and not even once did their words of encouragement falter. To them, all of my love and admiration.

Lourdes Crivelli

Introduction

I.1 - Study Context

Natural Language Processing (NLP) has been getting more attention since the publication of the first transformer model in 2017.^[38] Compared to previous recurrent-network based models, the work of Vaswani et al. had many benefits, among them the ability to be parallelized (which allows a significant decrease in training time) and the ability to handle long-term dependencies (allowing to train on longer sentences).^{[38][22]} Since then, many transformer models have been introduced, achieving state-of-the-art results in various NLP tasks including translation, summarization, etc.^[39]

Transformer models are revolutionizing Natural Language Generation (NLG), a subfield of NLP, by enabling the training of language models (LM) on unprecedentedly large amounts of text data.^[23] The latest state-of-the-art algorithm (GPT-3) has more than 175 billion parameters and is trained on over 570 GB of data.^[21] Language Models can be used to create — in an automatic and unsupervised manner— fluid and interesting text, to the point that readers can't differentiate between human and machine generated articles.^[10] While these algorithms achieve impressive results, to the point of researchers fearing releasing the models out of concerns of misuse,^[22] they are still limited when it comes to constraining, or controlling, the resulting generation. In most cases, transformer-based Language Models will only allow for a short prompt, such as a title or an opening sentence, to generate unsupervised text in a sequential way. To give an example, the Language Model GPT-2 allows the user to give the beginning of a sentence (“The nicest thing about Liechtenstein is”) and the model will produce the next sentences (“that it's the only country in the world that doesn't have a national currency.”)

In the last year, several advances have been made in creating controllable Language Models, with the objective of constraining the unsupervised generation. Following our example before, instead of simply letting GPT-2 write about Liechtenstein, we could be interested in steering the generation in other directions, such as a tourist brochure or an informative guide of countries in Europe. Among the controllable Language Models, a group focuses on conditioning the produced text on metadata (tags).^{[10][18][4]} To illustrate, GROVER (a Transformer-based LM)

gives the user the possibility to condition on the following metadata to produce an article: domain (e.g.: wikipedia.com), a date, a headline, authors and/or a body.^[10] The trained model understands the metadata and uses them to produce a text that adapts to these requirements. By doing so, the generation process can output articles that are better suited to the users demands. An example of GROVER taken from their paper^[10] with the metadata (title, date, domain) in blue:

Timing of May’s ‘festival of Britain’ risks Irish anger April 13, 2019 theguardian.com The prime minister’s decision to launch her much-hyped “great Britain” conference on Monday – the same day as a Lisbon treaty event paving the way for Brexit-free member states to leave the European Union – will be seen as provocative by some, according to senior Tories.

The tags selected to train the controllable Language Model depend on the end-goal and are quite varied. Some common tags include domain, topic, date, keywords, entities and their relationships, etc. ^{[4][10]} By adding these metadata during the training (or fine-tuning), we expect the language model to learn underlying relationships between the target text and the tags. Then, when using the model to generate text, the user would give the metadata as part of the input sequence (prompt) and the model would condition the output based on the relationships it has learnt. ^[4]

Despite all the advancements, the effect of each metadatum in the final output is not always clear. Though some metadatum have visible effects (the writing style of Wikipedia is not the same as Reddit) others have a less clear impact (did it write a murder mystery because of the title “death cloud” or because of the outline “he found his friend dead”?). This is not trivial when we consider there are many costs associated with metadata. To name a few, there is a cost in acquisition (some tags are more difficult to obtain than others), a cost in re-training the language models (depending on the base language model, a new metadata tag might require architectural modifications and training from zero) as well as an opportunity cost (most LM have input constraints, adding more metadata reduces the available space for our target text). As such, it is important to understand how each type of metadata helps in conditioning the generation, to determine which ones add the most value at a lower cost.

I.2 Problem & Research Question

Our aim is to understand the role of metadata for conditioning story generation. In particular, when fine-tuning a language model to write a story, we want to know which tags will enable us to successfully guide the output.

We build on previous studies done by PlotMachines^[18] and Martin et al^[68] to assert the impact of different tags on a Language Model’s generation. We use and study the effect of the following metadata in conditioning a language model to write a story: title, genre (drama, crime, fiction, etc.), discourse (introduction, body, conclusion), plot outline and the previous paragraph. We use both automatic evaluation metrics and qualitative human perspectives to assess their capacity in guiding the produced text. By doing so, we aim at answering the following research questions: What metadata (tags) should be used to condition creative-story generation? How do we evaluate their contribution? Does the language model benefit from longer prompts, or would it hinder its generation capacity? These are the questions we aim at answering with our work.

Our work starts with an individual study of each of the tags and their effect on the perplexity of a language model. We use the publicly available language model GPT-2, whose generation capabilities revolutionized the field of language modelling. After determining the effect of each metadata-type by itself we combine them, to deepen our understanding of how they can work together to produce a conditioned story. By doing so we hope to determine the cost-benefit of each model in terms of automatic metrics and produced output. We analyze the results using automatic evaluation scores (ROUGE, Self-BLEU, perplexity) as well as human evaluations.

I.3 Contributions to the field

Our contributions to the field of Natural Language Generation are the following:

- 1) A curated Wikipedia plot summary dataset (Wikiplots 2) with the additional genre information. In this dataset, we have taken different precautions to limit title repetitions. This is done so that the different sets (training, validation, and testing) can be independent, reducing the risk of having a title in two sets. This will make the evaluation metrics fair, as the model has no access to the evaluation data.

Further information on this dataset, and the importance of a fair evaluation, are presented in Chapter 2.

- 2) A quantitative study on the effect of different types of metadata on generation from the GPT-2 model, both at an individual level and by combining different metadata.
- 3) A qualitative study of the generations. Unlike other ablation studies, where only the automatic metrics are presented, we show and evaluate the generated samples to determine the impact of the metadata. We firmly believe that a qualitative approach is needed as automatic metrics lack the capacity to assert the effects of the conditioning.

Additionally, we discuss the current state of conditional LM and raise awareness about the importance of automatic evaluation metrics, both for model fine-tuning and for fair comparison between models.

I.4 Dissertation Flow

We start with a literature review in Chapter 1. We, briefly, discuss the evolution of the field of Natural Language Generation, with a focus on the developments that lead to the Transformer-based Language Models (LMs). We also introduce the different evaluation metrics as well as ethical issues raised by making powerful Language Models publicly available.

In Chapter 2 we describe the Transformer architecture and explain how LMs are trained and fine-tuned for task-specific activities. We present conditional-based language models and their limitations. Chapter 3 focuses on our research methods and presents both our models and our dataset, Wikiplots 2. We show our results —along with some sample generated text— in Chapter 4.

Chapter 1

Literature Review

In this chapter we introduce the field of Natural Language Generation (NLG) and how it has advanced throughout the years. In Section 1.1 we go over the history of NLG, focusing on how story generation has evolved from the first rule-based methods to the current transformer-based language models. We describe the most common evaluation metrics for NLG in Section 1.2 and present the unresolved problem of automatic text evaluation for creative tasks. Finally, we discuss in 1.3 the ethical stakes involved with advancing NLG models, and how they can be potentially misused. More details on the models and their architectures will be provided in Chapter 2.

1.1 - History of NLG

Advances in the field of NLG have been driven by the desire to tackle challenging tasks such as summarization, machine translation and conversation systems. While creative writing has not been the main focus of past research, the models designed for other tasks were able to be adapted to it. In this section we talk about the advances in NLG as a whole and link their progress to our target task of creative generation and language modeling.

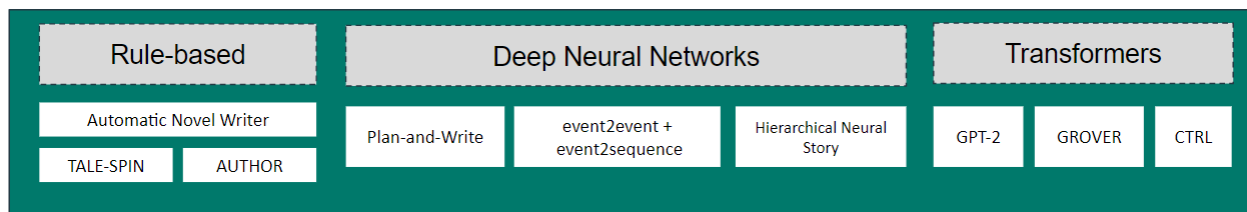


Figure 1. History of Natural Language Generation

1.1.1 - Rule-based Methods and Statistical Language Generation

Natural Language Generation is not a new academic field. Studies on automatic creative writing date back to 1970s^[62] with Automatic Novel writer^[66] and TALE-SPIN.^[65] As most NLP methods of that time, the first approaches are mostly rule-based simulations.

Automatic Novel writer is a rule-based method capable of producing a mystery novel in seconds. Limited to one setting (a weekend party murder), the user gives the system the characters

and their relationships and the algorithm develops the story, deciding who are the murder and victim (based on their characteristics and randomness), the motive (fear, greed, anger, and jealousy) and the events. Differences between executions is sparse, limited to the characters, who committed the murder, the motive, etc.^{[66][67]} TALE-SPIN is another example of a rule-based method, but allows more customization by giving the user the option to set for each character (always forest animals) a goal (e.g.: eat), a personality (kindness, vanity, honesty and intelligence), a relationship (affection, trust, deceit, etc.) among other traits.^[65] As such, characters do sub-tasks to fulfill their goals (e.g.: go find food) and based on their personalities and relationships they trigger events.^[65] Some example of TALE-SPIN:

“Once upon a time, there were two bears named Jack and Joe, and a bee named Sam. Jack was very friendly with Sam but very competitive with Joe, who was a dishonest bear. One day, Jack was hungry. He knew that Sam bee had some honey and that he might persuade Sam to give him some...”^[65]

“Once upon a time, there was a dishonest fox named Henry who lived in a cave, and a vain and trusting crow named Joe who lived in an elm tree. Joe had gotten a piece of cheese and was holding it in his mouth. One day, Henry walked from his cave, across the meadow to the elm tree. He saw Joe Crow and the cheese and became hungry...”^[65]

AUTHOR^[69] is one of the first A.I. papers to consider the creative writing process: in particular, how the creator (author) plans the story. In the paper, Dehn states that one should consider a story as an event-based generation, instead of a world simulation (like TALE-SPIN). In an event-based generation, an author has a series of events planned and the story follows this reasoning, creating sub-events to link them.^[69] To achieve this, Dehn proposes adding a memory to A.I models, so they can recall previous characters, events and relationships.^[69]

Many other creative writing simulations were proposed: UNIVERSE^[70] (can generate coherent characters for a TV Soap Opera), MINSTREL^[71] (as AUTHOR, a goal-oriented generation that writes stories of the knights of Camelot), MEXICA^[72] (a two-step approach capable of producing a Mexican folklore tale and evaluating coherence and novelty of the generation), Virtual Storyteller^[73] (uses both the script and character-based approaches for generation) among others.

Rule-based methods show multiple limitations in producing stories as 1) they require the researcher to be knowledgeable on both computer sciences and linguistics.^[100] 2) It is not possible to understand human language based only on rules and exceptions.^[100] 3) the results are always based one genre or topic (a murder mystery, animal stories, Camelot tales, etc) and are unable to generalize.

To solve these limitations, researchers took inspiration from statistics, trying to devise algorithms capable of modeling human language.^[100] Language Modeling is a task of NLP which aims at understanding the probabilistic distribution of words in a sentence in order to predict the next word.^[95] One of the first statistical models is the n-gram model, which uses maximum likelihood to predict the next word. For example, to estimate the probability of “day” following “beautiful” it counts all bigrams (“beautiful day”) in the corpus, and divides it by all the bigrams that start with “beautiful”.^[96] While easy to train, n-gram cannot handle long-term dependencies (as n increases, so does the number of parameters and combinations become sparse) and relies heavily on the corpus used (“beautiful day” could be missing from the corpus, giving a probability of 0) making them unable to generalize.^[100]

1.1.2 - Deep Neural Networks

Deep Neural Networks (DNNs) address the issues raised by rule-based and statistical methods. DNN models can now learn syntactic dependencies^[40] and model different varying-length inputs.^[97] Because of their high-dimensional hidden states with non-linear dynamics, Recurrent Neural Networks (RNN) are able to recall previous information and are effective with sequence based prediction.^[100] Parameters in RNN are optimized by using gradient descent and back-propagation algorithms, which brings the problem of exploding/vanishing gradients.^[41] Different methods and models have been designed to solve these complications, including gradient clipping, dynamic weight initialization and the Long-Short Term Memory (LSTMs) model, capable of forgetting previous stored information or adding new information to the memory unit.^[42] Despite these advances, these methods^[42] require a target and input of fixed length and are complicated to train, limiting their applications.^[41]

In 2014 Google presents Seq2Seq, a multilayer LSTM where the input sequence length differs from the generation sequence length.^[56] The main motivation is to improve machine translation,

as translated sentences do not necessarily have the same number of words as the original sentence (“speed limit.” translates to “Geschwindigkeitsbegrenzung.” in German).^[56] To achieve this, Seq2Seq first encodes the original sequence into a vector of fixed-length so that later the decoder can use this internal representation to generate the translation.^{[56][57]} An example is given in the figure below.^[93]

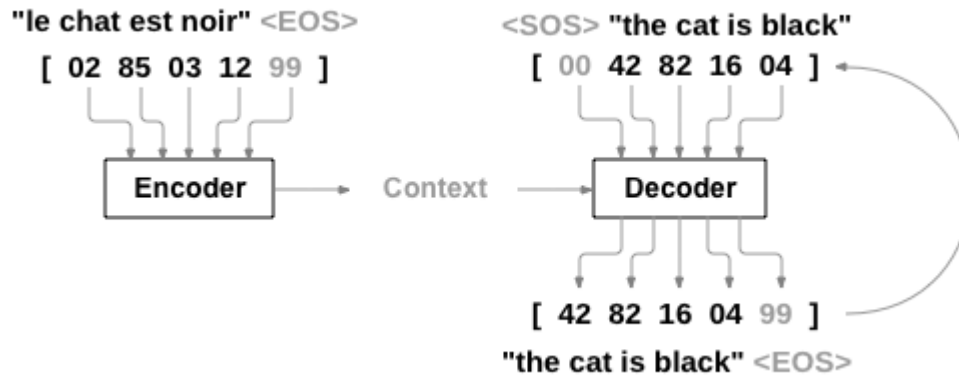


Figure 2. Sample of Seq2Seq architecture

The encoding to a fixed-length vector quickly becomes the bottleneck, and the model’s performance decreases when using long sentences.^[57] To lift these limitations, researchers applied a known technique of computer vision and pattern recognition into NLG: the Attention Mechanism.^[57] Instead of encoding all the input into one context-vector and feeding it to the decoder, attention allows the decoder to select at each decoding time-stamp what words of the context it will attend to. In other words, at the i -th time, it will only focus on the relevant words to the i -th word. As such, it creates a distinct context vector for each target word, giving different weights to words in the input sentence.^{[57][58][59]} We explain Attention in more detail in Chapter 2.

These new models gave rise to a new wave of creative writing algorithms, such as the seq2seq event generation from Martin et al.^[68] and the more recent Plan-and-Write.^[63] Using Wikipedia Movie Plots, Martin et al. propose an event generation model consisting of two seq2seq networks: first an event2event network that encodes the event and predicts the following action, and second an event2sentence network which translates the previous (not human readable) output into human readable sentences.^[68] To represent events, event2event encodes the sentence into a tuple, a combination of subject (s), verb (v), object (o) and a modifier (o). For example, given the original sentence “He thinks he’s on a top-secret spying mission”, event2event extracts the event

(“he think 0 0”) and predicts the following event (“she come know truth”) and finally event2sentence transforms the event into a coherent sentence: “She has come to the truth”.[68]

In addition, they explore giving —or removing— additional information to the event, such as including genre (based on topic-modeling, 100 different genres), abstracting the subjects and locations (<NE> instead of a name and <NER> for a location), numbering entities, etc. [68] Their work brings some interesting insights to the effect of additional metadata in creative generation. Starting from a baseline of 704.82 sentence-level perplexity they first manage to reduce it to 54.23 by generalizing nouns, verbs and replacing named entities and later lower it to 48.04 by adding a genre tag.[68] While not present in their research, it would have been interesting to know the effect of the genre tag before generalizing the input sequence.

Like the event generation algorithm, Plan-and-Write is also a two-step approach: given a title as input it first generates a storyline (using Seq2Seq) and then uses the storyline to create a story (with another Seq2Seq model).[63] To train the storyline generation, they extract from a corpus (ROCStories) the main keywords using the RAKE algorithm^[94], which uses graph and word frequency metrics to determine the important words in a passage.[63] In their paper, they study two possible ways of using a storyline: static (produced before creating the story) or dynamic (at each step, both the storyline and the next sentence in the story are created).[63] Plan-and-write demonstrates that a static storyline is superior to a dynamic one when looking at coherence, diversity and on-topic generation.[63] An example from Plan-and-Write:

Input: [The Virus]. Static Storyline: [work fix called found day] Story: “I had a virus on my computer. I tried to fix it but it wouldn’t work. I called the repair company.They came and found the virus. The next day, my computer was fixed.”

Other DNN models for creative text generation include Fan et al seq2seq base model [64] (uses a three step approach: given a prompt it creates an action sequence, then a story narrative — with anonymized entities— and finally a full story with entities), Scheherazade^[74] (which uses topic based plot graphs to learn relationships between events), Hierarchical Neural Story Generation^[2], etc.

1.1.3 - Transformers

Three years after the introduction of Seq2Seq, Google presents the Transformer.^[38] Building on previous research of attention and NLG, the team proposes a new model, focusing exclusively on the attention mechanism and not using it as a simple addition to a RNN.^[38] Because of their relevance to our work, we briefly explain the Transformer architecture in Chapter 2.

Transformers achieved state-of-the-art results in machine translation^[38] and researchers wasted no time in modifying them to various NLG tasks such as creating a document from summaries^[3], question answering^[61] and summarization^[27].

Removing the sequential nature of recurrent networks allows Transformers to be parallelized, greatly reducing training time.^[38] This made previous unfeasible tasks achievable, in particular, the idea of building models capable of understanding human language.^[60] High-capacity language models (LMs) can be built by training transformer models on large text corpora, allowing them to encode facts and commonsense knowledge.^{[9][59]} This induced a paradigm shift towards Transfer Learning^{[9][60]}: instead of building task-specific architectures, massive LM are first pre-trained to understand linguistic properties and later fine-tuned to specific tasks showing impressive results.^{[9][59]}

What make LM superior to other types of knowledge encoding —among other benefits— is the fact that they can be trained on unsupervised data, which is abundant compared to label data.^{[60][59]} This is possible as LM are trained to optimize word prediction (could be the next one or a specific position) based on context information.^{[7][59]}

LMs have been exponentially growing in the last years, not only in the amount made available to the general public but also in the number of parameters that they possess. A year ago, a state-of-the-art model could have 1.5 billion parameters (GPT-2^[23]) while nowadays we are talking about 175 billion parameters (GPT-3^[21]). The scale of these new models will undoubtedly bring forward another paradigm shift, as it is currently impossible to fine-tune such large models.^[21] We look at LMs in more detail in the following section due to their importance for this work.

1.1.4 – Transformer-Based Language Models

In this section we take a closer look into current transformer-based language models. In less than 2 years a wide variety of models have emerged, each achieving state-of-the-art results on different tasks. Just to name a few in chronological order: GPT [22], BERT [61], GPT-2 [23], Transformer XL [30], XLNet [25], ROBERTA [24], GPT-3 [21], BART [39]. This rapid progress is not without consequences. As we explore in future sections, the wide variety of techniques used as well as the different downstream tasks in which they are employed and the different datasets for training, makes comparing these language models difficult.^[60] In this section we go over the main characteristics of some of the most popular Language Models. We look into their architecture in more detail in Chapter 2, along with LMs that have been specifically altered for conditional text generation.

GPT [22]: (June 11, 2018) One of the first models to demonstrate the benefits of Transfer Learning. Generative Pre-Training (GPT) by OpenAI uses a multi-layer Transformer decoder architecture and the BookCorpus dataset to train a language model.^[22] In particular, the model tries to maximize the following conditional probability:

$$L_1(\mathcal{U}) = \sum_i \log P(u_i | u_{i-k}, \dots, u_{i-1}; \Theta)$$

with Θ the model parameters, u_1, \dots, u_n the corpus tokens and k the context window.^[22] In simpler terms, it maximizes the probability of token u_i being the next token given the previous tokens and the model parameters.

For tasks with particular input sequences (question answering, textual entailment) there is no need to change the model architecture and only slight modifications to the input data are required.^[22] After training, the LM was tested on multiple-tasks, defeating task-specific models for commonsense reasoning, and surpassing all submissions on the GLUE benchmark, among others.^[22] GPT has 110 million parameters.^[61]

BERT^[61]: (October 11, 2018) Unlike GPT, which is unidirectional (the model has a left-to-right architecture) BERT (Bidirectional Encoder Representations from Transformers) uses both

the left and right context (bidirectional) to train a transformer-based language model. For example, when we input the phrase “I am better than GPT” in both models, to predict “better” GPT will only see the previous tokens (left-to-right “I am”) while BERT will additionally see the following tokens (right-to-left “than GPT”). To be more precise, instead of predicting the next word, BERT masks a random word from the input sequence and tries to predict it. (In our example, it would be “I am [MASK] than GPT”). Masking is done to prevent the model from reading the answer. BERT is also trained on next sentence prediction by adding a separator token [SEP] between two sentences and giving an additional label indicating if the second sentence is a continuation from the first one. This is useful for downstream tasks such as question answering, where the models need to understand the relationship between sentences.^[61] BERT is trained on BooksCorpus and English Wikipedia. As GPT, when BERT was introduced it was also able to beat state-of-the-art results in numerous downstream tasks, including the GLUE benchmark and SQuAD, among others. The vanilla implementation of BERT (base) has 110 million parameters while the larger version has 340 million parameters.^[61]

GPT-2 (February 14, 2019): As GPT, GPT-2 is also trained to predict the next token (word) but with 10 times more data and parameters: GPT-2 used 40GB of Internet data (8 million web pages dataset, WebText) and has more than 1.5 billion parameters.^[23] In their paper, OpenAI demonstrate that — due to the volume of data used during training— the model has learnt how to solve tasks in an unsupervised manner, with no need to change the model architecture nor to train it on a task-specific dataset. Under this “zero-shot” setting, GPT-2 managed to achieve state-of-the-art-results in various language modeling tasks, including machine reasoning (Winograd Schema Challenge) and modeling long-range dependencies (LAMBADA dataset).^[23]

GPT-2 also shows incredible progress when it comes to automatic text generation. Given a prompt, the model can generate coherent, well-written and engaging text. A sample of GPT-2, with the prompt in blue:

In a shocking finding, scientist discovered a herd of unicorns living in a remote, previously unexplored valley, in the Andes Mountains. Even more surprising to the researchers was the fact that the unicorns spoke perfect English.

The scientist named the population, after their distinctive horn, Ovid's Unicorn. These four-horned, silver-white unicorns were previously unknown to science.

Now, after almost two centuries, the mystery of what sparked this odd phenomenon is finally solved.

Dr. Jorge Pérez, an evolutionary biologist from the University of La Paz, and several companions, were exploring the Andes Mountains when they found a small valley, with no other animals or humans. Pérez noticed that the valley had what appeared to be a natural fountain, surrounded by two peaks of rock and silver snow.^[23]

Their advances brought new implications, in particular, the dangers of releasing this powerful language model to the public.^[52] We go into more details at the end of this chapter, in the Ethics section.

TransformerXL: (July 2019) While enabling the training of larger LM, Transformers are unable to model long inputs. When training on large datasets, the data has to be split in smaller segments, losing context information, causing inefficient optimization and reducing performance.^[30] TransformerXL proposes to add recurrence to the transformer architecture: by keeping the hidden states from the previous segment it builds a memory, adding previous context to the new segment.^[30] In addition, the model requires relative positional encoding to prevent tokens from two different segments from having the same positional encoding.^[78] We discuss positional encoding in Chapter 2, when we present the transformer architecture.

GPT-3 (June 11,2020). The latest model from OpenAI, trained on 570 GB of data and boasting an impressive 175 billion parameters, surpassing all previous language models in terms of parameters.^[21] As with GPT-2, GPT-3 is able to perform on new-untrained tasks. In their paper, OpenAI tested the model on three types of methods: zero-shot (no example is given. Example: “Translate English to Spanish”), one-shot (one example is provided. Example: “Translate English to Spanish: Bye => Adios”) and few-shot (a couple of demonstrations are provided).^[21] In no case is the model fine-tuned on additional data (the weights remain unchanged and are the same for all tasks).^[21] As noted in the examples, the tasks are described as sentences (“translate English to Spanish”). GPT-3 shows promising results, achieving state-of-the-art results (or closing in to models particularly fine-tuned for specific tasks) in various NL tasks.^[21] Currently, access to GPT-3 is not available to the general public, and its fine-tuning performance is yet to be determined.

1.2 - Evaluation Methods

While the generation algorithms have seen many improvements in the last years, we can't say the same about the evaluation methods.^[17] The golden standard for evaluating NLG models is still human evaluation, which is costly, time-consuming and not always reproducible.^[15] Thus, improvement of NLG methods is hindered by a lack of good automatic evaluation metrics.^[17] In this section, we briefly look into the different types of evaluation methods and go deeper into the main ones we use in our models.

When it comes to evaluating the text generated by NLG, we can separate the metrics into three groups ^[15]: (1) Untrained Automatic Evaluation Metrics, (2) Machine-Learned Evaluation Metrics and (3) Human-Centric Evaluation Methods. ^[15]

1.2.1 - Untrained Automatic Evaluation Metrics

Untrained Automatic Evaluation Metrics refers to the group of methods that are able to compare the similarity between the generated text and the original document (gold standard) in the form of a numeric score. These methods tend to be fast and efficient, allowing to compare different models for the same task and/or measure progress of a same model with varying hyperparameters.^[15] An example is the ROUGE (Recall-Oriented Understudy for Gisting Evaluation) score.^[43] The most popular type of ROUGE is ROUGE-N, which measures the overlap of n-grams units between the original and generated texts.^[43] In other words, how word-by-word similar both texts are. As many models presented in the previous section, ROUGE is a metric originally designed for summarization but was later adopted by other NLG tasks such as question-answering and short story generation.^[15]

If we have the pair of sentences:

<i>Original:</i>	<i>Mama just killed a man</i>	<i>The future will be shaped by NLG</i>
<i>Generated:</i>	<i>just killed a man Mama</i>	<i>NLG will shape our future</i>

ROUGE-1 (measuring unigrams) will give a perfect score (1) to the first pair while giving a significant lower score to the second pair (0,43). If we take ROUGE-2 (bigrams) the first pair gets a score of 0.75 while the second pair gets the lowest possible value (0). Any human can tell

that the second pair of sentences are almost the same, but for ROUGE-2, no two bigrams (“NLG will”, “will shape”, “shape our”, “our future”) match the original sentence bigrams.

Another automatic metric worth introducing is BLEU. BLEU is similar to ROUGE, in the sense that it also compares how many n-grams overlap. The key difference is that BLEU measures how many words in the generated text appear in the original document while ROUGE compares how many words in the original document appear in the generated text. In other words, BLEU accentuates precision while ROUGE focuses on recall.^[45]

BLEU is the basis for Self-BLEU, a metric used to measure lexical diversity.^[15] Self-BLEU keeps one of the generated sentences as a hypothesis and calculates how similar it is to the other generated ones (the references) using BLEU. The final score is the average of all calculated BLEU scores. By doing so, we compare how similar (or dissimilar) the generated sequences are. The higher the Self-BLEU score, the less diversity in the generation.^[44] Self-BLEU is a computationally expensive metric, so it tends to be calculated on a subsample of the test set.^{[1][14]}

Let us give the following sets of sentences:

Sample 1

I need some holidays

I can not wait for the holidays

Next holidays can't come fast enough

Sample 2

I would like to go to Paris

I would like to go to Argentina

I would like to go to Canada

For the first group, the average Self-BLEU 4 score (four-gram) is 0,04 (high diversity) while for the second group it is 0.81 (little diversity).

The final automatic metric we would like to introduce is perplexity. Perplexity measures the quality of the language model, and how confident it is in predicting test data.^[16]

$$\text{perplexity} = \prod_{t=1}^T \left(\frac{1}{P_{LM}(\mathbf{x}^{(t+1)} | \mathbf{x}^{(t)}, \dots, \mathbf{x}^{(1)})} \right)^{1/T}$$

Equation for perplexity, with T the number of words in the corpus, P_{LM} the probability for our language model of predicting the next word ($t+1$) given the previous words ($1, \dots, t$).^[96]

In simpler terms, perplexity measures how syntactically correct our generated sentences are. As such, the higher the value of P_{LM} the more confident it is in the predictions. We normalize by T (Vocabulary) to get the probability per word. Since we are using an inverse probability, the lower the final value, the better.^[98] Most language models are trained to reduce perplexity.

While Perplexity allows us to see how well the model fits the data it does not correlate with good performance at the desired task.^{[13][16]} As such, we can have a model with low perplexity that creates bland generic responses that won't satisfy the users.^[15]

1.2.2 - Machine-Learned Evaluation Metrics

The metrics described in the section above rely on the assumption that the target task involves generating text that will overlap with the golden standard, as is the case with machine summarization or translation, but this is not the case for open-ended generations like short story writing.^[2] Since most NLG models are evaluated by humans, a couple of metrics have been designed to mimic human judgment by using machine-learning models.^[15]

Machine-Learned evaluation metrics are not simple to build. For example, to train ADEM (Automatic Dialogue Evaluation Model) the researchers first built a dataset of human responses to tweets and later trained a neural model (in their case a hierarchical RNN) to predict the human response.^[46] This is not only computationally and time expensive, but it is also task specific, making it unable to generalize.^[47] Should one wish to use it for another type of dialogue, one would need to replicate their experiment.

Given the popularity of transformer models, academia has also studied how to apply them to evaluation. In particular BERT has been the root for many models, including ROBERTA-STS (to detect unrelated, illogical or contradictory sentences)^[48] and BLEURT (based on BLEU score,

this BERT model uses human annotated and synthetic data to train an evaluator capable of better predicting similarity between sentences).^{[17][15]} While useful, machine-learned metrics are still task-specific.^[47] As such, for our open-ended generation, we won't be using any machine-learned metric. It is also worth mentioning that, as any metric that requires human judgment, they depend heavily on the correct definition of the human evaluator task and can be prone to biases.^[47]

1.2.3 - Human-Centric Evaluation Methods

As we stated at the beginning of this section, Human-Centric Evaluation Methods are the main form of evaluation in NLG papers.^[15] Regardless of their popularity, these methods are vulnerable to errors. We list shortly some of the main problems with using human evaluation:

- Humans are good at measuring the quality of a text but fail to realize plagiarism (generation diversity).^[49]
- Most models lack details on how the human-evaluation task was defined, preventing reproducibility and evaluating possible biases. This is not a minor detail, as noted by the authors of ADEM, who realized their human evaluators would generate text that satisfied the minimum requirement and had no incentive to give longer answers.^[47]
- When the evaluation metrics are present (for example in PlotMachines^[18]), they tend to be subjective and rely on the good-will of the evaluator to consciously select the right answer.

Regardless of all these drawbacks, Human evaluation is still the main metric and will continue to be until a good automatic metric that correlates highly with human judgement is presented.

1.3 - Ethics

We would like to finish the Literature Review chapter by discussing some challenges NLG faces regarding ethics, both in use of LM and in generation bias. While our task is mainly the study of tags and their effect on conditioning the generation, we can not ignore some problems present in current language models and the consequences for society of their usage.

When GPT-2 was first introduced, the researchers expressed concerns about making the language model available to the public out of fear of its misuse.^[52] Their worries were not without ground,

as language models can be used in the generation of Fake-news^[10], automation of phishing or fake content^[52], fraudulent academic essay writing^[21], etc. What makes Language Models dangerous is the fact that both humans and machines have a hard time detecting if they are reading a generated text.^{[52][10][21]} While latest advances can mitigate the harm of these models (GROVER is capable of producing fluid text *and* detecting if text was generated by another GROVER model^[10]) the danger is still real and should be a concern for researchers.^[21]

Recently, attention has been drawn to the issue that language models have learnt harmful stereotypes^[50] and biases^[51], generating prejudiced texts.^[21] This is not surprising, as the models have been trained on vast amounts of internet collected data. Even when carefully selecting the sources, such as Wikipedia^{[4][18]} or Reddit^{[2][18]} one cannot guarantee that the models will be free of toxicity, racism or bias. This is by no means done intentionally. For example, the Wikiplots dataset^[28], which contains plot descriptions of books and movies, might have innate bias regarding the role of women in a family — mainly because it aggregates stories from different periods of time, when values were different. A priori, there is no way for us to know about it, unless we do an extensive reading of the plots (unfeasible considering the amount of data that is used to train these models) or limit the dataset to books and movies published in the last years (which might not be enough to fine-tune the model). While still a challenge for academia, researchers are generating awareness^[21] and have begun developing tools to assert the bias in language models^{[50][51]} and tasks to evaluate bias in new models.^[50]

Last but not least, we would like to address the concerns for AI models replacing humans in the workforce. Natural Language Generation is not an exception. In particular, journalism has been greatly affected by advances in data-to-text NLG models.^[53] Data-to-text models take as input structured data and output a fluid text containing the key information, greatly reducing the time and the need for human inputs.^[100] Major information sites as The Washington Post^[55] and the Los Angeles Times^[54] are using artificial intelligence to assist in the creation of articles, allowing them to reduce the speed and increase the coverage of news.^[55] At the moment, we believe long open-ended text generation will still be a human task, as current language models (among other problems) are not easily constrained and produce text that is not-factually correct. As NLG models become more powerful and more fluid, the possibility of them replacing humans in long-text generation tasks such as content creation, writing and journalism becomes more real.

Still, we would like to conclude this chapter in a positive note, stating that language models have still to show the creativity humans are capable of.

Chapter 2

Transformer-Based Language Models

In the previous chapter we showed how advances in the field of Natural Language Processing led to the creation of the Transformer and its subsequent applications in building language models. In this chapter, we explain how the Transformer model works (section 2.1) and how Language Models are trained and fine-tuned for task-specific activities (section 2.2). In Section 2.3 we present the Language Models specifically designed for the tasks of conditional text generation and in section 2.4 we highlight the main problems they have, along with possible solutions.

2.1. The Transformer Architecture

The Transformer model is built on an encoder-decoder architecture: the input sequence is encoded to a fixed-length representation and is fed to the decoder, which extracts from this internal representation the output words.^[79] Instead of using one encoder and one decoder layer, the base Transformer uses 6 of each^[38] (See figure 3, image on the left).

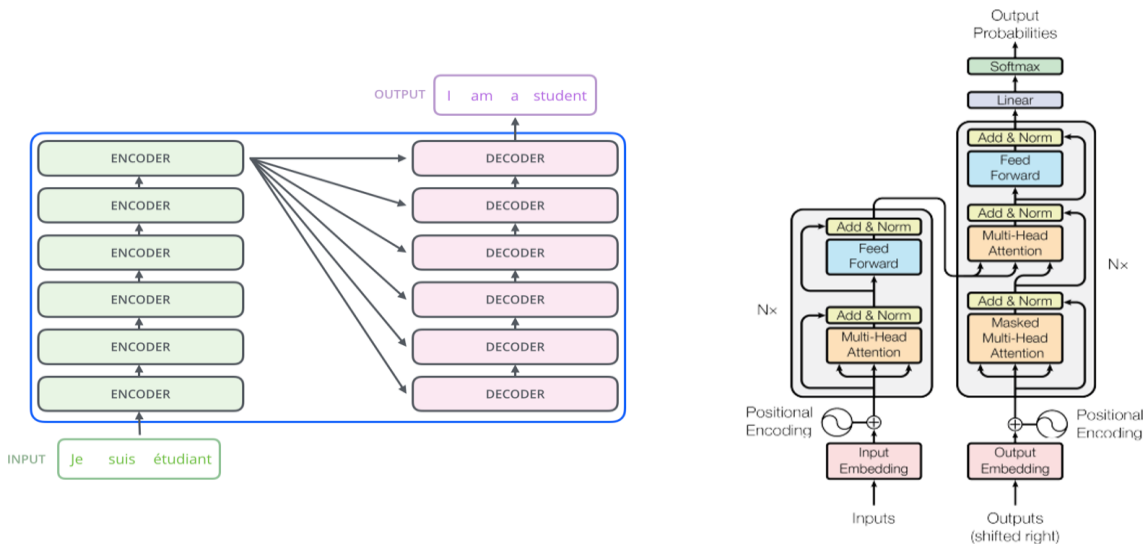


Figure 3. Base Transformer architecture

Left: A simplified representation of the base Transformer architecture^[75] showing all 6 layers for the encoder and decoder. Right: The original paper representation^[38] showing how each layer in the encoder (first column) and decoder (second column) are composed.

Each encoder layer is identical and is composed of two sub-layers (see figure 4): a multi-head self-attention layer (we go into more details later) and a feed-forward neural network. The output of each sub-layer is computed as: $LayerNorm(x + Sublayer(x))$ with $LayerNorm$ being a layer normalization function. All of the 6 layers in the decoder are composed of a multi-head self-attention sublayer, another multi-head attention (for the encoder output) and the feed-forward neural network.^[38]

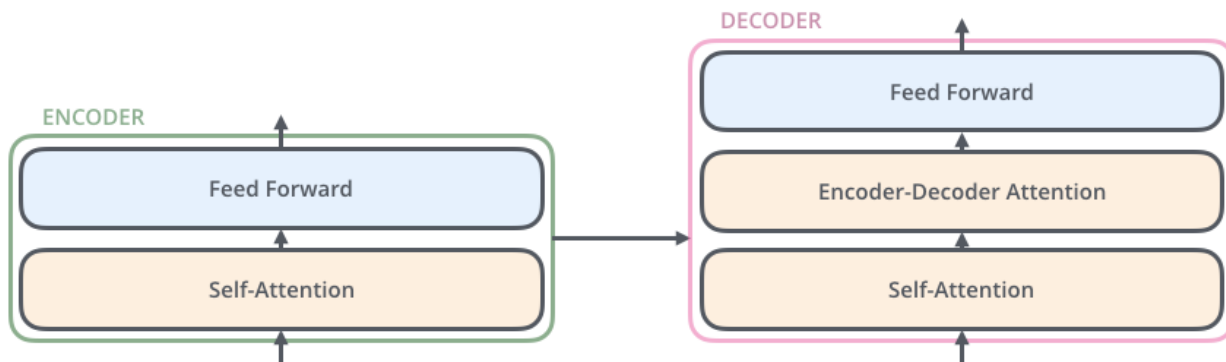


Figure 4. Encoder and Decoder layers

Figure extracted from “The Illustrated Transformer” guide^[75]

The feed-forward network (FFN) is the same for both the encoder and the decoder with two linear transformations and a ReLU activation. The transformations are the same for all layers; only the parameters vary.^[38]

$$FFN(x) = \max(0, xW_1 + b_1)W_2 + b_2$$

We look at the process in more detail. The model first calculates the word embeddings of a given sentence (a vector representation of the word. Similar words have similar representations^[76]). Since a word can have a different meaning depending on its position (Take for example “it” in the following sentences: “Tuesday was a horrible day. It was raining”, “The dog barked because

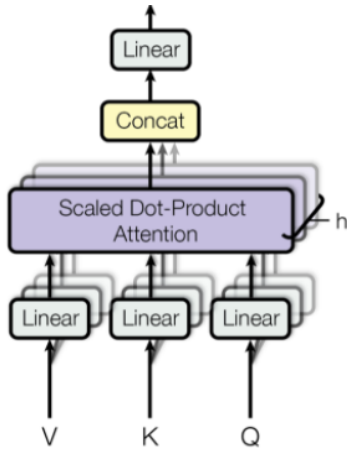
it was happy”), a positional encoding is added to the word embedding (there are multiple-ways of doing so, in the original paper the authors propose using a sin and cosine function^[38]). Since each word now has information on its context, there is no need to pass words in a recurrent manner. (In RNNs, at each timestamp we pass one word, and its hidden state depends on the previous word’s hidden state). Instead, each word can follow its own path, allowing the Transformer to parallelize the input sequence.^{[38][75][77]}

Next, the encoder receives the embeddings and passes them through a multi-head self-attention layer. Following the example provided before (“The dog barked because it was happy”), the self-attention layer allows the model to relate “it” with “dog”. To do so, it uses a query, a key and value concept. In our example, “it” would represent the search query. The model then looks at all the other words (the keys) to see how related they are. The degree to which the query is related to the other keys is calculated by the dot product between the two vectors (a positive inner product indicates that the words are similar). This is normalized using a softmax function so that all scores add up to 1, so that the positions with the highest values are more related to our query. Finally, we multiply the softmax result with the original information we had for each key (the value). The output of the self-attention layer is the sum of the resulting weighted vector.^[75]

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

with Q : a matrix of queries, K a matrix of keys, V a matrix of the values, d_k the dimension of the keys. In the original paper the authors add a scaling factor ($1/\sqrt{d_k}$) as to have more stable gradients.^[38]

The Transformer model does not calculate one attention function. Instead, it calculates many of them in parallel (hence, the Multi in Multi-Head Self-Attention, see equation below) applying different linear transformations to the queries, keys and values. By doing so, the model is able to attend to different information of the input sentence, learning distinct representations. In the original paper, they propose using 8 heads, each initialized to a different random state. These representations are then concatenated, and a linear transformation is applied to reduce the dimensions to those expected in the Feed-Forward network.^{[38][75]}



$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \dots, \text{head}_h)W^O$$

$$\text{where head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

Where the projections are parameter matrices $W_i^Q \in \mathbb{R}^{d_{\text{model}} \times d_k}$, $W_i^K \in \mathbb{R}^{d_{\text{model}} \times d_k}$, $W_i^V \in \mathbb{R}^{d_{\text{model}} \times d_v}$ and $W^O \in \mathbb{R}^{hd_v \times d_{\text{model}}}$.

Figure 5. Multi-Head Attention

Equation and graph taken from the original paper “Attention is All You Need”.^[38]

Finally, the decoder takes as input the internal representations of the encoder and outputs words until a special token (indicating end of the sentence) is predicted. To be more specific, at each time-step, the decoder uses both the encoder output and the previous generated words to output a vector of floats. To turn this numeric representation into a word, both a linear and a softmax layer are needed (see figure 3) to calculate the probability of each word in the vocabulary of being the next word (and normalizing the probabilities so that they add up to 1). The highest value is selected as the next token and is given back to the decoder so it can keep predicting the next words. Unlike the encoder, in which the self-attention can see all words in the sequence, the decoder can only attend to previous generated tokens in the output sequence.^[75]

2.2. Training and Fine-Tuning Transformer-Based Language Models

Now that we have a better understanding on how the base Transformer works, we can go deeper into how Language Models are trained and fine-tuned for specific tasks. We look at the architecture of GPT-2, the model we use for our experiments.

GPT-2 is a decoder-only Transformer-based language model: there is no encoding layer, and the decoding block is composed of only two-sublayers (the Feed Forward NN and only one self-attention layer). While the base Transformer uses 6 decoding layers, GPT-2 takes this a step further, by greatly increasing the number of stacked layers in the architecture. The smallest model (GPT-2 small) uses 12 layers (same number as GPT) while the largest model (GPT-2 extra-large)

has 48 decoding layers (see figure 6).^[23] As stated in the previous section, GPT-2 is trained on 8 million web pages (WebText dataset) to maximize the likelihood of a word appearing giving the previous context.^{[22][23]} The result is a Language Model capable of producing coherent and human-like text, and outperforming task-specific models in multiple benchmarks.^[23]

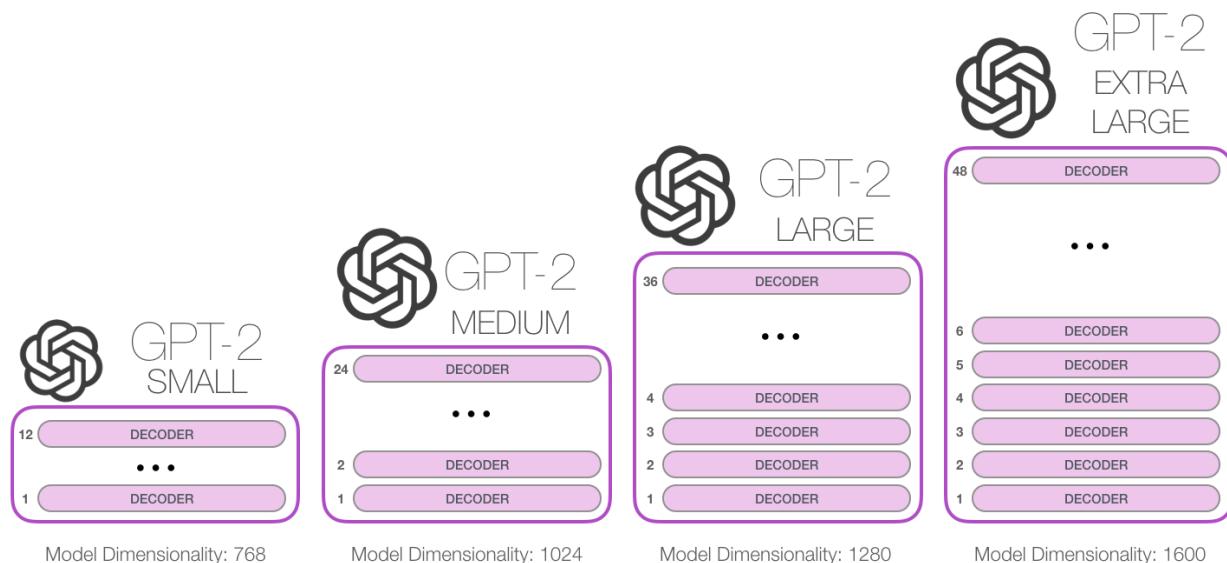


Figure 6. GPT-2 models

Above: figure showing the different architectures for each of the GPT-2 models ^[80]

In previous sections we have described the history, architecture, and process of pre-training GPT-2. While powerful, language models are not perfect, and they might fail in some specific tasks. For example, when we asked the default GPT-2 small (the GPT-2 model with 12 layers) to write about COVID (prompt in blue), it produces the following output:

“What is COVID? It's an acronym for "Conscious Objective Expression," so upperclassmen have been learning about it for years. It really doesn't appear that proprietary MIT offers anything new, unfortunately.”

The result is expected. After all, when GPT-2 was trained, COVID had yet to impact our lives. In these cases, instead of re-training the language model from zero or adapting their architecture, it has been proven enough to simply fine-tune the model on a new dataset, leveraging on the learnt semantic representations. In our example above, in order to generate COVID-related articles, we would need first to gather a specific dataset (like an updated news database or scientific

articles about COVID) to use for fine-tuning GPT-2. This way, the model would be able to learn what COVID means and the different words that are associated with it (e.g.: pandemic, global outbreak, etc.).

Fine-tuning is an additional learning step^[82], which leverages the previously obtained token embeddings^{[81][83]} and (often) adds a small number of new parameters to increase performance in the desired task.^[82] In other words, it allows the model to adapt the general embeddings, trained on generic tasks, to the new task-domain. This allows to transfer the knowledge from a high-resource domain to a limited-resource one.^[84] The common way of fine-tuning Language Models is to simply give the model new domain-specific examples to learn.^[85] If the examples include labels (for example, given a text classify its genre), one could simply add a linear output-layer to predict the target value.^[22]

$$P(y|x^1, \dots, x^m) = \text{softmax}(h_l^m W_y). \quad [22]$$

with W_y the parameters, h_l^m the output of the final transformer layer with x as input tokens and y the target.

In this particular case, the objective to maximize would be:

$$L_3(\mathcal{C}) = L_2(\mathcal{C}) + \lambda * L_1(\mathcal{C}) \quad [22]$$

with L_1 the standard language model objective

$$L_1(\mathcal{U}) = \sum_i \log P(u_i | u_{i-k}, \dots, u_{i-1}; \Theta) \quad [22]$$

and L_2 the labeled-task specific objective

$$L_2(\mathcal{C}) = \sum_{(x,y)} \log P(y | x^1, \dots, x^m). \quad [22]$$

While there are more advanced methods of fine-tuning, like reinforcement based-fine tuning learning ^[85] most models simply fine-tune using a new corpus. Due to the high-quality of the pre-trained embeddings, fine-tuning usually only requires a few iterations.^[82]

2.3. Conditional Text Generation with Transformer-Based LM

Though LMs can be used to generate fluid text (as we saw in the case of GPT-2) further customization based on additional data —beyond the original writing prompt— remains challenging. Many transformer-based LM have emerged during the last years as alternatives, each offering a different way to condition the generation. In following subsections, we explore in more detail these models, their advantages and disadvantages, and link their progress to our thesis research. Broadly speaking, we can distinguish two groups of approaches: those that train their own custom Language Model and those that build on previously released Language Models.

2.3.1 Language Models for Conditional Generation

Conditional Transformer Language (**CTRL**)^[4] is a Language Model that offers the possibility to guide the generation based on multiple control codes: 1) domain (e.g.: Wikipedia, Project Gutenberg, etc.), 2) style, 3) topics, 4) dates, 5) entities, 6) relationships between entities, 7) plot points and 8) task-related behavior, among others.^[4] CTRL was trained on 140 GB of data in multiple languages (English, Spanish, French, German) and has a vocabulary of 250k tokens and a transformer architecture of 48 layers.^[4] It is worth mentioning that CTRL does very little data preprocessing. As such, most of the control codes rely on the URL structure: the URL from each document is given at the start of the input sequence, allowing the model to learn subdomain, dates, and entities.^[4] At generation time, the URL is given as a prompt to condition the output.^[4] While multiple examples of generations are given, CTRL does not offer any evaluation metric, nor does it compare itself to other models. Additionally, it is not clear the effect of some of these control codes in the generation. Below some generation examples extracted from the paper with the prompt in blue (the prompt indicates to the model what to generate and resembles an actual URL, but does not resolve to any webpage).

“Links <https://www.cnn.com/style/09/20/2018/george-clooney-interview> George Clooney on the future of his acting career\n\nBy\n\nUpdated 10:51 AM ET, Thu September 20, 2018\n\nChat with

us in Facebook Messenger. Find out what's happening in the world as it unfolds.
Photos: George Clooney, 'Ocean's 8'
Actor George Clooney attends a photocall for "Ocean's 8" at Grauman's Chinese Theatre on August 31, 2018, in Los Angeles. . “[4]

“Links <https://www.cnn.com/politics/09/20/2018/george-clooney-interview> JUST WATCHED
George Clooney on the Trump administration
MUST WATCH
(CNN) Actor and activist George Clooney, who has been a vocal critic of President Donald Trump, said he is "ready to go back into the political arena" after his role in an anti-Trump documentary was cut from theaters this week. .” [4]

Progressive INsertion based Transformer (**POINTER**) [12]: Trained on 12 GB of Wikipedia (approximately 1.99 million sentences of 64 tokens maximum [12]), POINTER differs from the previous models based on the fact that it generates text using an insertion-based method [12], iteratively refining the output. POINTER is a non-autoregressive model: given the keywords (constraints) the model first outputs high-level words (nouns, verbs, adjectives) between the constraints and continues inserting lower-level words (pronouns, prepositions) until it considers the sentence to be complete [12]. While innovative, this can lead to some undesired results. As we can see in the generated example below, some words that we would like to keep together (like “Natural Language Generation”) are separated in the iterative process. This might be solved by training on a larger corpus and/or expanding the available vocabulary. Below, we show a sample we generated using their on-line demo (prompt 1) and an example extracted from their paper (prompt 2):

PROMPT 1: *thesis natural language generation*

GENERATED: *and his 1965 doctoral thesis , a dissertation about the natural processes in the human language , in the first generation of theory .*

PROMPT 2: *sources sees structure perfectly*

First Iteration: *sources company sees change structure perfectly legal*

Second Iteration: *sources suggested company sees reason change tax structure which perfectly legal .*

Third iteration: *my sources have suggested the company sees no reason to change its tax structure , which are perfectly legal*

Fourth iteration : *my sources have suggested the company sees no reason to change its tax structure , which are perfectly legal* ^[12]

GROVER ^[10]: GROVER is a language model which can generate text conditioned on domain, date, authors, headline and body. GROVER was trained using the same architecture as GPT-2 and uses RealNews, a news article corpora extracted from Common Crawl^[10]. GROVER stands out from the rest of the models for multiple advantages: 1) while it can condition the body generation based on the context (e.g., date and authors) it can also be given a body and additional context and create the missing information. For example, given a body, headline, date and domain it can generate the author names (see figure 7) ^[10] 2) Human evaluators classified GROVER generated fake-news as trustworthy mainly because of their quality.^[10] 3) GROVER responds to concerns raised by OpenAI regarding misuse of LM and designs a countermeasure : GROVER is able to detect text generated by GROVER models, with an accuracy of 92%^[10] which is not trivial considering how human evaluators rated GROVER’s generations.

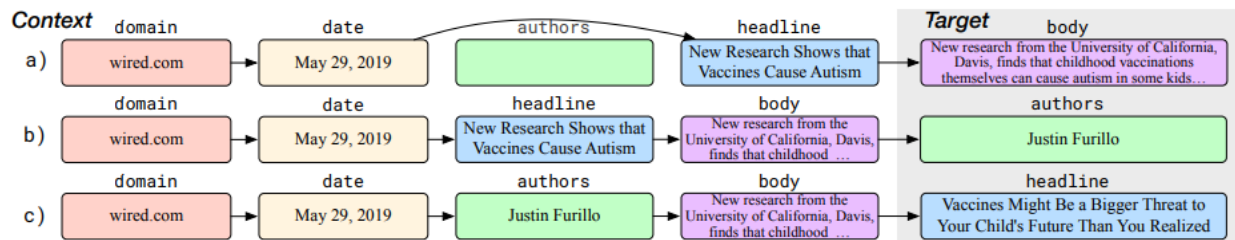


Figure 7. GROVER context example

Above: figure showing how GROVER can be used to generate the missing context ^[10]

2.3.2 Modifications to existing Language Models

The Plug and Play Language Model (**PPLM**)^[6] is based on GPT-2 and aims to give the user control over GPT-2’s generation.^[6] Unlike other models, which require fine-tuning or new architectures, Plug and Play steers GPT-2 by modifying its gradients.^{[6][86]} In more details, it takes a partially generated sentence and calculates

- 1) the probability distribution of the text $\log(p(x))$
- 2) the likelihood of an attribute being present in sentence x : $\log(p(a|x))$

3) the gradients of (1) and (2) with respect to the hidden representations of the LM.^[86]

It uses the gradients calculated in step 3 to update the parameters of the model, guiding the generation to include the desired attribute.^[86] It is worth mentioning that this only works for an autoregressive model (like GPT-2).^[86] Attributes can be words or a bag-of-words for a topic: For example, we could make a bag-of-words for the topic “Fantasy” by adding related words such as wizard, dragon, princesses. We present a sample generation of the model extracted from the paper, an open-story generation using negative as an attribute.

[Negative] Once upon a time, a young woman with a very strange, very dangerous disease was found in a hospital. Every day, the patient would come to her room, sit down on the bed, lay down, and then suddenly scream. But, one day, the doctor discovered that the patient was not dying of cancer, but of tuberculosis. Because of that discovery, his doctor had the patient taken to the hospital’s hospital and treated there. Until, finally, the doctor died. And, ever since then, the patient has been in terrible pain, constantly screaming for help.^[6]

Progressive Generation of Text (**ProGeT**)^[19]: Text produced by GPT-2 tends to be incoherent and repetitive, even after fine-tuning it on a domain-specific corpus.^[19] Progressive Generation of Text aims to tackle this problem by first generating informative words and adding finer-details in multiple-iterations.^[19] Instead of giving one language model the full-generation task, each stage has its own LM (with its own vocabulary) and it focuses on a specific sub-task, building the text in multiple-steps.^[19] Below an example from their paper, showing how the model generates an article about football star Beckham in multiple-steps:

First Stage: *beckham \n liverpool bayern chelsea \n beckham chelsea mancini \n liverpool \n club [...]*

Second Stage: *beckham newcastle \n footballer acquisition transfer liverpool bayern munich chelsea munich \n beckham chelsea season italian siro gianluigi mancini season italy \n season liverpool season \n wembley club \n [...]*

Final Stage: *(CNN) – England striker Ashley Beckham has joined Premier League strugglers Newcastle United. \n England Football Association chairman Stephen McClaren confirmed the acquisition of the 20-year-old England international on Wednesday, making the move official following a lucrative transfer agreement with Liverpool, Bayern Munich and Chelsea. \n Beckham spent the first half of this year on loan at Chelsea and did not start a single match during the season at the Italian capital’s San Siro despite being*

named the player of the season of the year by Italy. \n However, he played 17 times last season and 12 times this season for Liverpool, making 33 league appearances and scoring seven goals. \n He is currently third on the all-time England goalscoring list behind only England manager Alex Ferguson and new club teammate Paul Gascoigne. [...]^[19]

It is also worth mentioning that ProGeT can work with any Language Model or even a combination of languages models (in their original paper, they use GPT-2 with BART).^[19] By generating in steps, ProGeT can create large corpora of thousands of words.

PlotMachines^[18]: Built on top of a GPT-2 model, PlotMachines is an end-to-end trainable transformer model capable of conditioning the generation based on an outline.^[18] Wishing to emulate how writers create stories, PlotMachine uses an outline of the plot as input and outputs a long-format text which incorporates the elements and characters present in the outline.^[18] In order to do so, PlotMachines introduces a dynamic memory in the model, which keeps track of the plot elements as they are incorporated into the generation and tracks the written topics to maintain coherence across the different paragraphs.^[18] To be more specific, at each time step it uses the outline (o), the discourse type for the current paragraph (d^i) (introduction, body, conclusion), the previous paragraph (h^{i-1} encoded using GPT) and the previous iteration memory (M^{i-1}) to condition the generation.^[18]

$$(\mathbb{P}^i, h^i, M^i) = \text{PM}(o, d^i, h^{i-1}, M^{i-1})$$

The outline is a series of keywords, representing events and characters of the story (example: Harry Potter `_kw_ wizard school _endkw_`). In order to train the model, PlotMachines adds various unique tokens to the input data including: 1) `_kw_` for each plot point 2) `_endkw_` for the last plot point 3) `_i / _b / _c_` for the discourse representation 4) `_t_` for the title, etc.^[18]

The paper presents how the model performs under different configurations including a model without the memory, previous paragraph and discourse type, a model including the previous paragraph but without the discourse and memory and a model with the discourse and previous paragraph but no memory.^[18] This study on the effects of the tags is one of the main inspirations behind our thesis.

2.4. Problems with Conditional Text Generation

So far, we have presented a wide-variety of models for conditioning text generation including Language Models pre-trained specifically for the task (GROVER^[10], POINTER^[12] and CTRL^[4]) and those that modify a pre-existing Language Model (PlotMachines^[18], ProGet^[19] and PPLM^[6]). In this section we go over the current problems of conditioning text generation, and possible solutions.

2.4.1 General Text Generation Problems

Natural Language Generation has many known problems, including degeneration^[1], toxicity^[6], exposure-bias^{[5][8]}, context fragmentation, data contamination^[20], among others. In this section, we go briefly over them to understand what to expect from our models.

Degeneration

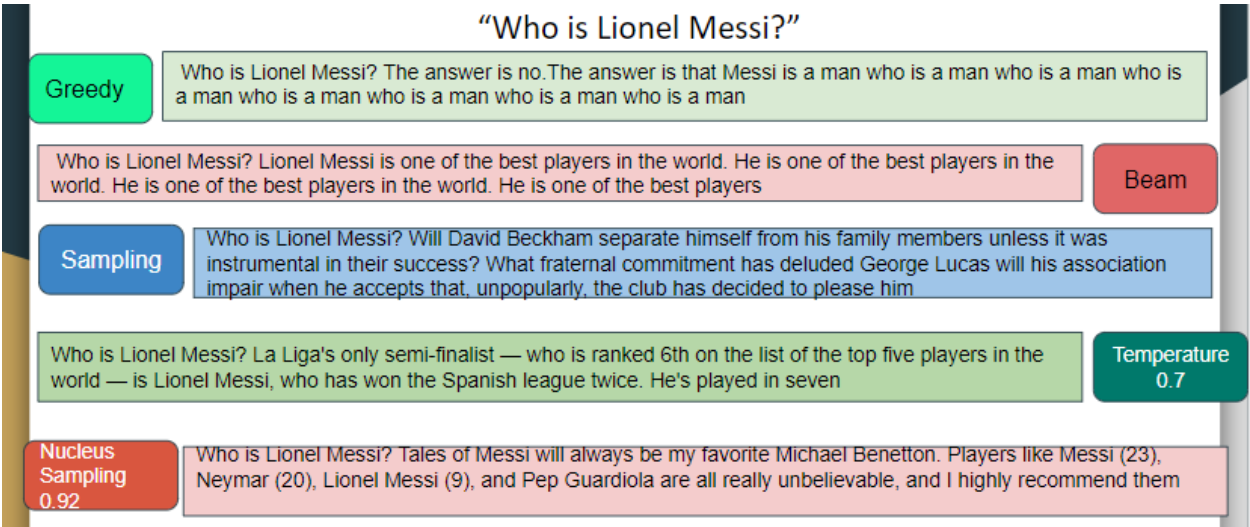


Figure 8. 'Who is Lionel Messi?' Degeneration examples

We say that degeneration occurs when the output text gets fixed on a repetitive loop, or the resulting text is bland, incoherent.^[1] The motive is quite simple: Language Models are trained to predict the next most likely word based on the context, so it is possible for them to get stuck in a loop where they just output the highest-probable words. As we can see in the sample above (**generated** using an out-of-the-box GPT-2 model) when using Greedy decoding (picking the

highest probable word at each time-step^[88]) the resulting sentence gets stuck in “a man who is a man”, producing a poor-quality result.

There are many decoding strategies that can be used to prevent degeneration. To name a few: beam-search, sampling, top-k, nucleus sampling, etc.^[88] We briefly go over them to understand their differences.

Beam-search introduces a modification to the Greedy algorithm: instead of selecting the most probable word at each time-step, beam keeps the n most probable words and picks the end-sequence with the highest probability. By doing so, we reduce the risk of missing a probable sequence hidden behind a “second-best” word.

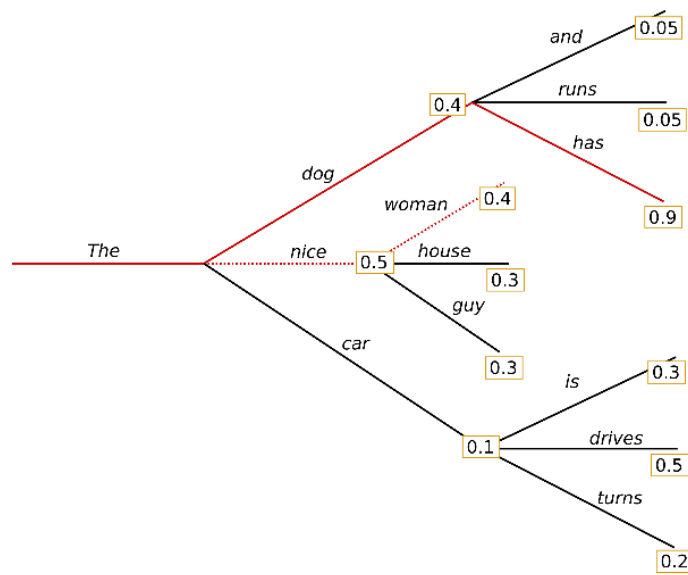


Figure 9. Example of Greedy and Beam Search

This is clearly illustrated in the figure above, from Hugging Face’s decoding guide.^[88] While the greedy algorithm only picks the sequence “The nice woman”, a beam-2 search also consider the sequence “The dog has”, allowing us to reach a more probable sequence (0.36 against 0.2).^[88]

We might not always want to pick the most probable word. At each timestep, the sampling method selects a random word based on its conditional probability distribution. While it reduces repetition, it can lead to incoherent generations^[88] (In our example, it goes from Messi to Beckham and then George Lucas). To prevent this, it is possible to transform the distribution by making

highly probable words more likely to be selected (and vice-versa with not so probable words) by scaling the values of the tokens.^[88]

Another popular decoding strategy is top-k sampling, where the algorithm simply selects the next word from the most likely k words.^[88] Recently introduced, Nucleus sampling takes top-k a step further: at each timestep, the decoding algorithm selects from the top-p words whose probability sums up to p. In a simpler example, for one word (“Lionel”) we can have a pool of 10 candidate words with a total probability of 92% (“Messi”, “is”, etc..) while in the next iteration the algorithm will have to pick from a new pool, with another set of words (“is”, ”was”, ”will”, ”born”) whose probability sum to 92% .^[88]

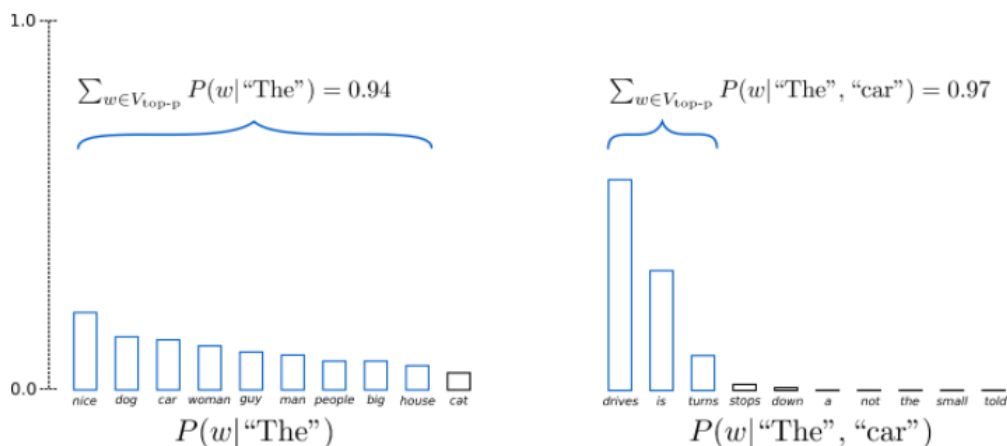


Figure 10. Nucleus sampling example

Visualization of nucleus sampling, from Hugging Face’s decoding guide.^[88] We can see how in the first iteration 9 words are needed to get a probability above 92% while in the second case only three are considered.

It is worth mentioning that these methods can be used in unison: one could use Top-k to reduce the pool of candidate words and Top-p to select from them. While the results are more promising than the Greedy algorithm, it is still possible for nucleus-sampling or top-k to produce degeneration.^[88]

Toxicity

We have briefly presented the problem of bias in Chapter 1, and how LM can learn harmful stereotypes and present bias against genders, religions, and races.^[50] This is a problem that all LM face, including the recent GPT-3.^[21] While it is alarming, we notice that researches are addressing the issue (GPT-3's paper includes a detailed evaluation of the model bias^[21]) and new metrics and tasks are being developed, such as StereoSet, a dataset to evaluate stereotype bias.^[50]

From the models described in the previous section, only PPLM has evidence on reducing toxicity. By adding toxic words as a control attribute and changing the gradient update to negative, PPLM is capable of conditioning the model to avoid the usage of toxic words.^[6] Additionally, since it does not require fine-tuning, new words can be included as soon as they are detected. We expect to see an increase in metrics and tasks to produce bias-free LM in the next years.

Exposure Bias

While training, the Language Model learns to predict the next word based on the ground-truth context. However, during the generation, the model uses its own generated words as context. Because of this, the model could be biased, only performing well when ground-truth tokens are present.^{[5][8][89]} This prevents the model from generalizing to new data and can reduce the quality of generated text.^[90] Most popular approaches for solving this problem include Generative Adversarial Networks or reinforcement learning.^[89] Interestingly, because the GROVER classifier is trained on GROVER generation it reduces its exposure bias.^[10]

Other Problems

Data contamination occurs when the model has seen the answer to the test case in the training data, leading to a false increase in results.^[20] With the high volume of datasets being used (570 GB for GPT-3^[21]) we can no longer guarantee the perfect split of train-test data. As we will see in Chapter 3, even for a small dataset like Wikiplots we can find data contamination if we are not careful. How to decontaminate test datasets has quickly become an important area of research.^[21]

Transformer (and the derived Language Models) have a fixed content length. Mainly due to the self-attention layer, Transformers are not able to take long sequences of text (the computational and memory needs for self-attention grow quadratically with sequence length).^[11] This forces researchers to cut down passages into smaller sequences, fragmenting the context.^[11] This is relevant to our tasks, since we train a Language Model on long summaries from Wikipedia and need to reduce the input length. In Chapter 3 we explain the different measures we use to prevent relevant information from being lost.

2.4.2 Reproducibility

To end this chapter, we would like to discuss one of the main problems of conditional text generation: reproducibility. Unlike other NLG tasks, where standard metrics or task-specific datasets can be used to measure model performance, conditional text generation lacks a standard to enable a fair comparison between the models, relying on reproducibility to test them.

While all models sound impressive, and the sample texts in the papers are engaging and coherent, few of the models presented in section 2.3 have ready-to-use implementations or tested code repositories. Although most of them have code, the resources required to reproduce the experiments are not trivial in both time and computational capability.

In particular, we tested the reproducibility of PlotMachines^[18] and found out that:

- 1) training data is not provided (additionally, the cited extraction method was not working)
- 2) the code is incomplete (among others, the functions to encode the previous paragraph is omitted) and, as presented, does not run (we took the time to correct multiple errors in the code - like missing variable initiation and missing arguments when calling functions- only to find that it would require more computational resources than what we had available).
- 3) Experimental information is missing (for example, the number of training epochs).
- 4) There is data contamination in the test set (explained in Chapter 3).

While we could not test the reproducibility of PlotMachines, we found their work inspiring, and decided to expand on their ablation study. In the next Chapter we present their experiments

and the new models that we use to measure the effect of different metadata in conditioning the generation.

Chapter 3

Methodology

In the previous chapters, we discussed the current state-of-the-art language models and the different methods to adapt them to conditional text generation. In this chapter, we present our proposed experiments and their expected results. Our aim is to determine which metadata (tags) should be used in conditioning a model (GPT-2) for generating a story. We evaluate the results using both automatic evaluation metrics (perplexity, Self-BLEU, ROUGE) and human evaluations. We also go into details about our dataset, Wikiplots+, one of the contributions of our work.

3.1 Wikiplots Dataset

For our experiments we use the Wikiplots^[28] dataset, which contains 112,936 English plots and titles extracted from Wikipedia. Wikiplots consists of 319,205 unique words, with an average length of 354 words per plot. Since the data has been scraped from Wikipedia, we can trust the summaries to maintain the grammatical and structure quality characteristic of Wikipedia articles.

It is worth mentioning that Wikiplots contains plot summaries from not only books but also movies, tv shows, videogames and textbooks. While some plots indicate if the summary belongs to a movie, book, tv show or game, on a general note it is not possible to distinguish between them. Popular tv series will have a detailed plot description for more than an episode, as is the case for How I Met Your Mother, The Simpsons, Buffy, among others. For examples, see Appendix A - Wikiplots. In the graph below we can see the distribution of words per plot. Most plots have between 100 and 199 words.

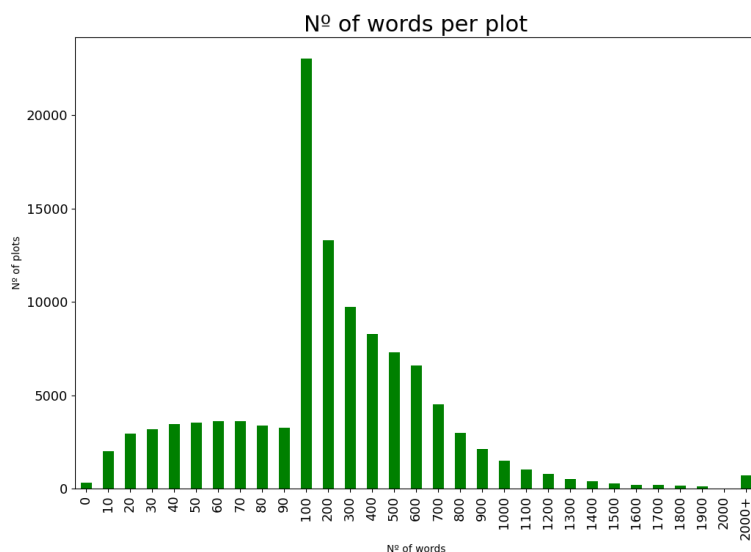


Figure 11. Number of plots per word group for Wikiplots

3.1.2 Expanding Wikiplots

In order to match each plot with a genre tag, we combine the Wikiplots dataset with the publicly available Internet Movie Database (IMDB).^[35] IMDB contains information on movies, games and tv shows, including —but not limited to— directors, genres, year of publication, etc.^[36] We use IMDB to add to Wikiplots the genres that are missing. Additionally, we expand with two other Wikipedia plot datasets: Wikipedia Movie Plots (WMP)^[33] and Book Summary Dataset (BSD)^[34] which contain the title, plot, genre and author (among other metadata) extracted directly from Wikipedia. The final dataset, Wikiplots +, has 105,124 unique plots and titles.

Dataset	N° Titles	Repeated Titles	Repeated Plots	Unique Titles
BSD	12,841	204	24	12,613
WMP	34,885	2,515	411	31,926
Wikiplots+ IMDB	112,936	13,090	28	99,818
BSD + WMP+ Wikiplots+IMDB	144,357	38,759	474	105,124
Wikiplots+	105,124	0	0	105,249

Figure 12. Information on all datasets

Before joining the datasets, we need to bring attention to some problems that might arise, in particular, we want to avoid repetition of plots and titles. We pay special attention to this to

prevent our test and validation sets from having samples present in the training set. We also want to avoid having the same sample twice in the training set. We found that when working with plot datasets authors tend to ignore these problems and simply split the data using a random seed, overexposing their training to the same plot multiple times and/or testing on instances that the algorithm has already seen before. For example, in PlotMachines^[18] the paper is evaluated on “Harry Potter and the Order of the Phoenix” but was trained on “Harry Potter and the Order of the Phoenix (Film)”. To prevent this, we highlight all problems that can be found when working with books and their summaries in the next section. In particular we identify 4 cases: (A) same title, similar plot (B) same title, different plots (C) different title, same plot (D) translated titles.

Same Title, Similar Plot

As we can see in the table above, it is not uncommon for a dataset to have repeated titles. The most usual case is when the title belongs to a popular series or a classic that was adapted to more than one medium (books, movie, tv series, etc). For example, Anne of Green Gables by Lucy Maud Montgomery, can be found 5 times in the original Wikiplots dataset: one for the book, and one for each film in 1919, 1934, 1956 and 1985. While the repetition of a title might be an indicator of its popularity (after all, the book was adapted to 4 films), we have decided to keep only one sample of each.

Same Title, Different Plot

Not every repetition is due to popularity. For example, we can have titles with the exact same name but completely different plots. This is the case when the title is made of common words or phrases. For example, “The Immortals” (see Appendix B to compare the plots). Though we tried using extra metadata (author or director) to determine if the titles were unique, we concluded it was not sufficient. This is largely because movie adaptations have a different author (director) than the original book. For example: Harry Potter and the Order of the Phoenix was directed by David Yates, while the original book was written by J.K.Rowling. To prevent repeating titles, we only keep one of each.

Different Title, Same Plot

So far, we have been focusing on titles alone. Though rarer than title repetition, we do have multiple cases of repeated plots. For this work, we consider two plots to be similar if they are exactly the same, word by word. Most cases we have found are caused by changes in the title. For example : “Mr. Bug Goes to Town” and “Mister Bug Goes to Town”. While the title is different, the plot is the same. We keep only one of the duplicated plots.

Translated Titles

While Wikiplots contains only English plots, the titles are usually found in their original language. For example: “La boda de Valentina” (Spanish) and “Inuyashiki” (Japanese). This can cause plot repetition. To prevent it, we match all titles with IMDB dataset, which includes both the original title and the commercialized English translation. We only keep the commercialized English translation. It is worth mentioning that not all titles have a translation and can be commercialized using their original name.

3.1.3 Building Wikiplots +

All three datasets (Wikiplots, BSD, WMP) are joined by title. In order to prevent duplicates as discussed above, we made sure to record all titles with their commercialized English name (translated using IMDB) and removed any additional information attached. Looking back at the Anne of Green Gables example, we removed the additional information from the title to detect duplicates (Original was : Anne of Green Gables (film 1919) is reduced to Anne of Green Gables).

Once we join the datasets, there are two sources for the plot genre (classification). One is the original classification from the dataset (both BSD and WMP have a classification tag) and the other one is the classification from IMDB. For each matched title, we join all classifications and order them by frequency. This way, the most popular tag will appear first. Example: “unknown, drama, horror, drama” would turn into “drama, horror, unknown”. The unknown tag is used to replace classification tags that appear less than a hundred times in our dataset (e.g: famous line, heist, ninja, etc.).

For titles with multiple instances in the IMDB dataset (ex: “The Last Command”) we group all similar instances in IMDB and pick the genres with the most repetition. We work under the assumption that similar titles will have similar genres (ex: “Trauma” —unless it’s a parody— is highly unlikely to be part of the comedy, romance genre). More details about the genre selection will be given in the next section.

The final Wikiplots + dataset has 105,124 unique plots. Only 32,015 (30.5%) are classified as pure unknown. Additional information on Wikiplots + is summarized in the sections below.

Tag	N° Cases	Classification	N° Tags	Classification	N° Tags
[unknown]	32015	unknown	102690	sci-fi	10853
[unknown,'drama']	4031	drama	40019	fiction	7901
[unknown,'comedy']	2604	comedy	34163	history	5408
[drama,'unknown']	1184	action	18797	adult	4437
[unknown,'comic/manga','comedy']	1143	crime	15895	biography	4066
[comedy,'unknown']	962	romance	15519	musical	3567
[unknown,'comedy','drama']	861	adventure	15213	western	3430
[unknown,'drama','romance']	832	family	14490	war	2877
[unknown,'comic/manga','comedy','family']	814	comic/manga	13983	sports	2769
[unknown,'action']	638	documentary	12259	teens	815
[unknown,'fiction']	618	fantasy	11856	dystopian	126
[unknown,'sci-fi','fiction']	594	mystery	11384	thriller	115
[drama]	542	horror	11047		

Figure 13. Most popular tags in Wikiplots+

3.1.4 Creating Wikiplots 2

While Wikiplots + contains the title, plot and genre needed for our work, since our end-goal is to create a long format text (300 —500 words) we need to curate the dataset before we can use it. The curated version, Wikiplots 2, has been built on the following premises:

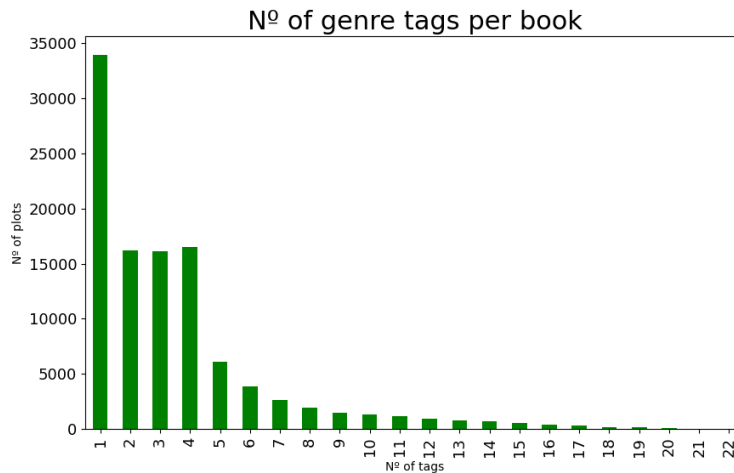


Figure 14. Number of plots per genre tags for Wikiplots+

1) Multiple Genre Tags

As we can see in the graph above, our Wikiplots + dataset has multiple genre tags per plot. Though on average, each book has 3.4 tags, we have cases with more than 6 tags per book (see example below “The Girl Who Loved Tom Gordon”). We reduce the tags to simplify training. In particular, we: (A) For plots with more than 1 tag, remove the unknown classification. Example: [horror, unknown] becomes [horror]. (B) For plots with more than 3 tags, we keep only the first three. We believe that the order in which tags are present (ordered by repetition, with the first being the most frequent tag) indicates their relevance to the plot. For example: Oedipus Rex [classic, fiction , drama, adult , history, fantasy] will be shortened to [classic, fiction, drama].

Sample of a plot with more than 12 tags:

Title: The Girl Who Loved Tom Gordon

Tags: horror, unknown, fiction, mystery , fantasy , adventure, adult , teens , thriller, drama , family, sports

Plot: The story is set in motion by a family hiking trip, during which Trisha's brother, Pete, and mother constantly squabble about the mother's divorce from their father, as well as other topics. Trisha falls back to avoid listening and is therefore unable to find her family again after she wanders off the trail to take a bathroom break. ... Now and then she listens to her Walkman to keep her mood up, either to learn of news of the search for her, or to listen to the baseball game featuring her favorite player, and "heartthrob," Tom Gordon ... Eventually, a combination of fear, hunger, and thirst causes Trisha to hallucinate. She imagines several people from her life, as well as her hero, Tom Gordon, appearing to her. It is left unclear whether increasingly obvious signs of supernatural events in the woods are also hallucinations ... Eventually she begins to believe that she is headed for a confrontation with the God of the Lost, a wasp-faced, evil entity who is hunting her down...

2) Number of Words

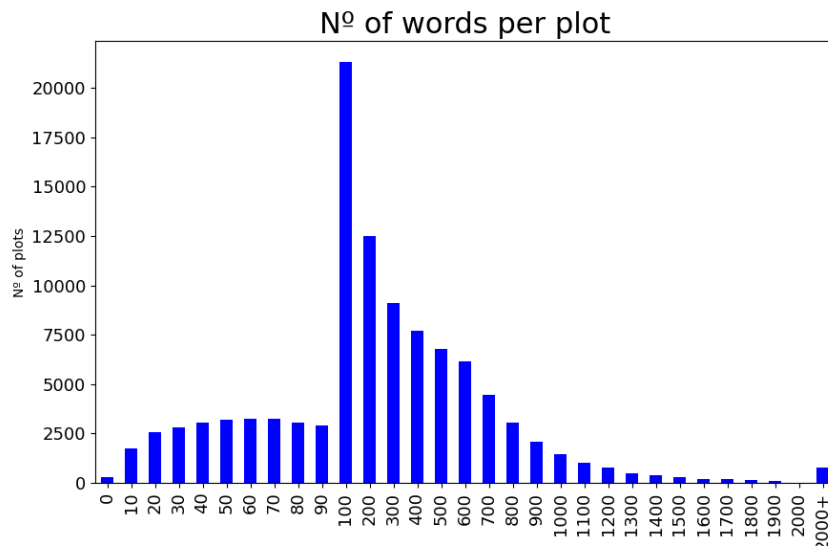


Figure 15. Number of plots per word group for Wikiplots+

The graph above shows the distribution of plots by number of words. As we can see, the bracket 100-199 words has the greatest number of plots. The average plot length is 367 words. In the original Wikiplots dataset, it is possible to have plots with less than 20 words (e.g Title: The Devil Makes Sunday Plot: In 1840 on Norfolk Island, convicts rise up against their guards). Our end-goal is to fine-tune a model that is able to produce a long-range text (300-500 words). We discard all plots with less than 200 words (47,427 plots, or 45% of the data). We use 57,697 plots from which 14,585 (25.28%) are classified as pure unknown.

3) Context Fragmentation

Since most natural language models have an input constraint (GPT-2 can take —at most— 1024 tokens^[99]) it is not uncommon to see datasets adapted to fit their requirements. It is common for the text to be cut after a preset word limit is surpassed, regardless of the sentence structure or any other semantic boundary. We can observe this in PlotMachines,^[18] Peters et al. 2018^[32] and Al-Rfou et al. 2018^[31] among others. This is referred to in academia as context fragmentation.^[30]

In our particular case, the plot with the greatest number of words has 10,344 words (Title: Kushiel's Mercy). To prevent context fragmentation, instead of simply cutting the plots with more than 900 words we use extractive summarization to summarize them. This way, we are able to keep 5,874 plots that would otherwise be discarded.

As the original dataset is composed of extracted lines of Wikipedia, we have no way of knowing the original paragraph structure. We have decided to arbitrarily split each text into five groups of equal size (so each group has 20% of the lines). We name the first group introduction and the last conclusion. Only the groups that belong to the body are summarized. By doing so, we keep both the world building (introduction) and the outcome (conclusion) intact and only shorten the connection between them (body). Below, a sample of a summarized plot from “Animal Farm” by George Orwell.

[_start_](#) [_t_ animal farm _i_](#) Old Major, the old boar on the Manor Farm, calls the animals on the farm for a meeting, where he compares the humans to parasites and teaches the animals a revolutionary song, 'Beasts of England'. When Major dies, two young pigs, Snowball and Napoleon, assume command and turn his dream into a philosophy. The animals revolt and drive the drunken and irresponsible Mr Jones from the farm, renaming it "Animal Farm". They adopt Seven Commandments of Animal-ism, the most important of which is, "All animals are equal". Snowball attempts to teach the animals reading and writing; food is plentiful, and the farm runs smoothly. The pigs elevate themselves to positions of leadership and set aside special food items, ostensibly for their personal health. Napoleon takes the pups from the farm dogs and trains them privately. Napoleon and Snowball struggle for leadership. When Snowball announces his plans to build a windmill, Napoleon has his dogs chase Snowball away and declares himself leader. Napoleon enacts changes to the governance structure of the farm, replacing meetings with a committee of pigs, who will run the farm. Using a young pig named Squealer as a "mouthpiece", Napoleon claims credit for the windmill idea. [_end_](#)

[_start_](#) [_t_ animal farm _b1_](#) Once Snowball becomes a scapegoat, Napoleon begins purging the farm with his dogs, killing animals he accuses of consorting with his old rival. Mr Frederick, one of the neighbouring farmers, attacks the farm, using blasting powder to blow up the restored windmill. Though the animals win the battle, they do so at great cost, as many, including Boxer the workhorse, are wounded. Despite his injuries, Boxer continues working harder and harder, until he collapses while working on the windmill. Benjamin, the cynical donkey, who "could read as well as any pig", notices that the van belongs to a knacker, and attempts to mount a rescue; but the animals' attempts are futile. During a poker match, an argument breaks out between Napoleon and Mr Pilkington, and the animals realise that the faces of the pigs look like the faces of humans, and no one can tell the difference between them. The pigs Snowball, Napoleon, and Squealer adapt Old Major's ideas into an actual philosophy, which they formally name Animalism. Soon after, Napoleon and Squealer indulge in the vices of humans (drinking alcohol, sleeping in beds, trading). [_end_](#)

[_start_](#) [_t_ animal farm _c_](#) Later, Napoleon and his pigs secretly revise some commandments to clear them of accusations of law-breaking (such as "No animal shall drink alcohol" having "to excess" appended to it and "No animal shall sleep in a bed" with "with sheets" added to it). The changed commandments are as follows, with the changes bolded: * 4 No animal shall sleep in a bed with sheets. * 5 No animal shall drink alcohol to excess. * 6 No animal shall kill any other animal without cause. Eventually these are replaced with the maxims, "All animals are equal, but some animals are more equal than others", and "Four legs good, two legs better!" as the pigs become more human.

This is an ironic twist to the original purpose of the Seven Commandments, which were supposed to keep order within Animal Farm by uniting the animals together against the humans, and prevent animals from following the humans' evil habits. Through the revision of the commandments, Orwell demonstrates how simply political dogma can be turned into malleable propaganda. _end_

The full plot can be found in Appendix D. An explanation of extractive summarization and its limitations is present in Appendix C.

3.1.5 Wikiplots 2

Below we summarize the main features of Wikiplots 2, the dataset that we use to fine-tune the language model. Wikiplots 2 is composed of 57,679 plots, with an average plot length of 500 words and a total vocabulary of 251,576 unique words. No plot has more than 3 tags. The most common genre is drama (21,612 tags) followed by comedy (18,095) and then unknown (14,585). We have 29 plots with 900 words, while the shortest has 200 words.

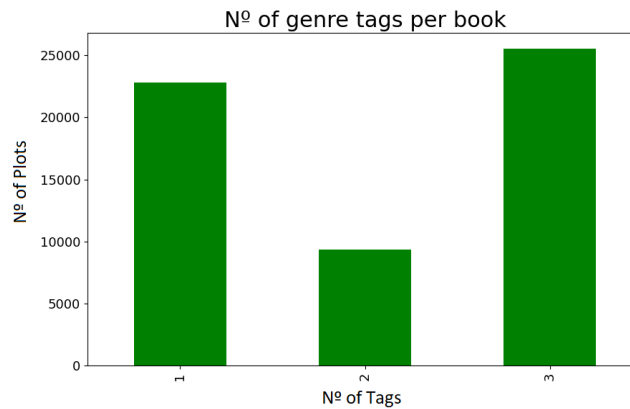


Figure 16. Number of tags per book for Wikiplots 2

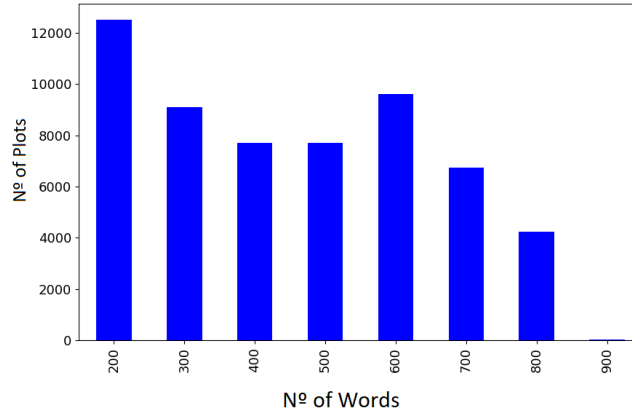


Figure 17. Number of plots per word group for Wikiplots 2

3.2. Methods

As we have seen in the previous chapter, there are two groups of transformer-based Language Models that allow conditioning text generation: custom built (trained from zero) and those that build on publicly available models. We use the second group. As we have stated during the literature review, many language models are very recent, and their capacity still needs to be assessed. We opted for a published language model for four reasons: (1) Training a language model from scratch is too expensive, both in time and computational resources; (2) Using a published Language Model makes our results applicable to a larger range of other works; (3) By using a published language model, we have access to a larger range of resources, including tutorials and published papers to compare our results. This is not trivial: during our experiments of the reproducibility of PlotMachines we were unable to get the required information (like hyperparameters or file samples) from the authors. (4) Last year, we were amazed by the Unicorn tale published by OpenAI. We believe that GPT-2 has already learnt the semantic representations needed for a good story. All that it is missing is the guidance, the capacity to be conditioned. We hope to pave the way to other researchers working on conditioning GPT-2.

We expand the research of PlotMachines^[18] on the effects of metadata for conditioning text generation. While this was not the aim of the paper (PlotMachines presents a novel transformer model with memory that allows conditioning on the plot^[18]) we found their ablation study inspiring. In particular, it surfaced the following questions: What metadata (tags) should creative generation be conditioned on? How do we evaluate the contribution of each tag? Does GPT-2

benefit from longer prompts, or would it hinder its generation capacity? These are the questions we will aim at answering with our work.

The use of metadata for conditioning is not new: both transformer-based language models (GROVER^[10] and CTRL^[4]) and non-transformer approaches (Martin et al^[68], Plan-and-Write^[63]) have already used various types of metadata to guide generation. However, few models study the effects of the tags, which might lead to suboptimal results. Though there are some papers that present an ablation study (including PlotMachines^[18] and Martin et al^[68]) they do not give examples of text produced by each of the models. Considering how human evaluation is still the golden metric for evaluating conditional generation, they are missing a key component for determining the success of the model.

In this thesis, we expand the experiments presented in PlotMachines^[18] with six new ablation studies. In particular we take a deeper look at the effect of discourse, genre, outline representation (using Extractive Summarization) and the previous paragraph in conditioning the generation of a GPT-2 model. Our work differs from PlotMachines^[18] in several ways:

- 1) We do not modify the original GPT-2 architecture. We study the capability of the language model —as is— to learn to condition text generation. There will be no memory nor changes in the architecture to incorporate additional parameters (like previous paragraph encoding^[18]).
- 2) Our prompts only use text, unlike the original paper where the previous paragraph was encoded using GPT.^[21] Additionally, instead of using RAKE^[94] to extract keywords, we use extractive summarization. More details will be given in section 3.2.2.
- 3) Following the research of Martin et al.^[68] on metadata, we add the genre tag. We also place a stronger importance on discourse in order to evaluate all models on the same data structure. See section 3.2.1 for more information.
- 4) We use a new dataset, Wikiplots 2, created for lowering data contamination and adding the genre tag.

- 5) Most importantly, we study the effect of each metadata both by itself and in combination with others. We also show generated samples for each of the models, in order to better understand their generation capability. As we will see in Section 4, using automatic evaluation metrics by themselves are not enough to determine the potential of these models.

In total, we present 6 models and a baseline. Models are classified in two: Individual tag studies and Multiple Tags studies. Following the approach presented by Martin et al [68] we first run the experiments on the Individual section (Models 1–3) and determine the tag with the most effect on perplexity. We then build on that tag, adding the others to understand how they interact (Models 4–6). By doing so, we aim to find the model that achieves the lowest perplexity.

Below a summary of the metadata study of PlotMachines (blue) for GPT-2^[18] and the models we study in this thesis (gray). In Chapter 4 we present the results that shows that genre is the tag with the most effect from the individual set.

Model	Title	Discourse	Genre	Outline	Previous Paragraph (embeddings)	Previous Paragraph (Text)
GPT-2	*			*		
PM-NoMem-NoDisc	*			*	*	
PM-NoMem	*	*		*	*	
Distill GPT-2 (D)	*					
Distill GPT-2 + Genre (DG)	*	*	*			
Distill GPT-2+ Outline (DO)	*	*		*		
Distill GPT-2 + PP (DP)	*	*				*
DGPT2 + Genre + Outline (DGO)	*	*	*	*		
DGPT2+Genre+PP(DGP)	*	*	*			*
DGPT2+Genre+O+PP (DGOP)	*	*	*	*		*

Figure 18. Comparison between the different models

All models share the same hyperparameters with a learning rate of $5e-5$, a batch size of 4, a random seed set at 42, a gradient accumulation of 4 and 5 epochs. Batch size and gradient accumulation were based on the server's capacity, trying different combinations until we could get it to fit our available memory. We present each model below.

3.2.1 Model 0 - DistilGPT-2 (D)

Our first model is the standard DistilGPT-2 fine-tuned on our Wikiplots 2 dataset. DistilGPT-2 is a reduced version of GPT-2, which weights less (37% less) and runs faster than its

original counterpart^[92], making it easier to train on limited computational capabilities. DistilGPT-2 is obtained by distilling GPT-2: the process of distilling involves training a small model to imitate the results (output distribution) of a larger model.^[91]

Model 0 introduces the discourse tag. Discourse indicates if the paragraph belongs to the introduction (`_i_`), body (`_b_`) or conclusion (`_c_`) as presented in the PlotMachines paper^[18] and described in the previous section. When segmenting the plots, we make sure to respect semantic boundaries so that no phrase is cut short. An example of the input data for this Model (blue represents the outline)

`_start_ _t_hero _i_` HERO is a rock opera modernizing Jesus's last two years of life, as described in the Bible. The story takes place in New York City, in Brooklyn. The world government in this near-future dystopic Earth is centered under the International Confederation of Nations (ICON). Under the iron fist of CON, nearly all religion in the world has been wiped out, except for small occult and mystic sects. Only one synagogue in Brooklyn exists. Currently, New York City is a police-occupied warzone between ethnic gangs and small, isolated revolutionary groups fighting CON.
`_end_`

`_start_ _t_hero _b1_` Of all the ancient world religions, only Judaism survives and flourishes, at least, as much as it can. In Bethlehem, PA, a child named Jesus, but referred to as HERO, is born and forced to flee with his family to the small Jewish section of Brooklyn. Jesus grows up and begins to preach and teach the principles of Christianity to the people of New York City, teaching people to love their enemies and care for each other CON.
`_end_`

`_start_ _t_hero _b2_` realizes HERO is a threat, and the Chief of police Devlin (a derivative of the Devil or Satan), with the help of chief Rabbi Kai (Caiaphas), conspire to end HERO's revolutionary teachings. The Opera is narrated by "Agent Hunter", a former CON agent who met HERO and was soon thrown into prison for joining him against CON. The opera also features Petrov (Peter), Maggie (Mary Magdalene), and Jude (Judas Iscariot) the latter who conspires with Kai and Devlin to betray HERO.
`_end_`

`_start_ _t_hero _c_` The storyline progresses through several stories about Jesus' miracles and sermons, using references from the Bible's four gospels, continues through Jesus' execution, at the hands of CON's angry mob, and eventually ending with his resurrection.
`_end_`

Before we continue, we would like to talk about fairness of both the task and comparison between the models. While it could be possible to train DistilGPT-2 without segmenting the input in paragraphs (our plot lengths are less than 900 words, which tokenized would keep all plots — except some outliers— below the maximum limit of 1024 tokens for GPT-2) this is no longer possible once we start adding the outline and even less when we want to condition on the previous

paragraph. In order to have a fair comparison between all models, they all share the same paragraph structure but different outlines. As such, we keep the paragraph length for each discourse to 100-200 words, so that when they are used as the previous paragraph, they all keep a similar length and below the 1024 limit. Finally, we address the fairness of the task itself. While we study the ROUGE of the models against the original ground truth paragraph, we are not expecting this model to have a high ROUGE. After all, it is unlikely for a model based on just the title and discourse alone to match the golden standard (unless there is data contamination, or the language model has been extensively trained on that particular story). We use this model as our baseline. Any further experiments are done with the intention of improving upon this result.

3.2.2 Individual Tags Models

In this section we go into details about each of the initial tag experiments. All of them respect the discourse structure present in Model 0. In all cases, we fine-tune the original DistilGPT-2.

Model 1 - DistilGPT-2 + Genre (DG)

Our first model with additional tags uses the genre tag (`_g_`), unique to our experiments and not present in the PlotMachines^[18] paper. As with model 0, we are not expecting a high ROUGE. Instead, we hope to see some genre related keywords in the generation (if the genre is fantasy, we expect some mention of magic, knights and other related keywords). For more information on genre selection and how to build the dataset, please refer to the previous section.

Model 2 - DistilGPT-2 + Outline (DO)

For this second experiment, we condition based on the plot outline. While the original paper^[18] suggests using a keyword extractor (RAKE^[94]) we find the extractions to be a poor sample of the plot. We have opted to use extractive summarization, picking the 3 most representative sentences with less than 15 words of each plot. Following PlotMachines, each sentence is separated with a `_kw_` token. We append `_endkw_` at the end to indicate the model that the outline is finished.^[18]

Examples for a is for a-l-i-v-e

(using Rake) plls meet detective gabriel holbrook _kw_ jessica dilaurentis returns _kw_ girls discover wilden dead _kw_ mysterious woman dressed _kw_ disk showing footage _kw_ mona retrieve based _kw_ black veil appears

(using Extractive Summarization) _kw_ The next morning, the girls discover Wilden dead in the street by the church surrounded by coroners. _kw_ The girls flee the scene, with Mona removing the disk showing footage of Ashley running over Wilden. _kw_ In an attempt to earn some of their trust, Mona reveals a bunch of secrets to the PLLs.

As we can see from the outlines above, while some characters might appear in the outline, it is unclear who is the protagonist, or where the action takes place. If we were to ask anyone to help us draft a plot, it would be part of the basic requirements. While it could be possible to append to each outline the main character and place using name entity recognition ^[37] this would significantly add to the computationally expense of the task (in our experiments, a good name entity recognition could take up to 7 seconds per book which makes the cost prohibitive for 140,000 books). This could be feasible in a smaller dataset or using resources that allow for parallelization. Some examples of main characters and places extracted using name entity recognition:

HERO : Jesus , New York City.

A Wizard of Earthsea : Ged, Earthsea

Since the model now has access to phrases it should include in the output, we expect to see an increase in ROUGE score should DistilGPT-2 learn the relationship between the phrases and the plot.

Model 3- DistilGPT-2 + Previous Paragraph (DPP)

For our third model, we use the previous paragraph. We have opted for using text representation and not an encoded paragraph for it would require an architecture change for DistilGPT-2, defying the purpose of finding practical ways of conditioning the generation. In the sample below, we mark the previous paragraph with blue and the current paragraph with green. We use the tokens `_startpv_` and `_endpv_` to indicate that a section belongs to the previous paragraph.^[18] When there is no previous paragraph, we simply write NA.^[18]

`_start_ _t_ hero _startpv_ _i_ HERO is a rock opera modernizing Jesus's last two years of life, as described in the Bible. The story takes place in New York City, in Brooklyn. The world government in this near-future dystopic Earth is centered under the International Confederation of Nations (ICON). Under the iron fist of CON, nearly all religion in the world has been wiped out, except for small occult and mystic sects. Only one synagogue in Brooklyn exists.`

Currently, New York City is a police-occupied warzone between ethnic gangs and small, isolated revolutionary groups fightingCON. _endpv__b1_ Of all the ancient world religions, only Judaism survives and flourishes, at least, as much as it can. In Bethlehem, PA, a child named Jesus, but referred to as HERO, is born and forced to flee with his family to the small Jewish section of Brooklyn. Jesus grows up and begins to preach and teach the principles of Christianity to the people of New York City, teaching people to love their enemies and care for each otherCON. _end_

3.2.3 Multiple Tags

Our study is done in two parts: In the first one (described above) we aim at understanding the individual effect of each tag (metadatum) in the generation. After determining which metadatum has the most positive impact on perplexity, we keep building on that model, with the aim of finding the best combination to produce a coherent output conditioned on the prompt. In our case, we will see in Chapter 4 that genre (Model 1: DG) is the tag with the greatest effect on perplexity.

Based on these initial results, we now present three additional models: Model 4 with genre and outline (DGO), Model 5 with genre and previous paragraph (DGP) and Model 6 with all the available tags (genre, outline and previous paragraph: DGOP). Below, an example of an input for Model 6:

start _t_ hero _g_ drama _g_ comedy _endg_ _kw_ Jesus _kw_ New York _kw_ Of all the ancient world religions, only Judaism survives and flourishes, at least, as much as it can. _kw_ Currently, New York City is a police-occupied warzone between ethnic gangs and small, isolated revolutionary groups fightingCON. _kw_ HERO is a rock opera modernizing Jesus's last two years of life, as described in the Bible. _endkw_ _startpv__i_ HERO is a rock opera modernizing Jesus's last two years of life, as described in the Bible. The story takes place in New York City, in Brooklyn. The world government in this near-future dystopic Earth is centered under the International Confederation of Nations (ICON). Under the iron fist ofCON, nearly all religion in the world has been wiped out, except for small occult and mystic sects. Only one synagogue in Brooklyn exists. Currently, New York City is a police-occupied warzone between ethnic gangs and small, isolated revolutionary groups fightingCON. _endpv__b1_ Of all the ancient world religions, only Judaism survives and flourishes, at least, as much as it can. In Bethlehem, PA, a child named Jesus, but referred to as HERO, is born and forced to flee with his family to the small Jewish section of Brooklyn. Jesus grows up and begins to preach and teach the principles of Christianity to the people of New York City, teaching people to love their enemies and care for each otherCON. _end_

At generation-time we would then give the title, discourse, genre, outline and the previous paragraph to model 6, and expect it to write a coherent follow-up. As the reader can notice, the

number of words we are giving to our model has drastically increased to the point of our training sample being more prompt (text to condition the generation) than target text. It is to be determined if this drastic amount of information has a significant impact on the generated result.

3.3 Evaluating the Models

We analyze 4 metrics to understand the performance of our models: Perplexity, ROUGE, Self-BLEU and human evaluation. For a detailed description of the first three metrics, please see the Literature Review. Our best model should achieve low perplexity (meaning DistilGPT-2 is confident in the prediction), low Self-BLEU (it produces diverse samples) and a fair overlap with the original model (we would be cautious of a high ROUGE as it could be that the sample was present in training).

For the human evaluation we generate three samples per validation prompt, using nucleus sampling of 0.9, k-sampling of 50 and a repetition penalty of 1. We then pick the best of the three samples based on: 1) how well it fulfills the task 2) fluidity 3) coherence. In more detail:

- 1) Task fulfillment: This depends on the metadata used. For all models, we expect to see some conditioning based on the discourse (conclusion should give us an ending while introduction should build the setting, etc.). For models with the genre tag, we expect to see some keywords related to the conditioned genres (in a crime book we expect to see murder and police, in fantasy magic, etc.). All models with the outline will be evaluated in how the outline phrases are used in the generated story (e.g., does the output include the keywords, does it simply copy the outline, or does it keep on mentioning characters and locations present in the outline?). Finally, for models with the previous paragraph, we expect a natural continuation and will pay attention to how they integrate characters and events that have already been introduced.
- 2) Fluidity: How easy to read it is. The less degeneration (repetition, random tokens, excessive use of characters, etc) present in the output, the better fluidity.
- 3) Coherence between the generated sentences. From pronoun coherence to plot structure (did the murdered character come back to life in a non-fantasy setting?) we evaluate how much the generation makes sense in order to select the best outline.

For the automatic metrics, we use a validation set consisting of approximately 100 prompts belonging to 25 stories, with one generation per prompt. For human evaluation, we show and analyze in detail the results of 3 prompts in all 6 models (and the baseline) for a total of 21 generated samples.

Chapter 4

Experiment Results

In this chapter we present results —both numerical and text generation samples— using the models presented in Chapter 3. We do an in-depth analysis of the generated samples and present an additional ablation study of generation hyperparameters. We also propose modifications for future works along with possible venues of studies.

4.1 Evaluation Metrics

As we stated in Chapter 3, we first run the individual experiments (models 0–3) and register their perplexity after 5 epochs. Based on that, we chose the model with the lowest perplexity (model 1- genre tag) and use it as the base for our multiple-tag experiments (models 4–6). Below we summarize the results for each of the models:

Experiment Type	Model	Perplexity	Avg Generation Len	R-1	R-2	R-L	B-4
Baseline	Model 0	18.62	96.216	18.68	0.71	14.78	0.06
Individual	Model 1	5.63	97.381	19.16	0.81	15.42	0.08
Individual	Model 2	7.81	94.784	15.66	0.44	12.47	0.06
Individual	Model 3	6.82	91.732	13.82	0.55	11.53	0.05
Group	Model 4	7.43	95.144	15.90	0.74	12.81	0.07
Group	Model 5	6.25	95.186	16.38	0.60	13.17	0.06
Group	Model 6	5.59	90.289	14.99	0.46	12.50	0.07

Figure 19. Results from the experiments

Results obtained from evaluating the models on a sample of the validation dataset, using 100 prompts from 50 random books. For each generation we use nucleus sampling of 70, a repetition penalty of 1,4 and a length of 120. Hyperparameters were taken from the PlotMachines paper to compare our models with their results.^[18] We report ROUGE-1 (R-1: unigram), ROUGE-2 (R-2: bigram) and ROUGE-L (R-L: longest common subsequence). For Self-BLEU we report the B-4 (4-gram) score.

In terms of perplexity, Model 1 outperforms all other tags in the individual setting, obtaining a score of 5,63. This is not completely unexpected, as the genre by itself encapsules relevant information about a plot (given the same title “Love Forever” the image the reader has will change drastically if we say the genre is romance or crime/horror). Despite this performance,

we need to take into account that genre was the most expensive tag to acquire, as it forced us to join multiple datasets. Considering this cost, the previous paragraph does a fair job at lowering perplexity (6,82), and obtaining it is easy, as it is already in the plot. Adding the outline, while also good for lowering perplexity (7,81), is more expensive than the previous paragraph, as it requires running an extractive summarization algorithm. From this results alone, we can conclude that all three tags improve conditioning, as they all managed to significantly reduce the perplexity of the baseline.

Looking at the results of all models, lower perplexity is achieved by combining all three tags, without sacrificing diversity (Self-BLEU). This is in accordance with the work of Martin et al. ^[68] where they notice an improvement in perplexity as they add more information (events) into the model. ^[68] It is also worth noting that adding one tag to genre did not improve its perplexity (models 4: DGO and model 5: DGP) but still achieves better results than using the other tag by itself. To be more concrete, outline and paragraph by themselves obtained a higher perplexity (7,81 and 6,82) than combining them with genre (7,43 and 6,52 respectively) but didn't manage to get a lower score than using genre alone (5,63). Based on automatic evaluation metrics alone, we could conclude that genre is a strong tag with the capacity of lowering perplexity for other tags. We will see if this holds in the generated samples.

Regrettably, we have no direct paper to compare our perplexity, as PlotMachines does not report the perplexity of the models. While we could look at other papers with a similar task such as Martin et al. with Wikipedia Movie Plots (they achieve a perplexity of 45,23 on their event-to-event model and a score of 53,964 on their event-to-sequence model) ^[68] or the Hierarchical Text Generation of Fan et al ^[2] using WritingPrompts dataset (with a perplexity of 36,08) ^[2] we need to be cautious as perplexity depends on the vocabulary —among others—, making the results not directly comparable. Additionally, the vocabulary of the model is not necessarily the same as the dataset, as it depends on the tokenizer: for our models, we are using GPT-2 's default byte-pair encoding which gives a vocabulary of approximately 50,257 byte-pair tokens ^[23] while Fan et al use NLTK, tokenizing on a word level. For example, the word “tokenization” would be encoded as two tokens in byte-pair (‘token’ and ‘ization’) while it would be only one in NLTK. In simpler terms, we measured our models on how well they predicted the next sub-word while Fan et al

measure on words. Regardless, a perplexity of 5,63 is a satisfactory result and a clear improvement from the original 18,62 baseline.

We compare our baseline, our best individual model and all three multiple tags models with the original PlotMachines^[18] ablation results using ROUGE and Self-BLEU. We would like to remind the reader that we made some changes to the original Wikiplots dataset to avoid data contamination making our results not directly comparable. To put our difference in datasets into context, PlotMachines reported a Self-BLEU of 0,4 for the Gold Test while we have 0,063 in our validation samples, making our dataset more diverse.

Model	PM-GPT2	PM-NoMem-NoDisc	PM-NoMem	Model 0	Model 1	Model 4	Model 5	Model 6
R-1	22.80	19.30	20.50	18.68	19.16	15.90	16.38	14.99
R-2	6.50	1.70	4.90	0.71	0.81	0.74	0.60	0.46
R-L	17.50	13.90	15.50	14.78	15.42	12.81	13.17	12.50

Figure 20. ROUGE results

Our models have a lower ROUGE score, in particular, they have a sharper decrease between R-1 and R-2. While ROUGE by itself is not a good metric for creative story generation, we were expecting a higher value for some of our models, in particular Model 6 since it has the most information (previous paragraph and outline). Surprisingly, other models with less information were able to get better n-gram overlap with the Golden Test. This goes against our initial hypothesis, were we theorized that —based on having an outline and/or the previous paragraph— models with a more detailed prompt would lead to better ROUGE scores as they would be incentivized to add the characters and events into the generation. In order to understand these results, we now explore generated samples.

Model 0 [PROMPT] _start_ _t_ death cloud _i_

After a short visit to his home, a small group of mysterious people meet in a dark room on the outskirts of Paris. It seems that they are part spirits and have some secrets: The Doctor has created a strange device called **Death Cloud** which can be used by all living things if touched by sunlight. He explains it will help all who die because he is able to sense the appearance from one person's eyes and sense its presence using its own special eyes.

Model 6 [PROMPT] _start_ _t_ death cloud _g_ crime, fiction _kw_ On returning home he visits his tutor's home, where he meets Crowe's daughter Virginia. _kw_ He detects a yellow powdery substance around the body and takes a sample of it. _endkw_ _startpv_ NA _endpv_ _i_

Death Cloud is set in 1860s London by an army officer named Professor Oliver Jane Watson (Lily) who happens to have lost important sources at that time due two months earlier on 29 July 1886 but was just returned from recovering leave after a three-month absence for the Continent 2 war with Britain killing out 37 per cent or "generally damaging" 1st Lieutenant Dayal Gamboniludan Moyoaccio(Sergio Paola), whom Oliver himself refers as Button Robesian One day Lucy Bowden Thomas

McLean puts Mabrogba into protective custody instead because Valentine Mac

As we can see from the previous samples, the hyperparameters cause model 6 to output a degenerate solution. To be more concrete, it produces just one sentence that never ends, introduces random characters and ignores the prompt. To prevent this, we change the generation hyperparameters to nucleus sampling of 0,9, k sampling of 50 and a repetition penalty of 1, resulting in a sharp improvement in generations both in terms of fluidity and task fulfillment.

Model 6 [PROMPT] _start_ _t_ death cloud _g_ crime, fiction _kw_ On returning home he visits his tutor's home, where he meets Crowe's daughter Virginia. _kw_ He detects a yellow powdery substance around the body and takes a sample of it. _endkw_ _startpv_ NA _endpv_ _i_

John **Crowe**, a British man with a long-term relationship with his teenage sister **Virginia**, is the subject of a **mysterious black powdery substance called death cloud**. In the town of Whitehall, a young man named John **Crowe** visits his **tutor's home**, where he meets Crowe's daughter Virginia. Crowe is suspicious of this mysterious man and is determined to find out more about him. Crowe and his friend Simon, who lives near Whitehall, are fascinated by **the strange substance**, and decide to investigate it.

While there is still some repetition in place (everyone seems to be called “Crowe”) the text is coherent and it uses information from multiple tags: the title is included (in the form of a powder by the name “death cloud”), the paragraph creates an introduction (presents characters and starts building the story), it is the correct genre (a mystery novel) and it incorporates the characters mentioned in the outline. We show more samples from the generations in the next section. We show below a comparison of all our models using the original generation hyperparameters from PlotMachines (left) and our new hyperparameters (right):

Model	nucleus sampling 0.7 / repetition penalty = 1.4				nucleus sampling 0.9 / top k 50 / repetition penalty = 1			
	R-1	R-2	R-L	B-4	R-1	R-2	R-L	B-4
Model 0	18.68	0.71	14.78	0.06	20.05	1.40	15.10	0.251
Model 1	19.16	0.81	15.42	0.08	19.75	1.56	15.25	0.166
Model 2	15.66	0.44	12.47	0.06	24.87	4.90	19.59	0.258
Model 3	13.82	0.55	11.53	0.05	22.04	1.78	16.25	0.186
Model 4	15.90	0.74	12.81	0.07	24.35	3.92	19.01	0.262
Model 5	16.38	0.60	13.17	0.06	22.04	1.85	16.73	0.177
Model 6	14.99	0.46	12.50	0.07	23.89	3.93	18.55	0.232

Figure 21. Results using different decoding strategies

On a general note, we see improvements in ROUGE at the cost of a drop in diversity (Self-BLEU increases). All models that use outline or the previous paragraph outperform the ROUGE score of the baseline by 2 to 4 points. This aligns with our initial hypothesis: the model has learnt

to incorporate the outline and/or paragraphs into the generation, increasing the match with the golden standard.

As we can see, the generation hyperparameters can have a great impact on the performance of the model. The wrong decoding strategy can lead to the erroneous conclusion that the model has failed in achieving its task. As such, fine-tuning language models require two sets of ablation studies: one for training the model (epochs, learning rate, tags) and another for generating the samples (nucleus sampling, repetition penalty, etc.). This is not trivial when we consider that the end-goal (producing condition stories) still requires human evaluation. No matter how we optimize these hyperparameters, if the resulting text is bad (bland, degenerate, etc.) our model has failed. This highlights the importance of being able to create an evaluation metric which allows us to emulate human judgment, as we could (1) automatically search for the best generation hyperparameters for each model (2) use it during LM training to further optimize the model for our tasks.

4.2 Generated Samples

We present generated samples for three plots (Alice in Wonderland, Death Cloud, Replaceable You) belonging to different genres (family/ fantasy, crime/fiction and comedy/comic-manga) and different discourse types (introduction “_i_”, body “_b_” and conclusion “_c_”). For each prompt, the models are asked to generate three responses, and we pick the best from each. The answer is selected based on the following criteria, in descending importance: (1) Task fulfilment (including the outline, genre-related paragraph, etc.) (2) Fluidity (3) Coherence. For more details, please refer to the previous chapter. In bold we highlight the parts of the generations that fulfill the task requirement. All prompts are marked in blue.

Introduction for Death Cloud

Model 0 [PROMPT] _start_ _t_ death cloud _i_

The story begins in the middle of a snowy field near the border of Canada and the Soviet Union. On the eve of World War II, a woman, Maria (Debra) is a nurse working in a hospital in Paris, and is living in a flat. She is a Russian who works for the Ministry of Health. Maria's father works as a bank assistant. His mother works for the Ministry of Health and is the daughter of an artist named Maria (Maria). Maria is one of the few people who does not have a child

Model 1 [PROMPT] _start_ _t_ death cloud _g_ crime, fiction _i_

In the book, in the year 1910, a man named James, a middle aged man who has been following a series of **murders, is found dead**. In this novel, we see two young, lonely, innocent girls (also known as "the girls"), who live in the country's wealthiest suburbs. James is taken into **custody**, while the girls' father is taken into **custody**. The story then moves to the present, where James is sent to see his mother at her house

Model 2 [PROMPT] _start_ _t_ death cloud _kw_ On returning home he visits his tutor's home, where he meets Crowe's daughter Virginia. _kw_ He detects a yellow powdery substance around the body and takes a sample of it. _endkw_ _i_

John **Crowe** (Tom Cruise) and his girlfriend Mary (Julie Bowen) are in a car when they see a blue "**death cloud**" near them and decide to take the photos. When they go back to the hotel they are attacked by a "**death cloud**" and they are taken to the hospital. **Crowe** is diagnosed with a rare genetic condition and is immediately declared a hero for saving his life. He is sent to see Dr. John Crowe's (John Hemsley) office where he is taken to see Dr. Mary (Julie Bowen) and Crowe's daughter **Virginia**

Model 3 [PROMPT] _start_ _t_ death cloud _startpv_ NA _endpv_ _i_

In the book, in the year 1910, a man named James, a middle aged man who has been following a series of **murders** in his house has committed suicide, while in the midst of the murders, one of the victims, is killed and two other dies. After a year of mourning and the loss of his family, James returns home, not to see the family, but instead to discover the fact that his own family, as well as his friends, are at the same time, haunted by his own own loss.

Model 4 [PROMPT] _start_ _t_ death cloud _g_ crime, fiction _kw_ On returning home he visits his tutor's home, where he meets Crowe's daughter Virginia. _kw_ He detects a yellow powdery substance around the body and takes a sample of it. _endkw_

After a month spent vacation in New Orleans, he receives a call from his **tutor's home** telling him that the **man has died** and that **his wife is dead**. He contacts the **sheriff's department** and learns that **Crowe** is on a hunt for the man, a former boxer named Bill Jones. He learns that Jones' wife **was killed** when he was a young man who had been trying to flee the country and was being used to make a living.

Model 5 [PROMPT] _start_ _t_ death cloud _g_ crime, fiction _startpv_ NA _endpv_ _i_

In the book, in the year 1910, a man named James, a middle aged man who **has been following a series of murders**, is found dead. In this novel, we look forward to a year in which James meets the "Mr. O'Brien" (Andrew Lincoln), who lives with his wife and two children on a rural farmhouse in Texas. He is the most famous farmer in the land and becomes very influential as a result. He has a son named Johnny.

Model 6 [PROMPT] _start_ _t_ death cloud _g_ crime, fiction _kw_ On returning home he visits his tutor's home, where he meets Crowe's daughter Virginia. _kw_ He detects a yellow powdery substance around the body and takes a sample of it. _endkw_ _startpv_ NA _endpv_ _i_

John **Crowe**, a British man with a long-term relationship with his teenage sister **Virginia**, is the subject of a **mysterious black powdery substance called death cloud**. In the town of Whitehall, a young man named John **Crowe** visits his **tutor's home**, where he meets **Crowe's daughter Virginia**. Crowe

is suspicious of this mysterious man and is determined to find out more about him. Crowe and his friend Simon, who lives near Whitehall, are fascinated by **the strange substance**, and decide to investigate it.

Body for Replaceable You

Model 0 [PROMPT] _start_ _t_ replaceable you _b3_

The boys manage to sneak into the office to meet the president, a powerful man with a long-standing rivalry with his superior (and the president himself) in Washington, DC, where he meets with the Chief of Staff, Deputy White House Counseling Officer, and President of the House of Representatives. In an attempt to stop this, the President and his team, including their secretary of state, Anita (a woman in a suit) and her friend Nancy (a former student of hers) are forced into the office.

Model 1 [PROMPT] _start_ _t_ replaceable you _g_ comedy, comic/manga _b3_

At home, the family finds the **dog's belongings empty**, and a letter has been delivered to him, saying "The boy is a wonderful boy". When his grandmother's mother shows up for school, he makes a call to the police, where he gets a call that the dog is in the lake. The police go after him and arrest him, as are his mom. He returns to school, and when he is questioned, he admits he had no problem being an alcoholic, which was why he was so popular with the kids

Model 2 [PROMPT] _start_ _t_ replaceable you _kw_ They succeed and as a result, Chief Wiggum releases all the seals, who return to the nursing home. _kw_ However, it is revealed that when the wiring is tampered with, they become violent attackers. _kw_ When Roz wins a Worker of Millennium award at the plant, Homer manipulates Burns into giving Roz a hug. _endkw_

Burns gives out a large **sum of money** to his co-worker and the company's head honcho, Dr. **Wiggum**. The CEO of the company, John Muntz, orders **Burns** to cut ties with the plant and put him at his home and company. **Homer** demands his life as a substitute teacher and demands that he take a vacation from work, only to discover that his co-worker is not there. He is confronted by the head honcho and Homer tells him to leave, and the company sends Burns to the hospital

Model 3 [PROMPT] _start_ _t_ replaceable you _startpv_ _b2_ Roz then compliments Homer for being much smarter than she expected, in terms Homer does not quite understand. Elsewhere, Bart is not ready for the upcoming Springfield Elementary science fair. He ends up working with Martin Prince and after Bart comes up with a general idea cool robotics Martin does all the work and constructs an adorable robotic baby seal. However, it is revealed that when the wiring is tampered with, they become violent attackers. The seal wins first prize at the Fair, to Lisa's disdain. _endpv_ _b3_

However, **Lisa takes advantage of her talents to win, using the ability to change minds so that everyone would know what was going on with the contest**. However, the school's new mascot, Bucky, is a completely different mascot and features much more of Bart and Homer's personalities. However, the school gets an Emmy Award for Outstanding Science Fiction Design and is chosen to win this award for the first time since the Simpsons and the **school's robot-powered robots**

Model 4 [PROMPT] _start_ _t_ replaceable you _g_ comedy, comic/manga _kw_ They succeed and as a result, Chief Wiggum releases all the seals, who return to the nursing home. _kw_ However, it is revealed that when the wiring is tampered with, they become violent attackers. _kw_ When Roz wins a Worker of Millennium award at the plant, Homer manipulates Burns into giving Roz a hug. _endkw_

However, it is revealed that when the wiring is tampered with, they become violent attackers. He and the others flee to the nursing home. Homer tries to tell the rest of the town, including Roz, but her boss says he's not coming to town and Homer and everyone else are in the nursing home. While doing this, the others begin shooting and killing all their residents. Roz eventually gets into a fight with another worker and a group of people shoot down the nurses and get away

Model 5 [PROMPT]_start_ _t_ replaceable you _g_ comedy, comic/manga _startpv_ _b2_ Roz then compliments Homer for being much smarter than she expected, in terms Homer does not quite understand. Elsewhere, Bart is not ready for the upcoming Springfield Elementary science fair. He ends up working with Martin Prince and after Bart comes up with a general idea cool robotics Martin does all the work and constructs an adorable robotic baby seal. However, it is revealed that when the wiring is tampered with, they become violent attackers. The seal wins first prize at the Fair, to Lisa's disdain._endpv_ _b3_

However, Lisa takes advantage of her own success and, using the ability to change minds, sends Bart and Bart to chase after them with the help of Lisa's friends, Marge and Puck. The final prize is a \$30,000 reward and Bart and Homer are able to save the fair by putting an impromptu seal on the roof. Meanwhile, Homer and his friends manage to win the contest. But before they can compete with Lisa's robot and his friends Homer wins their first prize, and the seal becomes useless.

Model 6 [PROMPT]_start_ _t_ replaceable you _g_ comedy, comic/manga _kw_ They succeed and as a result, Chief Wiggum releases all the seals, who return to the nursing home. _kw_ However, it is revealed that when the wiring is tampered with, they become violent attackers. _kw_ When Roz wins a Worker of Millennium award at the plant, Homer manipulates Burns into giving Roz a hug._endkw_ _startpv_ Roz then compliments Homer for being much smarter than she expected, in terms Homer does not quite understand. Elsewhere, Bart is not ready for the upcoming Springfield Elementary science fair. He ends up working with Martin Prince and after Bart comes up with a general idea cool robotics Martin does all the work and constructs an adorable robotic baby seal. However, it is revealed that when the wiring is tampered with, they become violent attackers. The seal wins first prize at the Fair, to Lisa's disdain._endpv_ _b3_

Homer then uses the opportunity to teach Lisa a lesson on how to control a robot, and the school board of trustees agree that the seal is a good idea. After the school board decides to vote for the school's seal, Bart and Lisa take an interest in the seal, and they all begin to enjoy the experience. Bart, Lisa, and Homer go out shopping and make friends with the other employees, and Homer finds out that the store owner (Roland Mabu) and his wife (Debra Messing) have a problem with it

Conclusion for Alice in Wonderland

Model 0 [PROMPT]_start_ _t_ alice in wonderland _c_

The movie ends with the narrator, Jack "Ethan" Cootman, who says "Well, when you start to live, you must be in good shape", referring to how he once lived, "and he must be there".

Model 1 [PROMPT]_start_ _t_ alice in wonderland _g_ comedy, family, fantasy _c_

The two girls share a hug as the children embrace their respective stories of the past, as the boys watch a cartoon of Alice in wonderland with their parents happily watching it.

Model 2 [PROMPT]_start_ _t_ alice in wonderland _kw_ Strangely, through all of this, the governess does not seem to notice what has happened. _kw_ In a situation exactly reversed from the book, she grows to enormous size after drinking the bottle's contents. _kw_ Unable to pass into the room beyond the locked door, she begins to cry._endkw_ _c_

The only way to escape is through the door and through the door, she begins to cry. The book ends

with the scene of Alice and her parents in the garden, smiling as they happily chat

Model 3 [PROMPT]_start_ _t_ alice in wonderland _startpv_ _b1_ Strangely, through all of this, the governess does not seem to notice what has happened. Alice looks out the window and suddenly sees a White Rabbit. She follows it to a rabbit hole and falls in. Seeing nobody else there, she comes upon a table with a key to a locked door, and a bottle that bears the sign "Drink Me". In a situation exactly reversed from the book, she grows to enormous size after drinking the bottle's contents. Unable to pass into the room beyond the locked door, she begins to cry._endpv_ _c_

The **rabbit enters the room**, and begins to scream. When **Alice** wakes up she notices her baby has been asleep on the table, and the room seems to be empty. The rabbit leaves again, and she is again awakened.

Model 4 [PROMPT]_start_ _t_ alice in wonderland _g_ comedy, family, fantasy _kw_ Strangely, through all of this, the governess does not seem to notice what has happened. _kw_ In a situation exactly reversed from the book, she grows to enormous size after drinking the bottle's contents. _kw_ Unable to pass into the room beyond the locked door, she begins to cry._endkw_ _c_

The only way to **escape** is through the **door** and through the door, it is not revealed where she is when she leaves her cell, but who she is in, and who is not in, and who is on the roof of the palace where she is seen sitting. It is also implied that she does not know who she is or what she is or what she is going to be in for the rest of her life, even if it was a dream. The story ends with the castle doors closing and the door closing, as the sun rises.

Model 5 [PROMPT]_start_ _t_ alice in wonderland _g_ comedy, family, fantasy _startpv_ _b1_ Strangely, through all of this, the governess does not seem to notice what has happened. Alice looks out the window and suddenly sees a White Rabbit. She follows it to a rabbit hole and falls in. Seeing nobody else there, she comes upon a table with a key to a locked door, and a bottle that bears the sign "Drink Me". In a situation exactly reversed from the book, she grows to enormous size after drinking the bottle's contents. Unable to pass into the room beyond the locked door, she begins to cry._endpv_ _c_

The **rabbit** follows her and makes her leave. But she's still unable to stop herself from running, so she throws it over a wall and runs away with **Alice**. **The story ends** with Alice walking back to the house to finish her **drink**.

Model 6 [PROMPT]_start_ _t_ alice in wonderland _g_ comedy, family, fantasy _kw_ Strangely, through all of this, the governess does not seem to notice what has happened. _kw_ In a situation exactly reversed from the book, she grows to enormous size after drinking the bottle's contents. _kw_ Unable to pass into the room beyond the locked door, she begins to cry._endkw_ _startpv_ Strangely, through all of this, the governess does not seem to notice what has happened. Alice looks out the window and suddenly sees a White Rabbit. She follows it to a rabbit hole and falls in. Seeing nobody else there, she comes upon a table with a key to a locked door, and a bottle that bears the sign "Drink Me". In a situation exactly reversed from the book, she grows to enormous size after drinking the bottle's contents. Unable to pass into the room beyond the locked door, she begins to cry._endpv_ _c_

The **rabbit** explains how she was **sent into a world** where no one can have dreams about her. The **book concludes** with the promise that Alice will never see her again, and she is happy that she has found the **key** and returned it to her.

4.3 Analysis of the Samples

In this section we do an in-depth analysis of all the samples from the previous section. For simplicity, we compare them by model.

Model 0 (D): We notice that it has learnt to incorporate the discourse (it successfully creates an introduction, conclusion and body) but it is unclear how the titles are used in the generation. While stories of World War II could be tied to “death cloud” there is no relationship between “Alice in Wonderland” and the presented conclusion.

Model 1 (DG): We start seeing how our prompt steers the generation. When adding crime, it immediately starts with a death followed by someone being taken into custody. It seems the model is also able to incorporate the title, as we see in “Alice in Wonderland”. Additionally, it gives a happy ending as it is expected from the “family” genres. While an alcoholic dog is not the author’s first choice for a comedy, it could be that the model has been exposed to more adult oriented cartoons, where the situation could be presented as funny. Regardless, it looks like a good body paragraph, which requires an introduction and can lead to an event conclusion.

Model 2 (DO): This model incorporates the keywords and titles into the generation: in both “Alice in Wonderland” and “Death Cloud” the model uses the title as part of the result. While it does suffer from some repetition problems (everyone seems to be named “Crowe” or “Burns”) it does a good job at incorporating part of the keywords. This model might benefit from a different decoding strategy to reduce the repetition.

Model 3 (DP): We can see the effect of the previous paragraph in model 3. The best example is “Replaceable You” where it immediately ties to the contest and continues with a cartoon-like event (power to changes everyone’s minds). It is worth mentioning that it seems to be building the paragraphs with more than just information from the last sentence: in “Alice in Wonderland” it brings the Rabbit back, although it was presented at the start of the previous paragraph. The model is also capable of writing without a previous paragraph, as seen in “Death Cloud”.

Model 4 (DGO) and 5 (DGP): Both models seem to be suffering from degeneration. In model 4 “Alice in Wonderland” it keeps repeating the door and philosophical questions, while in Model 5 we get Bart multiple times. This might be because we chose the best hyperparameters based on our study of Model 6. This shows the challenge of comparing the models: we have worsened the generation of some of our models in favor of others. Having an automatic metric that can detect the best generation hyperparameter for each model would allow us to compare each of

them at their peak performance. Under these hyperparameters, model 2 (without genre) produces better generations than model 4 and model 3 (without genre) to model 5.

Model 6 (DGOP): From all experiments, model 6 produces the best result: A coherent follow up to the last paragraph, it stays relevant to genre and includes the keywords (though we still have some coherency problems as Virginia being Crows daughter and sister). As we stated before, this good performance is no coincidence, as we optimized the hyperparameters based on model 6.

As a general note, all models have fulfilled their specified task, conditioning the generation based on the tags. Though some are clearer than others, DistilGPT-2 has proven to be able to learn from the patterns in the data and condition the resulting generation based on the prompt. In particular, we notice that DistilGPT-2 has learnt some interesting representations based on the data: it names a character and gives it an actor such as John Crow (Tom Cruise) and Mary (Julie Bowen). This leads us to believe that we have more movies in the dataset than initially intended. Additionally, despite the fact that the prompt drastically increases from model to model, there seems to be no apparent drawback to it. Model 6, with the largest prompts, is able to produce a coherent follow-up just as good as Model 3 with a shorter prompt.

After reading all the samples, one thing becomes clear: some models produce similar stories, with slight alterations. For example, model 1, 3 and 5 set “Death Cloud” in 1910 and have a middle-aged man “James” as the protagonist. This plot seems to be regardless of the genre or previous paragraph, so we are led to believe that — in absence of a previous paragraph — models 3 and 5 will behave as model 1.

Something similar happens between model 3 and 5, as they both give Lisa mind powers to alter the results of the fair. Additionally, model 2 and 4 start the conclusion for Alice in the same way. This is not completely unexpected: after all, the difference in perplexity between the models is low: model 2 and 4 (both including outline) have a difference in perplexity of 0,38, while model 3 and 5 (both with previous paragraph) have a difference of 0,04.

This leads us to the conclusion that genre, by itself, has a great power in lowering perplexity but loses influence to condition the generation when in presence of other metadata such as outline

and the previous paragraphs. This explains the relatively low drop in perplexity once it has been added to other tags and why models 2-4 and 3-5 produce similar outputs. It should be noted that this is not because the genre uses less tokens than the outline or the previous paragraph: all models are able to condition on the discourse (`_i_`), (`_b_`), (`_c_`) despite it being the tag with the least amounts of tokens.

This reveals the dangers of relying solely on perplexity for selecting the tags. While genre looks promising in our individual experiments, it was quickly outshined by the previous paragraph and the keywords during generation. Both clearly have a more direct relationship with the generated prompt. We conclude that not all tags are equal. Depending on the task some will have a more direct impact on the conditioning. Simply adding tags will not directly lead to better generations and might cause us to incur in unnecessary costs.

4.4 Further studies

We would like to finish this chapter by presenting some possible lines of research to continue building on conditional text generation.

Due to computational resources, we had to work with DistilGPT-2. It would be interesting to repeat the experiment with GPT-2 medium and large to understand the impact a larger model can have in conditioning the generation in terms of produced output, training time and in number of samples needed to reach a low perplexity.

Though we only use genre from the IMDB dataset, it could have been possible to include additional metadata, like author, year of publication, etc. Adding publishing year as a tag might allow models to learn underlying changes in society and literature throughout the years. The role of women, society values, even the importance of religion has changed in the last hundred years and it is being reflected in our literature. By doing so the tag might prevent our generations from falling into the bias we have presented in the Literature Review.

For our experiments we used extractive summarization. It is to be determined how the model would react to abstractive inputs, where the relationship with the generated plot is not as direct. It is also unclear at the moment if using full sentences (taken using extractive

summarization) outperforms a keyword outline (built using RAKE ^[94]). This would be an interesting follow-up experiment.

Finally, since our dataset was fragmented to fit previous paragraphs and outlines of various lengths, we believe that our models might have suffered from context fragmentation. As such, we might have made the task more difficult for models with outlines, as the keywords would not always be present in the plot being analyzed. This could be solved by using a model with memory (like transformers-XL^[30]) or by setting a maximum length for the outline such that all the plot (introduction, body and conclusion) could fit in the 1024 tokens of DistilGPT-2. It is yet to be determined the effect this could have in the quality of the conditioned generation. Another possible way to tackle this problem would be to build the outline so that it has three parts, one for each discourse type. By doing so, there would always be a relationship between the outline and the current plot being analyzed.

Chapter 5

Conclusion

Natural Language Generation has come a long way since the times of TALE-SPIN^[65] and Automatic Novel writer^[66]. From rule-based methods to Transformer Models, we went from algorithms that required the programmer to be well versed in linguistics to models capable of learning semantic representations by themselves. In particular, these last two years have been crucial to NLG, as the new paradigm of Transfer Learning was introduced, and large transformer-based language models were made publicly available. While there are still fears about its misuse and concerns about biases, the progress these models have brought to the field are undeniable.

In our work we have aimed at better understanding conditional language models for story generation. We have done an in-depth study showing how different metadata can be used to condition the generation. Particularly, we have illustrated how genre, discourse, outline, previous paragraph and title can be used to condition the resulting story. For our work we introduced an expanded dataset, Wikiplots 2, which brought the genre to the existing Wikiplots dataset and reduced data contamination.

Unlike other approaches, we based our study on both quantitative metrics (perplexity, ROUGE, Self-BLEU) and qualitative evaluations (human evaluation) to understand how the metadata conditioned the results. By doing so we were able to determine that genre, by itself, has a great capacity to lower perplexity but it is quickly replaced by other metadata when combining them. We have also shown that discourse is a simple tag that language models can use to further condition the generation, being always present in the resulting text. Additionally, we have seen how the previous paragraph can lower perplexity and create fluid connections between sentences.

Our final model is able to create coherent and fluid texts, conditioned on both the previous paragraph, outlines and discourse, adapting to the user's preferences. By doing so we show that GPT-2 does not need any architectural change to effectively control the generation. Our work also proves that GPT-2 can take long prompts with no apparent damage to the resulting generation.

Today, it is still unclear how GPT-3 will revolutionize the field. While the generated samples look promising, the model is not open to the public and its future remains unknown. The effectiveness of GPT-3's "short learner" might induce a new paradigm, where we no longer fine-tune our models. Should GPT-3 not be made public there should be no source for concern, as current language models are more than powerful enough to be trained on new tasks. We believe that we have still to reach a master understanding of GPT-2 and its capacities.

We would like to turn our attention, once again, to evaluation metrics. The field of open generation (stories, articles) is suffering from a lack of reliable metrics. Should a metric that mimics human judgment be developed, it would remove the bottleneck creative generation is suffering, and it will lead to a new wave of research. Most importantly, it would allow models to be comparable, allowing for a better understanding of their capacities.

As final thoughts, we would like to discuss the broader impact of this line of work and, in particular, the effect of having strong conditioning text generation could have on related jobs. We can not deny that there has been an increase in the usage of artificial intelligence for content creation. ^[53] When it comes to story generation, we believe that writers should not be alarmed. Though the generation capacities are improving (and will continue to do so) a fundamental part of any story is the way it is told and how the author breathes life into their characters and the world throughout their perspective and unique thoughts. Anyone (even algorithms) can write about an orphan boy that goes to a magical school, but only true authors can turn this simple plot into memorable books like Harry Potter or The Name of the Wind.

Bibliography

- [1] Holtzman, A., Buys, J., Du, L., Forbes, M., & Choi, Y. (2019). The curious case of neural text degeneration. arXiv preprint arXiv:1904.09751.
- [2] Fan, A., Lewis, M., & Dauphin, Y. (2018). Hierarchical neural story generation. arXiv preprint arXiv:1805.04833.
- [3] Liu, P. J., Saleh, M., Pot, E., Goodrich, B., Sepassi, R., Kaiser, L., & Shazeer, N. (2018). Generating wikipedia by summarizing long sequences. arXiv preprint arXiv:1801.10198.
- [4] Keskar, N. S., McCann, B., Varshney, L. R., Xiong, C., & Socher, R. (2019). Ctrl: A conditional transformer language model for controllable generation. arXiv preprint arXiv:1909.05858.
- [5] Xiao, D., Zhang, H., Li, Y., Sun, Y., Tian, H., Wu, H., & Wang, H. (2020). ERNIE-GEN: An Enhanced Multi-Flow Pre-training and Fine-tuning Framework for Natural Language Generation. arXiv preprint arXiv:2001.11314.
- [6] Dathathri, S., Madotto, A., Lan, J., Hung, J., Frank, E., Molino, P., ... & Liu, R. (2019). Plug and play language models: a simple approach to controlled text generation. arXiv preprint arXiv:1912.02164.
- [7] Dong, L., Yang, N., Wang, W., Wei, F., Liu, X., Wang, Y., ... & Hon, H. W. (2019). Unified language model pre-training for natural language understanding and generation. In *Advances in Neural Information Processing Systems* (pp. 13063-13075).
- [8] Caccia, M., Caccia, L., Fedus, W., Larochelle, H., Pineau, J., & Charlin, L. (2018). Language gans falling short. arXiv preprint arXiv:1811.02549.
- [9] Verga, P., Sun, H., Soares, L. B., & Cohen, W. W. (2020). Facts as Experts: Adaptable and Interpretable Neural Memory over Symbolic Knowledge. arXiv preprint arXiv:2007.00849.
- [10] Zellers, R., Holtzman, A., Rashkin, H., Bisk, Y., Farhadi, A., Roesner, F., & Choi, Y. (2019). Defending against neural fake news. In *Advances in Neural Information Processing Systems* (pp. 9054-9065).
- [11] Beltagy, I., Peters, M. E., & Cohan, A. (2020). Longformer: The long-document transformer. arXiv preprint arXiv:2004.05150.
- [12] Zhang, Y., Wang, G., Li, C., Gan, Z., Brockett, C., & Dolan, B. (2020). POINTER: Constrained Text Generation via Insertion-based Generative Pre-training. arXiv preprint arXiv:2005.00558.
- [13] Deng, Y., Bakhtin, A., Ott, M., Szlam, A., & Ranzato, M. A. (2020). Residual energy-based models for text generation. arXiv preprint arXiv:2004.11714.
- [14] Wang, A., & Cho, K. (2019). Bert has a mouth, and it must speak: Bert as a markov random field language model. arXiv preprint arXiv:1902.04094.
- [15] Celikyilmaz, A., Clark, E., & Gao, J. (2020). Evaluation of Text Generation: A Survey. arXiv preprint arXiv:2006.14799.
- [16] Zhou, W., & Xu, K. (2020). Learning to Compare for Better Training and Evaluation of Open Domain Natural Language Generation Models. In *AAAI* (pp. 9717-9724).
- [17] Sellam, T., Das, D., & Parikh, A. P. (2020). BLEURT: Learning Robust Metrics for Text Generation. arXiv preprint arXiv:2004.04696.
- [18] Rashkin, H., Celikyilmaz, A., Choi, Y., & Gao, J. (2020). PlotMachines: Outline-Conditioned Generation with Dynamic Plot State Tracking. arXiv preprint arXiv:2004.14967.

- [19] Tan, B., Yang, Z., AI-Shedivat, M., Xing, E. P., & Hu, Z. (2020). Progressive Generation of Long Text. arXiv preprint arXiv:2006.15720.
- [20] Févry, T., Soares, L. B., FitzGerald, N., Choi, E., & Kwiatkowski, T. (2020). Entities as experts: Sparse memory access with entity supervision. arXiv preprint arXiv:2004.07202.
- [21] Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., ... & Agarwal, S. (2020). Language models are few-shot learners. arXiv preprint arXiv:2005.14165.
- [22] Radford, A., Narasimhan, K., Salimans, T., & Sutskever, I. (2018). Improving language understanding by generative pre-training.
- [23] Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., & Sutskever, I. (2019). Language models are unsupervised multitask learners. OpenAI Blog, 1(8), 9.
- [24] Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., ... & Stoyanov, V. (2019). Roberta: A robustly optimized bert pretraining approach. arXiv preprint arXiv:1907.11692.
- [25] Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Salakhutdinov, R. R., & Le, Q. V. (2019). Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in neural information processing systems* (pp. 5753-5763).
- [26] Eyal, M., Baumel, T., & Elhadad, M. (2019). Question answering as an automatic evaluation metric for news article summarization. arXiv preprint arXiv:1906.00318.
- [27] Liu, Y. (2019). Fine-tune BERT for extractive summarization. arXiv preprint arXiv:1903.10318.
- [28] Riedl, M. (2017). Markriedl/WikiPlots Retrieved from, github.com/markriedl/WikiPlots.
- [29] Charis, J. J. (2018) Text Summarization Using SpaCy and Python. Retrieved from jcharistech.wordpress.com/2018/12/31/text-summarization-using-spacy-and-python/.
- [30] Dai, Z., Yang, Z., Yang, Y., Carbonell, J., Le, Q. V., & Salakhutdinov, R. (2019). Transformer-xl: Attentive language models beyond a fixed-length context. *arXiv preprint arXiv:1901.02860*.
- [31] Al-Rfou, R., Choe, D., Constant, N., Guo, M., & Jones, L. (2019). Character-level language modeling with deeper self-attention. In *Proceedings of the AAAI Conference on Artificial Intelligence* (Vol. 33, pp. 3159-3166).
- [32] Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., & Zettlemoyer, L. (2018). Deep contextualized word representations. arXiv preprint arXiv:1802.05365.
- [33] Robischon, J. (2018) Wikipedia Movie Plots, Retrieved from www.kaggle.com/jrobischon/wikipedia-movie-plots.
- [34] Bamman, D. (n.d.). CMU Book Summary Dataset. Retrieved from <http://www.cs.cmu.edu/~dbamman/booksummaries.html>
- [35] IMDb Datasets. (n.d.). Retrieved from <http://www.imdb.com/interfaces/>
- [36] IMDb. (2020, December 09). Retrieved October 16, 2020, from <https://en.wikipedia.org/wiki/IMDb>
- [37] Saha, Aveek. (2019, October 31) Finding the Main Characters in a Novel. Retrieved October,16,2020 from home.aveek.io/blog/post/finding-main-characters/.
- [38] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. In *Advances in neural information processing systems* (pp. 5998-6008).
- [39] Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., ... & Zettlemoyer, L. (2019). Bart:

Denosing sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*.

[40] Wilcox, E., Levy, R., & Futrell, R. (2019). What Syntactic Structures block Dependencies in RNN Language Models?. *arXiv preprint arXiv:1905.10431*.

[41] Sutskever, I., Martens, J., & Hinton, G. E. (2011). Generating text with recurrent neural networks. In *ICML*.

[42] Ajao, O., Bhowmik, D., & Zargari, S. (2018). Fake news identification on twitter with hybrid cnn and rnn models. In *Proceedings of the 9th international conference on social media and society* (pp. 226-230).

[43] Lin, C. Y. (2004). Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out* (pp. 74-81).

[44] Zhu, Y., Lu, S., Zheng, L., Guo, J., Zhang, W., Wang, J., & Yu, Y. (2018). Tegygen: A benchmarking platform for text generation models. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval* (pp. 1097-1100).

[45] Nenkova, A. (2006). Summarization evaluation for text and speech: issues and approaches. In *Ninth International Conference on Spoken Language Processing*.

[46] Lowe, R., Noseworthy, M., Serban, I. V., Angelard-Gontier, N., Bengio, Y., & Pineau, J. (2017). Towards an automatic turing test: Learning to evaluate dialogue responses. *arXiv preprint arXiv:1708.07149*.

[47] Lowe, Ryan. (2019, July 30). A Retrospective for ‘Towards an Automatic Turing Test - Learning to Evaluate Dialogue Responses.’. Retrieved November 5, 2020, from [ml-retrospectives.github.io/published_retrospectives/2019/adem/](https://retrospectives.github.io/published_retrospectives/2019/adem/).

[48] Kané, H., Kocyigit, Y., Ajanoh, P., Abdalla, A., & Coulibali, M. (2019). Towards Neural Similarity Evaluator. In *Workshop on Document Intelligence at NeurIPS 2019*.

[49] Hashimoto, T. B., Zhang, H., & Liang, P. (2019). Unifying human and statistical evaluation for natural language generation. *arXiv preprint arXiv:1904.02792*.

[50] Nadeem, M., Bethke, A., & Reddy, S. (2020). StereoSet: Measuring stereotypical bias in pretrained language models. *arXiv preprint arXiv:2004.09456*.

[51] Sheng, E., Chang, K. W., Natarajan, P., & Peng, N. (2019). The woman worked as a babysitter: On biases in language generation. *arXiv preprint arXiv:1909.01326*.

[52] Radford, Alec, et al. (2019, February 14). Better Language Models and Their Implications. Retrieved on August 17, 2020 from openai.com/blog/better-language-models/.

[53] Carlson, M. (2015). The robotic reporter: Automated journalism and the redefinition of labor, compositional forms, and journalistic authority. *Digital journalism*, 3(3), 416-431.

[54] Oremus, Will. (2014, March 17) The First News Report on the L.A. Earthquake Was Written by a Robot. Retrieved November 07, 2020 from slate.com/technology/2014/03/quakebot-los-angeles-times-robot-journalist-writes-article-on-la-earthquake.html.

[55] WashPostPR. (2019). The Washington Post leverages automated storytelling to cover high school football. Retrieved November 07, 2020, from <https://www.washingtonpost.com/pr/wp/2017/09/01/the-washington-post-leverages-heliograf-to-cover-high-school-football/>

[56] Sutskever, I., Vinyals, O., & Le, Q. V. (2014). Sequence to sequence learning with neural networks. In *Advances in neural information processing systems* (pp. 3104-3112).

[57] Weng, L. (2018). Attention? Attention! Retrieved from <https://lilianweng.github.io/lil-log/2018/06/24/attention-attention.html>

- [58] Loginova, K. (2018). Attention in NLP. Retrieved from <https://medium.com/@joealato/attention-in-nlp-734c6fa9d983>
- [59] Petroni, F., Rocktäschel, T., Lewis, P., Bakhtin, A., Wu, Y., Miller, A. H., & Riedel, S. (2019). Language models as knowledge bases?. *arXiv preprint arXiv:1909.01066*.
- [60] 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. *arXiv preprint arXiv:1910.10683*.
- [61] Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.
- [62] Peng, N., Ghazvininejad, M., May, J., & Knight, K. (2018). Towards controllable story generation. In *Proceedings of the First Workshop on Storytelling* (pp. 43-49).
- [63] Yao, L., Peng, N., Weischedel, R., Knight, K., Zhao, D., & Yan, R. (2019, July). Plan-and-write: Towards better automatic storytelling. In *Proceedings of the AAAI Conference on Artificial Intelligence* (Vol. 33, pp. 7378-7385).
- [64] Fan, A., Lewis, M., & Dauphin, Y. (2019). Strategies for structuring story generation. *arXiv preprint arXiv:1902.01109*.
- [65] Meehan, J. R. (1977, August). TALE-SPIN, An Interactive Program that Writes Stories. In *Ijcai* (Vol. 77, pp. 91-98).
- [66] Klein, S., Aeschlimann, J. F., Balsiger, D. F., Converse, S. L., Foster, M., Lao, R., ... & Smith, J. (1973). *Automatic novel writing: A status report*. University of Wisconsin-Madison Department of Computer Sciences.
- [67] Gervás, P. (2012). Story Generator Algorithms. Retrieved November 12, 2020, from http://lhn.sub.uni-hamburg.de/index.php/Story_Generator_Algorithms.html
- [68] Martin, L. J., Ammanabrolu, P., Wang, X., Hancock, W., Singh, S., Harrison, B., & Riedl, M. O. (2017). Event representations for automated story generation with deep neural nets. *arXiv preprint arXiv:1706.01331*.
- [69] Dehn, N. (1981, August). Story Generation After TALE-SPIN. In *IJCAI* (Vol. 81, pp. 16-18).
- [70] Lebowitz, M. (1983). Creating a story-telling universe.
- [71] Turner, S. R. (1994). MINSTREL: A computer model of creativity and storytelling.
- [72] PÉrez, R. P. Y., & Sharples, M. (2001). MEXICA: A computer model of a cognitive account of creative writing. *Journal of Experimental & Theoretical Artificial Intelligence*, 13(2), 119-139.
- [73] Theune, M., Faas, S., Nijholt, A., & Heylen, D. (2003). The virtual storyteller: Story creation by intelligent agents. In *Proceedings of the Technologies for Interactive Digital Storytelling and Entertainment (TIDSE) Conference* (Vol. 204215, p. 116).
- [74] Li, B., Lee-Urban, S., Johnston, G., & Riedl, M. (2013). Story Generation with Crowdsourced Plot Graphs. In *AAAI*.
- [75] Alammar, J. (2018). The Illustrated Transformer. Retrieved November 15, 2020, from <http://jalammar.github.io/illustrated-transformer/>
- [76] Brownlee, J. (2017). What Are Word Embeddings for Text? Retrieved November 16, 2020, from <https://machinelearningmastery.com/what-are-word-embeddings/>
- [77] CodeEmporium. (2020). Transformer Neural Networks - EXPLAINED! (Attention is all you need). Retrieved November 15, 2020, from <https://www.youtube.com/watch?v=TQQIzhbC5ps>

- [78] Li, J. (2020). Transformer-XL Review: Beyond Fixed-Length Contexts. Retrieved November 16, 2020, from <https://towardsdatascience.com/transformer-xl-review-beyond-fixed-length-contexts-d4fe1d6d3c0e>
- [79] Brownlee, J. (2018). Encoder-Decoder Recurrent Neural Network Models for Neural Machine Translation. Retrieved 16 November 2020, from <https://machinelearningmastery.com/encoder-decoder-recurrent-neural-network-models-neural-machine-translation/>
- [80] Alammari, J. (2019). The Illustrated GPT-2 (Visualizing Transformer Language Models). Retrieved 16 November 2020, from <http://jalammar.github.io/illustrated-gpt2/>
- [81] Dodge, J., Ilharco, G., Schwartz, R., Farhadi, A., Hajishirzi, H., & Smith, N. (2020). Fine-tuning pretrained language models: Weight initializations, data orders, and early stopping. arXiv preprint arXiv:2002.06305.
- [82] Radiya-Dixit, E., & Wang, X. (2020). How fine can fine-tuning be? Learning efficient language models. arXiv preprint arXiv:2004.14129.
- [83] Beltagy, I., Lo, K., & Cohan, A. (2019). SciBERT: A pretrained language model for scientific text. arXiv preprint arXiv:1903.10676.
- [84] Jiang, H., He, P., Chen, W., Liu, X., Gao, J., & Zhao, T. (2019). Smart: Robust and efficient fine-tuning for pre-trained natural language models through principled regularized optimization. arXiv preprint arXiv:1911.03437.
- [85] Peng, X., Li, S., Frazier, S., & Riedl, M. (2020). Fine-Tuning a Transformer-Based Language Model to Avoid Generating Non-Normative Text. arXiv preprint arXiv:2001.08764.
- [86] Liu, R., Dathathri, S., Madotto, A., Molino, P., & Yosinski, J. (2019). Controlling Text Generation with Plug and Play Language Models. Retrieved 17 November 2020, from <https://eng.uber.com/pplm/>
- [87] Huggingface. (n.d.). Huggingface/transformers. Retrieved November 18, 2020, from <https://github.com/huggingface/transformers>
- [88] Von Platen, P. (2020). How to generate text: Using different decoding methods for language generation with Transformers. Retrieved November 19, 2020, from <https://huggingface.co/blog/how-to-generate>
- [89] He, T., Zhang, J., Zhou, Z., & Glass, J. (2019). Quantifying exposure bias for neural language generation. arXiv preprint arXiv:1905.10617.
- [90] Wu, Q., Li, L., & Yu, Z. (2020). TextGAIL: Generative Adversarial Imitation Learning for Text Generation. arXiv preprint arXiv:2004.13796.
- [91] Sanh, V. (2019). Smaller, faster, cheaper, lighter: Introducing DistilBERT, a distilled version of BERT. Retrieved November 24, 2020, from <https://medium.com/huggingface/distilbert-8cf3380435b5>
- [92] DistilGPT-2 model checkpoint. (2020). Retrieved 24 November 2020, from <https://transformer.huggingface.co/model/distil-gpt2>
- [93] NLP From Scratch: Translation with a Sequence to Sequence Network and Attention — PyTorch Tutorials 1.7.0 documentation. (2020). Retrieved 27 November 2020, from https://pytorch.org/tutorials/intermediate/seq2seq_translation_tutorial.html
- [94] Rose, S., Engel, D., Cramer, N., & Cowley, W. (2010). Automatic keyword extraction from individual documents. Text mining: applications and theory, 1, 1-20.
- [95] Jozefowicz, R., Vinyals, O., Schuster, M., Shazeer, N., & Wu, Y. (2016). Exploring the limits of language modeling. arXiv preprint arXiv:1602.02410.
- [96] Kapadia, S. (2019). Language Models: N-Gram. Retrieved November 30, 2020, from <https://towardsdatascience.com/introduction-to-language-models-n-gram-e323081503d9>

[97] See A. (2019) Natural Language Processing with Deep Learning. Personal Collection of See Abigail, Stanford University, California , from <https://web.stanford.edu/class/cs224n/slides/cs224n-2019-lecture06-rnnlm.pdf>

[98] Campagnola, C. (2020). Perplexity In Language Models. Retrieved 30 November 2020 from <https://towardsdatascience.com/perplexity-in-language-models-87a196019a94>

[99] Girling, E. (2020). Everything GPT-2: 4. Data Preparation. Retrieved 4 December 2020, from <https://rowlando13.medium.com/everything-gpt-2-4-data-preparation-514cb62f9f3b>

[100] Guo, B., Wang, H., Ding, Y., Hao, S., Sun, Y., & Yu, Z. (2019). c-TextGen: Conditional Text Generation for Harmonious Human-Machine Interaction. arXiv preprint arXiv:1909.03409.

Appendix A - Wikiplots

Samples from the Wikiplots dataset.^[28] In blue we highlight the exact moment where the summary distinguishes between the different sources (books, movies, tv shows, video games, etc.). It is to be noted that not all plots reference the original source. We present examples for both cases

[Video-Games]: Valkyrie Profile: Covenant of the Plume

Valkyrie Profile: Covenant of the Plume is based loosely on Norse mythology. The roots of the tree Yggdrasil contain the three realms of the world: Asgard, home of the Æsir and Vanir, Midgard, the realm of humans, and Niflheim, the realm of the dead where sinners suffer an eternity of torment by the death goddess Queen Hel. The souls of noble warriors who fall in combat are collected by Valkyries and brought to the Hall of Valhalla as einherjar to aid the Æsir in their struggle against the Vanir. [The game takes place mostly in](#) Midgard in the kingdom of Artolia, located between the neighboring nations of Villnore and Crell Monferaigne....

[Video-Games]: Super Mario Galaxy

Shortly after Mario is invited to the centennial Star Festival by Princess Peach to celebrate the comet that passes overhead, Bowser invades the Mushroom Kingdom with a surprise attack in a fleet of airships. Summoning a gigantic flying saucer, Peach's entire castle is removed from its foundations and is lifted into outer space. Mario is still at the castle's base until Kamek, one of Bowser's minions, launches Mario onto a small planet with his magic. On the planet, he meets an enchantress named Rosalina and her companions, the Lumas. Rosalina is a watcher of the stars, who uses the Comet Observatory to travel across the universe. However, Bowser has stolen all of the Power Stars that act as the Observatory's power source, rendering it immobile. Bestowed with the power to travel through space through one of the Lumas, Mario sets off on a journey across the universe to reclaim the Power Stars and restore power to Rosalina's observatory. Along the way, he finds friends from the Mushroom Kingdom such as Luigi and the Toads. Upon collecting enough Power Stars, the Comet Observatory regains the power to transform into a comet, and flies to the centre of the universe, where Bowser is holding Peach captive. Confronting Bowser, Mario learns that his plan is to rule the entire universe with Peach at his side, using a newly constructed sun of his own via the power of the Grand Stars.

Mario manages to defeat Bowser and free Peach, but Bowser's sun collapses into itself, becoming a supermassive black hole that begins consuming everything nearby. All of Rosalina's Lumas jump into the black hole to destroy it, but sacrifice themselves in the process. The black hole collapses into a singularity and explodes in a supernova. Rosalina appears to Mario as a giantess, revealing that dying stars are later reborn as new stars. Mario awakens in the restored Mushroom Kingdom, full with all of the creatures he had met in the galaxies, alongside Peach and Bowser, celebrating the new galaxy that has emerged in the skies.

[Movies]: Beethoven's 4th

[The movie starts](#) with a dad checking on his two kids before school. Their family includes a girl, boy, artist father, and campaigning mother, and a dog named Beethoven, who they are keeping for a relative

family. The children love Beethoven, but their parents do not, and want to get rid of him. So the children begin taking the dog to Obedience Training led by a former army sergeant. Brennan falls in love with a girl (Hayley), while Beethoven literally destroys the obstacle course in one day...

[Movies]: Tangled

Long ago, a drop of sunlight became a golden flower capable of healing illness, decay, and injury. Hundreds of years later, the flower is still used by Mother Gothel to retain her youth, until soldiers from nearby kingdom, Corona, steal it to heal their ailing queen. Shortly afterwards, the Queen gives birth to Princess Rapunzel. While attempting to recover the flower, Gothel discovers that Rapunzel's hair contains the flower's healing properties, and that cutting her hair destroys its power. Gothel takes the baby and raises her as her own daughter in an isolated tower. Once a year, the King and Queen release sky lanterns on Rapunzel's birthday, hoping for their daughter's return. On the eve of her 18th birthday, Rapunzel requests to leave the tower and discover the source of the lanterns, but Gothel refuses, claiming that the outside world is a dangerous place. Meanwhile, thief Flynn Rider steals Rapunzel's crown from the kingdom and inadvertently discovers the tower after ditching his allies, the Stabington brothers. Rapunzel captures Flynn and discovers the crown, but is unaware of its significance. Rapunzel plans to take Flynn to Gothel to prove she can take care of herself, but Gothel becomes angry. Rapunzel then asks for a special paint that will take Gothel three days' round trip to obtain and return, and Gothel accepts ... As Eugene is sentenced to death, the Duckling regulars help him escape. He is then carried back to Gothel's tower by Maximus. Eugene enters by climbing Rapunzel's hair, only to find Rapunzel bound with chains and gagged with a handkerchief. Gothel stabs Eugene and attempts to escape with Rapunzel, but Rapunzel agrees to lifelong captivity if she is allowed to heal Eugene. As Eugene slowly dies, he slices off Rapunzel's hair, destroying its magic and rapidly killing Gothel as she disintegrates into dust. A heartbroken Rapunzel grieves for Eugene. However, her tears, which still contain a bit of the sun's power, lands on his cheek and restores his life. The two return to the kingdom and Rapunzel reunites with her parents. The kingdom breaks out in celebration, and Eugene is pardoned for his crimes. Rapunzel and Eugene eventually marry.

[TV-Show] : Swabhimaan

Swabhimaan divulges the story of an attractive woman — Svetlana — who finds herself in a battle where there are no real winners. Insecurity, suspicion and fear threaten to erode her vivacious spirit as she struggles to come to terms with her position - that of a pampered mistress whose tycoon patron Keshav Malhotra (Naasir Abdulah) dies leaving her to cope with the ugly aftermath of the tragedy: inheritance wars, succession rights, property entanglements, petty quarrels and above all, emotional turmoil that threatens to destroy her. This serial was aired in the year 1995 to 1997. [This was the first Indian TV show to complete 500 episodes.](#) The show ended on the final episode, where most villains getting killed or jailed.

[TV-Show]: Belly Full of Turkey (How I Met Your Mother)

It's Thanksgiving time, and Marshall and Lily visit Marshall's family in St. Cloud, Minnesota. Marshall plays a game of "bask-ice-ball" (a no-rules, brawling combination of basketball and ice hockey that the Eriksen family invented) with his father and elder brothers while Lily helps his sister-in-law and mother in the

kitchen. The talk turns to babies, and Mrs. Eriksen reveals that her first-born son was almost twelve pounds at birth ("The doctor thought he was twins.") ... Meanwhile, Robin and Ted have no plans for Thanksgiving, and decide to help out at a soup kitchen. When they arrive, they are shocked to find Barney volunteering there already, and that he is highly regarded as one of the best volunteers on staff. The organizer tells Ted and Robin that they have enough volunteers, but after being vouched for by Barney, they are allowed to help... Another stripper named Tracy compliments Ted on his generosity, and Future Ted tells his kids "And that's the story of how I met your mother", shocking them before revealing, to their relief, that he is joking.

Appendix B - Additional Datasets

Example of duplicated title (same plot - BSW)

21624439 /m/05n03q2 Mr. Monk Goes to the Firehouse Lee Goldberg 2006-01-03 {"m/02n4kr": "Mystery"} A woman falls asleep while watching TV and a lit cigarette sets her house on fire. Minutes later, at a nearby firehouse, a firefighter is killed and Monk is blinded in a bizarre attack. Monk must use his other senses to find the killer.

3698358 /m/09w07k Mr. Monk Goes to the Firehouse Lee Goldberg 2006-01-03 {"m/02n4kr": "Mystery"} Adrian Monk and Natalie Teeger stop by the University of California to investigate an open-and-shut self-defense murder. Professor Jeremiah Cowan was giving a class when a gunman burst into the room and pointed a gun at him. Cowan shot the intruder before the intruder could get a shot. The shooter's name is Ford Oldman, who apparently made several threats at Cowan in the past. Monk explains to Captain Stottlemeyer and Lieutenant Disher how Cowan staged the scene, but before he can explain Cowan's motive, Natalie cuts him off, since the department hasn't paid Monk for his consulting. Later that day, Stottlemeyer calls Natalie to say that he has Monk's check at the station. Arriving at the station, they notice that the San Francisco Police

Example of duplicated title (different plot - BSW)

9075044 /m/027wr5m The Immortals James Gunn 1996-05-01 {"m/06n90": "Science Fiction", "m/014dfn": "Speculative fiction", "m/01hmnh": "Fantasy", "m/02x1f": "Fiction"} When a cure for AIDS turns out to be more virulent than the disease, the U.S. establishes quarantine camps in the desert southwest. Michael Barris, a TV producer, masquerades as one of the infected and travels to the camps in search of his son. He finds horrific conditions, and learns that the so-called quarantine camps are death camps where the infected are gathered, purposefully brutalized, and ultimately cremated alive, their ashes bulldozed into the desert sand. Barris's son escapes the camp before the cycle of immolation, carrying the evidence he needs to expose the governmental mis-information campaign.

21899586 /m/05p3w6_ The Immortals Chris Riddell 2009-02-05 {"m/01hmnh": "Fantasy", "m/03mfnc": "Young adult literature"} The story is set approximately 500 years after Freeglader. The Edge is much different from previous novels, with the advent of the Third Age of Flight, using stormphrax crystals as a source of power (stormphrax is highly volatile, gaining weight when in darkness and becoming unstable when in light. Twilight is the level of light needed for neutrality.). Three main settlements have arisen in the Deepwoods: Great Glade, Hive, and Riverrise. The protagonist is Nate Quarter, a lowly miner of phraxcrystals. Nate's father was the past mine sergeant before he died in a suspicious accident involving Grint Grayle, the present mine sergeant. Grayle is corrupt and only thinks about lining his own pockets. He doesn't care whether the people in his mine live or die.

Example of duplicated title (same plot - WMD)

1986,Srinivasa Kalyanam,Telugu,Kodi Ramakrishna,"Daggubati Venkatesh, Gouthami, Bhanupriya",unknown,https://en.wikipedia.org/wiki/Srinivasa_Kalyanam,"Srinivas (Venkatesh) an educated unemployed guy and his sister Lakshmi (Varalakshmi) lost their parents in childhood for saving a landlord Sridhar Rao (Vankayala) and stay with their uncle Nagayah (Suthi Velu) who is a chronic drinker, takes out all money given by Sridhar Rao and sells them. Srinivas's cousin Saroja (Gouthami), daughter of Nagayah helps them to get away from that place and she stays back. They are adopted by a couple (Gollapudi Maruthi Rao & Y.Vijaya) in the city. Srinivas does all kinds of jobs like newspaper delivery boy, auto driver and runs a dance school, where he gets introduced to Lalitha (Bhanupriya) who will be looking for a job staying at her sister's (Anitha) house and her brother-in-law Yenimdakula Venkatrao (Prasad Babu) who is a bad person. Srinivas gives Lalitha a job in his dance school and both of them fall in love. Meanwhile, Saroja changes her name to Swapna and comes to rent in Srinivas house not knowing that he's her cousin. Bokka Lambodharam (Mohan Babu) proprietor of a 5 Star hotel, is a big womanizer. Incidentally, Srinivas uncle Nagayah works as an assistant to him as Nisachara. He sees Lalitha in a dance program and is attracted to her. After a few days, Swapna knows the truth which restarts her childhood love with Srinivas. At the same time, Bokka Lambodharam traps Lalitha's brother-in-law Venkatrao to get her. To protect her, Srinivas brings her home and tells her that he wants to marry her. After listening to that Swapna sacrifices her love, while they are making marriage arrangements Lalitha also knows the truth so decides to go away by accompanying Swapna with Srinivas. She takes help of Bokka Lambodharam to stop the marriage, he agrees to that if she accepts his condition. Meanwhile, Swapna listens to everything and goes in place of Lalitha. Now Lalitha reveals the truth to Srinivas that Swapna is only Saroja, even Nagayah also recognizes his daughter. Srinivas goes and protects Swapna. Finally, Swapna/Saroja makes the marriage of Srinivas & Lalitha and she also stays along with them."

1987,Srinivasa Kalyanam,Telugu,Kodi Ramakrishna,"Daggubati Venkatesh, Gouthami, Bhanupriya",unknown,https://en.wikipedia.org/wiki/Srinivasa_Kalyanam,"Srinivas (Venkatesh) an educated unemployed guy and his sister Lakshmi (Varalakshmi) lost their parents in childhood for saving a landlord Sridhar Rao (Vankayala) and stay with their uncle Nagayah (Suthi Velu) who is a chronic drinker, takes out all money given by Sridhar Rao and sells them. Srinivas's cousin Saroja (Gouthami), daughter of Nagayah helps them to get away from that place and she stays back. They are adopted by a couple (Gollapudi Maruthi Rao & Y.Vijaya) in the city. Srinivas does all kinds of jobs like newspaper delivery boy, auto driver and runs a dance school, where he gets introduced to Lalitha (Bhanupriya) who will be looking for a job staying at her sister's (Anitha) house and her brother-in-law Yenimdakula Venkatrao (Prasad Babu) who is a bad person. Srinivas gives Lalitha a job in his dance school and both of them fall in love. Meanwhile, Saroja changes her name to Swapna and comes to rent in Srinivas house not knowing that he's her cousin.

Appendix C - Extractive Summarization

When summarizing text, we are looking to automatically generate a concise version of the original document that retains the most important information.^{[26][27]} In the field of machine learning, summarization is divided into the following paradigms: abstractive summarization and extractive summarization.^{[3][27]} Extractive summarization selects the most important sentences from the original document and copies them to build the summary, while abstractive summarization generates new text, usually including words or phrases that are missing in the original text.^{[3][27]} For our experiments, we use extractive summarization as we do not wish to change the original writing style of the document.

In our work, we use a simple extractive summarization algorithm^[29] which (1) builds a frequency table of words (how many times each word appears in the document/paragraph) (2) gives a score to each sentence based on the words it contains and their frequency in the document (3) Builds a summary by selecting the highest-ranking sentences. In particular, we select at most 7 sentences from each body part with a length of less than 100 words. We decided to use this method for its simplicity of implementation, low-resource consumption (does not require access to GPU) and relative good quality of summaries.

Unlike most academic papers, where the full document (or multi-document) is given to summarize^{[26][27][3]} we opted to do summarization per paragraph. This way, we ensured that (A) the resulting text keeps more than 200 words. (B) the narrative stays fluid. Below, we present a full document summarization against our per-paragraph approach for the title *Animal Farm*, which has 957 words. In the full document approach, the resulting text has 190 words, while our per-paragraph keeps 576 words and introduces more characters and events (Mr Pilkington, Boxer, etc). For the full *Wikiplot* story see Appendix D. For those rare cases where the first extractive summarization produces a resulting text with more than 900 words, we proceed to summarize it again, by creating new paragraphs and only summarizing the body. (e.g: *Don Quixote* originally has 1591 words. It is first reduced to 924 words and later reduced once again to 613 words).

Example of Full document summarization for “*Animal Farm*”. The resulting text has 190 words (original had 957)

When Major dies, two young pigs, Snowball and Napoleon, assume command and turn his dream into a philosophy. They adopt Seven Commandments of Animal-ism, the most important of which is, "All animals are equal". Once Snowball becomes a scapegoat, Napoleon begins purging the farm with his dogs, killing animals he accuses of consorting with his old rival. Mr Frederick, one of the neighbouring farmers, attacks the farm, using blasting powder to blow up the restored windmill. Though the animals win the battle, they do so at great cost, as many, including Boxer, are wounded. The Seven Commandments are reduced to a single phrase: "All animals are equal, but some animals are more equal than others". The animals, overhearing the conversation, notice that the faces of the pigs have begun changing. The pigs Snowball, Napoleon, and Squealer adapt Old Major's ideas into an actual philosophy, which they formally name Animalism. Soon after, Napoleon and Squealer indulge in the vices of humans (drinking alcohol, sleeping in beds, trading). Eventually these are replaced with the maxims, "All animals are equal, but some animals are more equal than others", and "Four legs good, two legs better!"

Only body paragraphs are summarized, one by one. The resulting text has 576 words (original had 957)

Old Major, the old boar on the Manor Farm, calls the animals on the farm for a meeting, where he compares the humans to parasites and teaches the animals a revolutionary song, 'Beasts of England'. When Major dies, two young pigs, Snowball and Napoleon, assume command and turn his dream into a philosophy. The animals revolt and drive the drunken and irresponsible Mr Jones from the farm, renaming it "Animal Farm". They adopt Seven Commandments of Animal-ism, the most important of which is, "All animals are equal". Snowball attempts to teach the animals reading and writing; food is plentiful, and the farm runs smoothly. The pigs elevate themselves to positions of leadership and set aside special food items, ostensibly for their personal health. Napoleon takes the pups from the farm dogs and trains them privately. Napoleon and Snowball struggle for leadership. When Snowball announces his plans to build a windmill, Napoleon has his dogs chase Snowball away and declares himself leader. Napoleon enacts changes to the governance structure of the farm, replacing meetings with a committee of pigs, who will run the farm. Using a young pig named Squealer as a "mouthpiece", Napoleon claims credit for the windmill idea. Once Snowball becomes a scapegoat, Napoleon begins purging the farm with his dogs, killing animals he accuses of consorting with his old rival. Mr Frederick, one of the neighbouring farmers, attacks the farm, using blasting powder to blow up the restored windmill. Though the animals win the battle, they do so at great cost, as many, including Boxer the workhorse, are wounded. Despite his injuries, Boxer continues working harder and harder, until he collapses while working on the windmill. Benjamin, the cynical donkey, who "could read as well as any pig", notices that the van belongs to a knacker, and attempts to mount a rescue; but the animals' attempts are futile. During a poker match, an argument breaks out between Napoleon and Mr Pilkington, and the animals realise that the faces of the pigs look like the faces of humans, and no one can tell the difference between them. The pigs Snowball, Napoleon, and Squealer adapt Old Major's ideas into an actual philosophy, which they formally name Animalism. Soon after, Napoleon and Squealer indulge in the vices of humans (drinking alcohol, sleeping in beds, trading). Later, Napoleon and his pigs secretly revise some commandments to clear them of accusations of law-breaking (such as "No animal shall drink alcohol" having "to excess" appended to it and "No animal shall sleep in a bed" with "with sheets" added to it). The changed commandments are as follows, with the changes bolded: * 4 No animal shall sleep in a bed with sheets. * 5 No animal shall drink alcohol to excess. * 6 No animal shall kill any other animal without cause. Eventually these are replaced with the maxims, "All animals are equal, but some animals are more equal than others", and "Four legs good, two legs better!" as the pigs become more human. This

is an ironic twist to the original purpose of the Seven Commandments, which were supposed to keep order within Animal Farm by uniting the animals together against the humans, and prevent animals from following the humans' evil habits. Through the revision of the commandments, Orwell demonstrates how simply political dogma can be turned into malleable propaganda.

Below we present some limitations from Extractive Summarization and examples in our database:

1) Introduction of new characters / new information

Since we are summarizing paragraphs, it is possible that an event — or character — was deemed unimportant in their introduction but appears in subsequent paragraphs. We can clearly see this in Example 2: A Wizard of Earthsea (see below), where extractive summarization ignores one of the two antagonists (an evil wizard) and introduces the protagonist's best friend at the end (Estarriol).

2) Sensibility to special tokens

Special tokens and structures (such as lists, numerals, references to bold text, etc) affect the extracted summary. For example, when summarizing Animal Farm, it ignores the seven commandments due to their list structure.

3) Incoherence / Plot Modifications

As summarized sentences are mashed together, it can happen that the previous sentence has no relationship with the next one or that plot elements are changed. We can see this in Les Miserables, where we reduce a full summary of 3.522 words to 577. Example :

The town's police inspector, Inspector Javert, who was an adjutant guard at the Bagne of Toulon during Valjean's incarceration, becomes suspicious of the mayor after witnessing his heroics. His friends, Listolier, Fameuil, and Blachevelle were also paired with Fantines friends Dahlia, Zphine, and Favourite.

At a single glance, a reader might think that there is nothing wrong with the sentences, albeit a bit awkward. In truth, we have altered the plot. In the original story, Listolier and company are not friends of Javert, but rather of Flix Tholomys. Since we keep the original introduction and conclusion, we have the guarantee that key aspects of the story are kept true.

Despite these limitations, we are able to greatly cut down on the input length while keeping semantic boundaries.

Example of Extractive Summarization:

Example 2: Extractive Summarization for A Wizard of Earthsea

Plot: (650 words) Ged is a young boy on Gont, one of the larger islands in the north of the archipelago of Earthsea. His mother is dead, his much older siblings have all left home, and his father is a dour, taciturn bronzesmith with nothing in common with his son, so the boy grows up wild and headstrong. Ged discovers by accident that he has an extraordinary talent for magic. His aunt, the village witch, teaches him the little she herself knows, but his power far exceeds hers. One day, he uses his talent and a fog-gathering spell he learned from a passing weatherworker to save his village from Karg raiders. The tale of his remarkable feat spreads far and wide, finally reaching the ear of a wise Gontish mage, Ogion the Silent. He recognizes that the boy is so powerful he must be trained so as not to become a danger to himself and others. In the rite of passage into adulthood, he gives the boy his "true name", Ged, and takes him as an apprentice. In this world, a magician who knows someone's true name has control over that person, so one's true name is revealed only to those whom one trusts completely. Normally, a person is referred to by his or her "use name". Ged's is Sparrowhawk. Though he has grown to love the old man, the youngster is drawn irresistibly to a life of doing, rather than being. At the school, Ged masters his craft with ease, but his pride and arrogance grow even faster than his skill and, in his hubris, he attempts to summon a dead spirit - a perilous spell which goes awry. It is driven off by the head of the school, the Archmage Nemmerle, who expends all of his power in the process and dies shortly thereafter. Normally, Roke's wizards are much sought after by princes and rich merchants, but the new Archmage sends a willing Ged to a poor island group instead, to protect the inhabitants from a powerful dragon and its maturing sons, who have been seen scouting the region. His guess is right and by using the dragon's name, Yevaud, he is able to force the dragon to vow that neither it nor its offspring will ever trouble the islanders. Then, with no idea how to deal with his other foe, Ged tries to return to the safety of Roke, but the magical, protective Mage-wind drives away the ship on which he is a passenger. The vessel sinks, but he manages to reach a small island inhabited by only two old people, a Kargish man and his sister, who were abandoned there as children and who have forgotten there is an outside world and other people. When he is ready to leave, he offers to take the pair wherever they want to go, but the man fearfully turns him down, and the woman does not seem to understand what he means. Ged follows the shadow south. On the island of Iffish, his luck begins to improve. Estarriol insists on accompanying him. The two wizards eventually leave behind the last known island of Earthsea and head out into the open sea. As they draw closer to the shadow, Ged perceives the water gradually congealing and turning into land, an immensely powerful magic. Though Vetch cannot see the transformation, the boat runs aground. Ged steps out of the boat and walks off to confront his waiting shadow. Though some of his teachers had thought it to be nameless, Ged and his adversary speak at the same moment, each naming the other "Ged". The two embrace and become one. The sea returns to its normal state; fortunately, Estarriol is able to fish his healed friend out of the water.

Extractive Summarization Plot: (578 words) The Plague

The text of *The Plague* is divided into five parts. In the town of Oran, thousands of rats, initially unnoticed by the populace, begin to die in the streets. A hysteria develops soon afterward, causing the local newspapers to report the incident. Authorities responding to public pressure order the collection and cremation of the rats, unaware that the collection itself was the catalyst for the spread of the bubonic plague. The main character, Dr. Bernard Rieux, lives comfortably in an apartment building when strangely the building's concierge, M. Michel, a confidante, dies from a fever. Dr. Rieux consults his colleague, Castel, about the illness until they come to the conclusion that a plague is sweeping the town. They both approach fellow doctors and town authorities about their theory, but are eventually dismissed on the basis of one death. However, as more and more deaths quickly ensue, it becomes apparent that there is an epidemic. Authorities, including the Prefect, M. Othon, are slow to accept that the situation is serious and quibble over the appropriate action to take. Official notices enacting control measures are posted, but the language used is optimistic and downplays the seriousness of the situation. A "special ward" is opened at the hospital, but its 80 beds are filled within three days. The separation affects daily activity and depresses the spirit of the townspeople, who begin to feel isolated and introverted, and the plague begins to affect various characters. One character, Raymond Rambert, devises a plan to escape the city to join his lover in Paris after city officials refuse his request to leave. Another character, Father Paneloux, uses the plague as an opportunity to advance his stature in the town by suggesting that the plague was an act of God punishing the citizens' sinful nature. But when Tarrou tells him that others in the city, including Dr. Rieux, also have loved ones outside the city whom they are not allowed to see, Rambert becomes sympathetic and changes his mind. Cottard and Tarrou attend a performance of Gluck's opera *Orpheus and Eurydice*, but the actor portraying Orpheus collapses with plague symptoms during the performance. Towards the end of October, Castel's new anti-plague serum is tried for the first time, but it cannot save the life of Othon's young son, who suffers greatly, as Paneloux, Rieux, and Tarrou look on in horror. Paneloux, who has joined the group of volunteers fighting the plague, gives a second sermon. Othon, however, does not escape death from the disease. Cottard is distressed by the ending of the epidemic, from which he has profited by shady dealings. Two government employees approach him, and he flees. Despite the epidemic's ending, Tarrou contracts the plague and dies after an heroic struggle. Rieux's wife also dies. In February, the town gates open and people are reunited with their loved ones from other cities. Rambert is reunited with his wife. Rieux reveals that he is the narrator of the chronicle and that he tried to present an objective view of the events. Cottard goes mad and shoots at people from his home. He is arrested. Grand begins working on his sentence again. Rieux reflects on the epidemic and reaches the conclusion that there is more to admire than to despise in humans.

Les Misérables (Original : 3522 words. Summarized : 577 words) The story starts in 1815 in Digne. The peasant Jean Valjean has just been released from imprisonment in the Bagnes of Toulon after nineteen years (five for stealing bread for his starving sister and her family, and fourteen more for numerous escape attempts). Upon being released, he is required to carry a yellow passport that marks him as a prisoner, despite having already paid his debt to society by serving his time in prison. Rejected by innkeepers, who do not want to take in a convict, Valjean sleeps on the street. This makes him even angrier and more bitter. However, the benevolent Bishop Myriel, the bishop of Digne, takes him in and gives him shelter. In the middle of the night, Valjean steals Bishop Myriel's silverware and runs away. He is caught and brought back by the police, but Bishop Myriel rescues him by claiming that the silverware was a gift and at that point gives him his two silver candlesticks as well, chastising him to the police for leaving in such a rush that he forgot these most valuable pieces. After the police leave, Bishop Myriel then "reminds" him of the promise, which Valjean has no memory of making, to use the silver candlesticks to make an honest

man of himself. Valjean broods over the Bishop's words. Purely out of habit, he steals a 40-sous coin from chimney-sweep Petit Gervais and chases the boy away. The town's police inspector, Inspector Javert, who was an adjutant guard at the Bagne of Toulon during Valjean's incarceration, becomes suspicious of the mayor after witnessing his heroics. His friends, Listolier, Fameuil, and Blachevelle were also paired with Fantines' friends Dahlia, Zphine, and Favourite. However, Thnardier, hoping to swindle more out of Valjean, runs after them, holding the 1,500 francs, and tells Valjean he wants Cosette back. Eight years later, the Friends of the ABC, led by Enjolras, are preparing an act of anti-Orlanist civil unrest on the eve of the Paris uprising on 26 June 1832, following the death of General Lamarque, the only French leader who had sympathy towards the working class. Valjean, learning that Cosette's lover is fighting, is at first relieved, but an hour later, he puts on a National Guard uniform, arms himself with a gun and ammunition, and leaves his home. Later, Thnardier approaches Marius in a disguise, but Marius is not fooled and recognizes him. Thnardier attempts to blackmail Marius with what he knows of Valjean, but in doing so, he inadvertently corrects Marius' misconceptions about Valjean and reveals all of the good he has done. He tries to convince Marius that Valjean is actually a murderer, and presents the piece of coat he tore off as evidence. Marius pulls out a fistful of five hundred and one thousand franc notes and flings it at Thnardier's face. He then confronts Thnardier with his crimes and offers him an immense amount of money if he departs and promises never to return. Thnardier accepts the offer, and he and Azelma travel to America where he becomes a slave trader. As Marius and Cosette rush to Valjean's house, he informs her that Valjean saved his life at the barricade. They arrive to see him, but the great man is dying. In his final moments, he realizes happiness with his adopted daughter and son-in-law by his side. He also reveals Cosette's past to her as well as her mother's name. Joined with them in love, he dies.

Appendix D - Full Wikipedia Plots

All plots have been extracted from Wikipedia. For the original dataset and code to build it, please see Mark Riedl's GitHub. ^[28]

Title: Animal Farm

Original Plot (957 words):Old Major, the old boar on the Manor Farm, calls the animals on the farm for a meeting, where he compares the humans to parasites and teaches the animals a revolutionary song, 'Beasts of England'. When Major dies, two young pigs, Snowball and Napoleon, assume command and turn his dream into a philosophy. The animals revolt and drive the drunken and irresponsible Mr Jones from the farm, renaming it "Animal Farm". They adopt Seven Commandments of Animal-ism, the most important of which is, "All animals are equal". Snowball attempts to teach the animals reading and writing; food is plentiful, and the farm runs smoothly. The pigs elevate themselves to positions of leadership and set aside special food items, ostensibly for their personal health. Napoleon takes the pups from the farm dogs and trains them privately. Napoleon and Snowball struggle for leadership. When Snowball announces his plans to build a windmill, Napoleon has his dogs chase Snowball away and declares himself leader. Napoleon enacts changes to the governance structure of the farm, replacing meetings with a committee of pigs, who will run the farm. Using a young pig named Squealer as a "mouthpiece", Napoleon claims credit for the windmill idea. The animals work harder with the promise of easier lives with the windmill. After a violent storm, the animals find the windmill annihilated. Napoleon and Squealer convince the animals that Snowball destroyed it, although the scorn of the neighbouring farmers suggests that its walls were too thin. Once Snowball becomes a scapegoat, Napoleon begins purging the farm with his dogs, killing animals he accuses of consorting with his old rival. He and the pigs abuse their power, imposing more control while reserving privileges for themselves and rewriting history, villainising Snowball and glorifying Napoleon. Squealer justifies every statement Napoleon makes, even the pigs' alteration of the Seven Commandments of Animalism to benefit themselves. 'Beasts of England' is replaced by an anthem glorifying Napoleon, who appears to be adopting the lifestyle of a man. The animals remain convinced that they are better off than they were when under Mr Jones. Squealer abuses the animals' poor memories and invents numbers to show their improvement. Mr Frederick, one of the neighbouring farmers, attacks the farm, using blasting powder to blow up the restored windmill. Though the animals win the battle, they do so at great cost, as many, including Boxer the workhorse, are wounded. Despite his injuries, Boxer continues working harder and harder, until he collapses while working on the windmill. Napoleon sends for a van to take Boxer to the veterinary surgeon's, explaining that better care can be given there. Benjamin, the cynical donkey, who "could read as well as any pig", notices that the van belongs to a knacker, and attempts to mount a rescue; but the animals' attempts are futile. Squealer reports that the van was purchased by the hospital and the writing from the previous owner had not been repainted. He recounts a tale of Boxer's death in the hands of the best medical care. Years pass, and the pigs learn to walk upright, carry whips and wear clothes. The Seven Commandments are reduced to a single phrase: "All animals are equal, but some animals are more equal than others". Napoleon holds a dinner party for the pigs and the humans of the area, who congratulate Napoleon on having the hardest-working but least fed animals in the country. Napoleon announces an alliance with the humans, against the labouring classes of both "worlds". He abolishes practices and traditions related to the Revolution, and changes the name of the farm to "The Manor Farm". The animals,

overhearing the conversation, notice that the faces of the pigs have begun changing. During a poker match, an argument breaks out between Napoleon and Mr Pilkington, and the animals realise that the faces of the pigs look like the faces of humans, and no one can tell the difference between them. The pigs Snowball, Napoleon, and Squealer adapt Old Major's ideas into an actual philosophy, which they formally name Animalism. Soon after, Napoleon and Squealer indulge in the vices of humans (drinking alcohol, sleeping in beds, trading). Squealer is employed to alter the Seven Commandments to account for this humanisation, an allusion to the Soviet government's revising of history in order to exercise control of the people's beliefs about themselves and their society. The original commandments are: # Whatever goes upon two legs is an enemy. # Whatever goes upon four legs, or has wings, is a friend. # No animal shall wear clothes. # No animal shall sleep in a bed. # No animal shall drink alcohol. # No animal shall kill any other animal. # All animals are equal. Later, Napoleon and his pigs secretly revise some commandments to clear them of accusations of law-breaking (such as "No animal shall drink alcohol" having "to excess" appended to it and "No animal shall sleep in a bed" with "with sheets" added to it). The changed commandments are as follows, with the changes bolded: * 4 No animal shall sleep in a bed with sheets. * 5 No animal shall drink alcohol to excess. * 6 No animal shall kill any other animal without cause. Eventually these are replaced with the maxims, "All animals are equal, but some animals are more equal than others", and "Four legs good, two legs better!" as the pigs become more human. This is an ironic twist to the original purpose of the Seven Commandments, which were supposed to keep order within Animal Farm by uniting the animals together against the humans, and prevent animals from following the humans' evil habits. Through the revision of the commandments, Orwell demonstrates how simply political dogma can be turned into malleable propaganda.

Example 2: A Wizard of Earthsea

Original Plot (1066 words): Ged is a young boy on Gont, one of the larger islands in the north of the archipelago of Earthsea. His mother is dead, his much older siblings have all left home, and his father is a dour, taciturn bronzesmith with nothing in common with his son, so the boy grows up wild and headstrong. Ged discovers by accident that he has an extraordinary talent for magic. His aunt, the village witch, teaches him the little she herself knows, but his power far exceeds hers. One day, he uses his talent and a fog-gathering spell he learned from a passing weatherworker to save his village from Karg raiders. The tale of his remarkable feat spreads far and wide, finally reaching the ear of a wise Gontish mage, Ogion the Silent. He recognizes that the boy is so powerful he must be trained so as not to become a danger to himself and others. In the rite of passage into adulthood, he gives the boy his "true name", Ged, and takes him as an apprentice. In this world, a magician who knows someone's true name has control over that person, so one's true name is revealed only to those whom one trusts completely. Normally, a person is referred to by his or her "use name". Ged's is Sparrowhawk. The undisciplined young man grows restless under the gentle, patient tutelage of his master. One day, at the taunting of the daughter of the local lord - who, it is later revealed, is also a witch - Ged seeks a powerful spell from one of Ogion's old books to impress the girl. As he reads the spell, to his horror, a shadowy being manifests. The shade advances on Ged, but is driven away by the timely return of Ogion. Ogion finally gives him a choice: stay with him or go to the renowned school for wizards, on the island of Roke. Though he has grown to love the old man, the youngster is drawn irresistibly to a life of doing, rather than being. At the school, Ged masters his craft with ease, but his pride and arrogance grow even faster than his skill and, in his hubris, he attempts to summon a dead spirit - a perilous spell which goes awry. The shadow seizes the chance to escape into the world and attacks him, scarring his face. It is driven off by the head of the school, the Archmage Nemmerle, who expends all of

his power in the process and dies shortly thereafter. Ged is wracked with guilt at causing the old man's death, but after a painful and slow recovery, he graduates from the school. Normally, Roke's wizards are much sought after by princes and rich merchants, but the new Archmage sends a willing Ged to a poor island group instead, to protect the inhabitants from a powerful dragon and its maturing sons, who have been seen scouting the region. Ged eventually realizes that he cannot both defend the islanders against the dragon and himself against the nameless thing he summoned into the world. He takes a desperate gamble; in the old histories, he has found the true name of a dragon which might be the one he faces. His guess is right and by using the dragon's name, Yevaud, he is able to force the dragon to vow that neither it nor its offspring will ever trouble the islanders. Then, with no idea how to deal with his other foe, Ged tries to return to the safety of Roke, but the magical, protective Mage-wind drives away the ship on which he is a passenger. On the far northern island of Osskil, his nemesis takes possession of a man and nearly catches him. Ged flees to what appears to be a safe haven in the castle of Benderesk, the lord of Terranon. Serret, his wife, is the same girl who taunted Ged years ago. She tries to enslave Ged using the power of a stone which harbors one of the "Nameless Ones", ancient malevolent powers that predate people. Fortunately Ged realizes his peril just in time and flies away in the form of a falcon. He instinctively returns to Gont and Ogion, who advises him to turn the tables on his shadow. In following his master's wise guidance, the roles of Ged and his enemy become reversed, and the shadow becomes the hunted. Ged pursues the shadow southwards across the ocean, but is nearly drowned when the shadow lures him into steering his boat onto rocks. The vessel sinks, but he manages to reach a small island inhabited by only two old people, a Kargish man and his sister, who were abandoned there as children and who have forgotten there is an outside world and other people. Despite their initial fear of him, they provide him with food and water. After Ged regains his strength, he constructs another boat. When he is ready to leave, he offers to take the pair wherever they want to go, but the man fearfully turns him down, and the woman does not seem to understand what he means. However, she gives him a parting gift of one of her few possessions, a broken half of an armlet. (The siblings' story and the gift's significance are revealed in the sequel, *The Tombs of Atuan*.) Back at sea, the shadow nearly takes Ged unawares, but he senses it just in time and comes to grips with it, forging a bond that cannot be broken. Ged follows the shadow south. On the island of Iffish, his luck begins to improve. The resident wizard is Vetch (true name Estarriol), the only friend he made at school. Estarriol insists on accompanying him. The two wizards eventually leave behind the last known island of Earthsea and head out into the open sea. As they draw closer to the shadow, Ged perceives the water gradually congealing and turning into land, an immensely powerful magic. Though Vetch cannot see the transformation, the boat runs aground. Ged steps out of the boat and walks off to confront his waiting shadow. Though some of his teachers had thought it to be nameless, Ged and his adversary speak at the same moment, each naming the other "Ged". The two embrace and become one. The sea returns to its normal state; fortunately, Estarriol is able to fish his healed friend out of the water.

Example 3: The Plague

Original Plot (1119 words): The text of *The Plague* is divided into five parts. In the town of Oran, thousands of rats, initially unnoticed by the populace, begin to die in the streets. A hysteria develops soon afterward, causing the local newspapers to report the incident. Authorities responding to public pressure order the collection and cremation of the rats, unaware that the collection itself was the catalyst for the spread of the bubonic plague. The main character, Dr. Bernard Rieux, lives comfortably in an apartment building when strangely the building's concierge, M. Michel, a confidante, dies from a fever. Dr. Rieux consults his colleague, Castel, about the illness until they come to the conclusion that a plague is sweeping the town.

They both approach fellow doctors and town authorities about their theory, but are eventually dismissed on the basis of one death. However, as more and more deaths quickly ensue, it becomes apparent that there is an epidemic. Authorities, including the Prefect, M. Othon, are slow to accept that the situation is serious and quibble over the appropriate action to take. Official notices enacting control measures are posted, but the language used is optimistic and downplays the seriousness of the situation. A "special ward" is opened at the hospital, but its 80 beds are filled within three days. As the death toll begins to rise, more desperate measures are taken. Homes are quarantined; corpses and burials are strictly supervised. A supply of plague serum finally arrives, but there is only enough to treat existing cases and the country's emergency reserves are depleted. When the daily number of deaths jumps to 30, the town is sealed and an outbreak of plague is officially declared. The town is sealed off. The town gates are shut, rail travel is prohibited, and all mail service is suspended. The use of telephone lines is restricted only to "urgent" calls, leaving short telegrams as the only means of communicating with friends or family outside the town. The separation affects daily activity and depresses the spirit of the townspeople, who begin to feel isolated and introverted, and the plague begins to affect various characters. One character, Raymond Rambert, devises a plan to escape the city to join his lover in Paris after city officials refuse his request to leave. He befriends some criminals so that they may smuggle him out of the city. Another character, Father Paneloux, uses the plague as an opportunity to advance his stature in the town by suggesting that the plague was an act of God punishing the citizens' sinful nature. His diatribe falls on the ears of many citizens of the town, who turned to religion in droves but would not have done so under normal circumstances. Cottard, a criminal remorseful enough to attempt suicide yet fearful of being arrested, becomes wealthy as a major smuggler. Meanwhile, Dr. Rieux, a vacationer Jean Tarrou, and a civil servant Joseph Grand exhaustively treat patients in their homes and in the hospital. Rambert informs Tarrou of his escape plan, but when Tarrou tells him that others in the city, including Dr. Rieux, also have loved ones outside the city whom they are not allowed to see, Rambert becomes sympathetic and changes his mind. He then decides to join Tarrou and Dr. Rieux to help fight the epidemic. In mid-August, the situation continues to worsen. People try to escape the town, but some are shot by armed sentries. Violence and looting break out on a small scale, and the authorities respond by declaring martial law and imposing a curfew. Funerals are conducted with more and more speed, no ceremony, and little concern for the feelings of the families of the deceased. The inhabitants passively endure their increasing feelings of exile and separation; despondent, they waste away emotionally as well as physically. In September and October, the town remains at the mercy of the plague. Rieux hears from the sanatorium that his wife's condition is worsening. He also hardens his heart regarding the plague victims so that he can continue to do his work. Cottard, on the other hand, seems to flourish during the plague, because it gives him a sense of being connected to others, since everybody faces the same danger. Cottard and Tarrou attend a performance of Gluck's opera *Orpheus and Eurydice*, but the actor portraying Orpheus collapses with plague symptoms during the performance. Rambert finally has a chance to escape, but he decides to stay, saying that he would feel ashamed of himself if he left. Towards the end of October, Castel's new anti-plague serum is tried for the first time, but it cannot save the life of Othon's young son, who suffers greatly, as Paneloux, Rieux, and Tarrou look on in horror. Paneloux, who has joined the group of volunteers fighting the plague, gives a second sermon. He addresses the problem of an innocent child's suffering and says it is a test of a Christian's faith, since it requires him either to deny everything or believe everything. He urges the congregation not to give up the struggle but to do everything possible to fight the plague. A few days after the sermon, Paneloux is taken ill. His symptoms do not conform to those of the plague, but the disease still proves fatal. Tarrou and Rambert visit one of the isolation camps, where they meet Othon. When Othon's period of quarantine ends, he elects to stay in the camp as a volunteer because this will make him feel less separated from his dead son. Tarrou tells Rieux the story of his life, and the two men go

swimming together in the sea. Grand catches the plague and instructs Rieux to burn all his papers. But Grand makes an unexpected recovery, and deaths from the plague start to decline. By late January, the plague is in full retreat, and the townspeople begin to celebrate the imminent opening of the town gates. Othon, however, does not escape death from the disease. Cottard is distressed by the ending of the epidemic, from which he has profited by shady dealings. Two government employees approach him, and he flees. Despite the epidemic's ending, Tarrou contracts the plague and dies after an heroic struggle. Rieux's wife also dies. In February, the town gates open and people are reunited with their loved ones from other cities. Rambert is reunited with his wife. Rieux reveals that he is the narrator of the chronicle and that he tried to present an objective view of the events. Cottard goes mad and shoots at people from his home. He is arrested. Grand begins working on his sentence again. Rieux reflects on the epidemic and reaches the conclusion that there is more to admire than to despise in humans.

Appendix E - Samples of Decoding Strategies

The following examples were generated using standard GPT-2 (no fine tuning) by giving it the prompt: “The nicest thing about Liechtenstein is’

- Greedy: *“The nicest thing about Liechtenstein is that it's a very small country. It's not a big country. It's not a big country. It's not a big country. It's not a big country. It's not a”*
- Beam: *“The nicest thing about Liechtenstein is that it's the only country in the world that doesn't have a national currency. It's the only country in the world that doesn't have a national currency. It's the only country in the”*
- Sampling: *“The nicest thing about Liechtenstein is that we don't show it in the entirety -- but we combine our efforts to win this guy at this point. We hold him accountable on many fronts.” After a tumultuous first win, Lie”*
- Temperature (0,7): *“The nicest thing about Liechtenstein is that it's at the center of many of the most pressing issues in our daily lives, including the treatment of the pensioners. If we don't take a hard look at the past, we're”*
- Nucleus Sampling (0,92): *“The nicest thing about Liechtenstein is that if you give it too much fanfare you can't even see it very clearly. And this is really one of those things where you don't want to actually see the real deal until you get”*

The next set of samples were extracted from one of the first GPT-2 fine-tunings that we did for this thesis. In it, we extracted 100 articles related to the country Liechtenstein from Google and fine-tuned GPT-2 to write a short sentence about it. Giving the same prompt as before: “The nicest thing about Liechtenstein is”

- Greedy: *“The nicest thing about Liechtenstein is that it's small. The government is very friendly, and the residents are very proud of their “vehemently”Russophobic” (i.e. non-Russophobic) political views. This is despite the government having strong ties to”*
- Temperature (0.7): *“The nicest thing about Liechtenstein is how peaceful it is. The capital, Vaduz, is almost dead except for the suitcase-snorting cars driving past. The only other European city completely without a postal parcel is St. Gallen in the olden age from 13. Despite”*

- Nucleus Sampling (0.92): *“The nicest thing about Liechtenstein is the fact that it has a very laidback, pretty government. The people are very proud of their new government, which is very much based on the 1921 constitution and follows a very modern constitution system. There is a skate park and a party is happening”*

As we can see, after fine-tuning the degeneration of the produced text has decreased, as GPT-2 now has more information about the topic. This shows that, no matter the decoding strategy, if the LM is missing information about a topic it will produce a bland, incoherent output. These samples also illustrate how fine-tuning improves the quality of the generated text.

