Generating Stories by Prompting Pre-trained Language Models

Paresh Pradhan

Interim Report for

Master of Science in Machine Learning & Artificial Intelligence

Liverpool John Moores University & upGrad

December 2022

# DEDICATION

# ACKNOWLEDGEMENTS

# ABSTRACT

# TABLE OF CONTENTS

[DEDICATION 1](#_Toc685544928)

[ACKNOWLEDGEMENTS 2](#_Toc1348095451)

[ABSTRACT 3](#_Toc2132963568)

[TABLE OF CONTENTS 4](#_Toc2044964634)

[LIST OF TABLES 7](#_Toc1574810178)

[LIST OF FIGURES 8](#_Toc279142929)

[LIST OF ABBREVIATIONS 9](#_Toc774127971)

[CHAPTER 1: INTRODUCTION 10](#_Toc1365356800)

[1.1 Background of the Study 11](#_Toc984350137)

[1.2 Research Questions 12](#_Toc1358486203)

[1.3 Aim & Objectives 12](#_Toc389038508)

[1.4 Significance of the Study 13](#_Toc751710371)

[1.5 Scope of the Study 13](#_Toc1062898036)

[1.6 Structure of the Study 14](#_Toc2068340653)

[CHAPTER 2: LITERATURE REVIEW 14](#_Toc1394020312)

[2.1 Automatic Story Generation 15](#_Toc2145768095)

[2.1.1 Introduction 15](#_Toc807839344)

[2.1.2 Structural Models 15](#_Toc948702747)

[2.1.2.1 Graph-Based Approaches 16](#_Toc1111422160)

[2.1.2.2 Grammar-Based Approaches 16](#_Toc1384145753)

[2.1.3 Planning Based Models 17](#_Toc965511201)

[2.1.4 ML Models 17](#_Toc1814778435)

[2.1.4.1 Story Abstraction 17](#_Toc1075589289)

[2.1.4.2 Script Learning and Generation 18](#_Toc389448381)

[2.1.4.3 Story Completion 20](#_Toc1330043707)

[2.1.4.4 Story Generation 22](#_Toc1978132616)

[2.1.7 Story Evaluation 25](#_Toc1248066672)

[2.1.8 Challenges in Automatic Story Generation 28](#_Toc693211711)

[2.2 Prompt Based Learning 30](#_Toc276314918)

[2.2.1 Brief History of NLP 31](#_Toc1083325890)

[2.2.2 Introduction to Prompting 32](#_Toc1954299913)

[2.2.2.1 Prompt Addition 32](#_Toc1038939926)

[2.2.2.2 Answer Search 32](#_Toc58683428)

[2.2.2.3 Answer Mapping 33](#_Toc1483462202)

[2.2.2.4 Design Considerations for Prompting 33](#_Toc765130583)

[2.2.3 Pre-trained Language Models 34](#_Toc607553140)

[2.2.3.1 Training Objectives 34](#_Toc1856758867)

[2.2.3.2 Noising Functions 34](#_Toc1076777778)

[2.2.3.3 Directionality of Representations 35](#_Toc369516428)

[2.2.3.4 Typical Pre-training Methods 35](#_Toc1661814254)

[2.2.4 Prompt Engineering 37](#_Toc23126687)

[2.2.4.1 Prompt Shape 37](#_Toc1363976326)

[2.2.4.2 Manual Template Engineering 37](#_Toc901991562)

[2.2.4.3 Automated Template Engineering 38](#_Toc408445568)

[2.2.5 Answer Engineering 40](#_Toc1790107174)

[2.2.5.1 Answer Shape 40](#_Toc872946957)

[2.2.5.2 Answer Shape Design Methods 41](#_Toc440128871)

[2.2.7 Training Strategies for Prompting Methods 42](#_Toc940814387)

[2.2.7.1 Training Settings 42](#_Toc318026337)

[2.2.7.2 Parameter Update Methods 43](#_Toc657530009)

[2.2.9 Prompt-relevant Topics 45](#_Toc2112832159)

[2.2.9.1 Few-shot Learning 45](#_Toc1276759940)

[2.2.9.2 QA-based Task Formulation 46](#_Toc332432820)

[2.2.9.3 Controlled Generation 46](#_Toc515059268)

[2.2.9.4 Data Augmentation 47](#_Toc809596006)

[2.2.10 Challenges in Prompting 47](#_Toc1603561539)

[2.2.10.1 Prompt Design 47](#_Toc4737187)

[2.2.10.2 Answer Engineering 48](#_Toc2058250863)

[2.2.10.3 Selection of Tuning Strategy 48](#_Toc836769974)

[2.2.10.4 Multiple Prompt Learning 49](#_Toc1007119775)

[2.2.10.5 Selection of Pre-trained Models 49](#_Toc636523240)

[2.2.10.6 Theoretical and Empirical Analysis of Prompting 49](#_Toc475897150)

[2.2.10.7 Transferability of Prompts 50](#_Toc1867068165)

[2.2.10.8 Combination of Different Paradigms 50](#_Toc870500709)

[2.2.10.9 Calibration of Prompting Methods 50](#_Toc86177093)

[2.3 Summary 51](#_Toc1055275576)

[CHAPTER 3: RESEARCH METHODOLOGY 51](#_Toc1752369079)

[3.1 Introduction 52](#_Toc12295305)

[3.2 Theoretical Background 52](#_Toc1630303715)

[3.3 Methodology 52](#_Toc707701410)

[3.3.1 End-to-End Pipeline 52](#_Toc1444532036)

[3.3.3 Data Preparation 53](#_Toc9736789)

[3.3.4 Implementation 54](#_Toc1685047017)

[3.3.5 Evaluation 55](#_Toc914356339)

[3.3 Tools 56](#_Toc964984672)

[3.5 Summary 56](#_Toc1524769939)

[REFERENCES 56](#_Toc4200066)

[APPENDIX A: RESEARCH PLAN 63](#_Toc360995990)

[APPENDIX B: RESEARCH PROPOSAL 64](#_Toc1141024679)

[APPENDIX C: ETHICS FORMS 65](#_Toc1467331277)

# LIST OF TABLES

# LIST OF FIGURES

# LIST OF ABBREVIATIONS

# CHAPTER 1: INTRODUCTION

## 1.1 Background of the Study

In recent years, with the emergence of large pre-trained language models, the quality of machine-generated text has improved significantly (Rashkin et al., 2018; Radford et al., 2019; Zhang et al., 2019; Brown et al., 2020; Guan et al., 2020; Bakhtin et al., 2021). Today, models can generate text that is indistinguishable from human-written text (Clark et al., 2021).

Controlling the generation is still a challenge however, despite the fact that large-scale PLMs have demonstrated excellent capabilities in producing coherent and intelligible text (Keskar et al., 2019; Radford et al., 2019; Zellers et al., 2019). A more thorough examination of generated text reveals problems like topic drift and self-contradiction (Fan et al., 2019; Bisk et al., 2020; Gao et al., 2020a; Tan et al., 2020; Dou et al., 2021; Dziri et al., 2021). These flaws stand out, in particular, for open-ended text generation tasks that require a high level of coherence, such as story generation.

Stories generated using language models have shown to lack discourse coherence (Bosselut et al., 2018; Ji and Huang, 2021), global planning (Hua and Wang, 2020; Tan et al., 2020) and common-sense knowledge (Ji et al., 2020; Xu et al., 2020). While the individual sentences in a generated text seem logical and fluent, when put together, the overall story often does not make much sense (See et al., 2019; Goldfarb-Tarrant et al., 2020). In long-form text generation, sentences tend to repeat which leads to reduction in story quality (Yao et al., 2019).

To provide structure to the generation process, recent works have tried to used explicit content planning. The content plan comes in different forms. (Fan et al., 2018a) used prompts. (Xu et al., 2018; Yao et al., 2019) used keywords and key-phrases. (Fan et al., 2019) used semantic frames. (Sun et al., 2020) used summaries. To make use of these content plans, PLMs generally require fine-tuning on content-plan related data. A challenge with fine-tuning PLMs is that, in addition to needing training data, the model has a tendency to learn frequently occurring events from the content plan and derives common sense information from them (Fan et al., 2019). This leads to lack of variety in generated stories.

Another novel approach to address linguistic issues without fine-tuning is provided by the recently proposed prompt-based learning (Liu et al., 2021). In this framework, task-specific prompts can be used to address text-based problems. Researchers have shown that using prompts, PLMs can solve existing or new generation tasks without need for fine-tuning (Brown et al., 2020; Li and Liang, 2021).

Although prompt-based learning looks promising, there are still some challenges. Prompts are highly task-specific and are hard to transfer or reuse for new tasks (Gao et al., 2020b). Even for the same task the prompts may not work well for all instances in a large population (le Scao and Rush, 2021).

## 1.2 Research Questions

This thesis tries to answer the following questions:

1. The approaches for story generation with fine-grained control require fine-tuning of PLMs. Can these approaches be used with Prompt-based learning to generate stories in a Few-Shot manner without fine-tuning?
2. The previous methods largely use GPT2 as base model. Can using the latest generation GPT3 (or alternatives) improve the text generation capabilities?
3. Prompt-based learning has been used to generate text in few-shot manner. Can this be extended to story generation task?

## 1.3 Aim & Objectives

This work tries to explore the Few-shot capabilities of GPT3 for long-form controllable story generation task.

Objectives:

* To conduct a comprehensive review of available literature with regards to Long-form story generation, Prompt-learning and Few-Shot text generation.
* To explore the viability and then develop a method to generate short and long form stories using few-shot generation and prompting.
* To evaluate the generated stories using automated story generation evaluation metrics
* To compare the developed method against existing/state-of-the-art methods.

## 1.4 Significance of the Study

Story Generation is a field under active research. While short-form story generation has been studied extensively, long-form story generation is relatively under-explored. Although fine-tuning based approaches have been used in previous works, there is a lack of research in generating stories without fine-tuning.

This work tries to fill these gaps by adding to the existing literature, providing benchmarks and contributing code. This work also explores recent developments in Prompt-based learning and Few-Shot generation.

In terms of application, this work helps story writers write better stories in conjunction with AI. This can help writers get new ideas or get over the writer’s block.

## 1.5 Scope of the Study

The scope of this thesis work is defined as follows:

* The thesis work is to be completed within 17 weeks after submission of research proposal.
* The experimentation will be conducted using open-source software and models.
* The experimentation will be conducted using publicly available GPU such as Google-Colab.
* Human evaluation of the generated story is not a part of this thesis work. The evaluation will only focus on automated metrics.

## 1.6 Structure of the Study

The structure of the study is as follows:

* Chapter 1 – Introduction: This chapter provides introduction and background for this research work.
* Chapter 2 – Literature Review: This chapter mentions the related works in the fields of Automatic story generation and prompt-learning.
* Chapter 3 – Methodology: This chapter gives a detailed walkthrough of the methodology followed during the experimentation stage.
* Chapter 4 – Implementation: This chapter details the different experiments performed for the story generation task.
* Chapter 5 – Results: This chapter discusses the results of the experiments performed in Chapter 4.
* Chapter 6 – Conclusion: This chapter concludes the work done in the thesis and discusses future improvements.

# CHAPTER 2: LITERATURE REVIEW

## 2.1 Automatic Story Generation

### 2.1.1 Introduction

Automated story generation involves creating a series of events or actions that fit certain criteria and can be told as a story using a computer program [90]. These stories often have a setting, characters, and objects, and may also have a specific message or goal that the author wants to convey to the reader through the events of the story. While systems for generating stories have been developed since the 1960s [150], they have not achieved the level of creativity and complexity seen in human-generated stories and are therefore considered to be weak artificial intelligence systems [75].

To be considered truly creative, a computer system must be able to generate stories that are different from those it has seen before. This requires taking into account a wide range of attributes, such as the setting of the story, the desires and motivations of the characters, and the interactions and conflicts between characters. The large number of potential attributes involved in a story can make it difficult for a computer system to efficiently search for and generate a unique story. Additionally, the goal of the story, its believability, and its interestingness further complicate the process of story generation. Open story generation, in which stories are generated without relying on pre-engineered domain models, presents two additional challenges: the automatic creation of a domain model and the evaluation of the story's progression to guide the generation process.

There has been a significant amount of research on computational narratives, including efforts to survey and classify different story generation systems. In 2009, Gervás [60] examined how these systems attempted to replicate human creativity and the extent to which they achieved key aspects of computational creativity. Since this study was published, there has been a surge in the development of automatic story generators. Young et al. [187] conducted a survey focusing on planning and reasoning in computational narratives, and more recently, Kybartas and Bidarra [84] classified story generation systems based on the degree of automation in plot and space generation into four categories: manual authoring, plot generation, space generation, and story generation that automates elements of both plot and space.

### 2.1.2 Structural Models

In the field of story generation, schemas are used to automatically generate structured stories by dividing the stories into slots that follow a predetermined structure. These slots are then filled by inserting similar slots from previously collected and annotated stories, taking into account the interactions between the contents of the generated story slots.

#### 2.1.2.1 Graph-Based Approaches

There are several approaches to generating stories using computer programs, including building story graphs and using schemas. In the story graph approach, a branching graph representing all possible stories in a given space is constructed, and a linear path through the graph is selected to create the story. The quality of the generated story depends on the quality of the constructed graph, and adding constraints to the search process can improve the results. Maranda [104] developed a graph-based system for generating folktales based on Propp's story structure, which searches a knowledge base for pieces that match the functions represented by the nodes in the graph and concatenates them to create the story. The SCHEHERAZADE system proposed by Li et al. [90] collects human experiences about a particular domain in the form of scripts and learns a plot graph based on these scripts, which is then traversed to generate stories. SCHEHERAZADE's graphs are similar to Maranda's, but they are acyclic and include mutually exclusive constraints on some of the events in the generated stories.

#### 2.1.2.2 Grammar-Based Approaches

Using grammar to generate stories began with Lakoff [85], who reformulated Propp's story structure into a story grammar and used expandable rewrite rules to generate stories using Propp's functions as the alphabet of the narrative's formal language. This inspired the development of other story grammars, such as Pemberton's [127] grammar for an old French epic, which was implemented as the GESTER program [128] that generates well-structured stories with a clear beginning, middle, and end. BRUTUS [31] is another system that generates betrayal stories based on story grammars, using frame structures to group elements of the story into themes. However, these specialized grammars are limited to a specific domain and can only generate a small range of stories, leading to the need for more general story grammars. Rumelhart [149] proposed the first general story grammar, and several others have been proposed since, including the widely adopted grammar of Thorndyke [170].

While story structural models are easy to implement and can generate well-structured stories, they are limited in their ability to generate stories with multiple protagonists, and their generated stories may not be coherent or believable due to a lack of focus on the semantics of the story. They may also suffer from the over-generation problem, producing non-story texts that are accepted as stories.

### 2.1.3 Planning Based Models

Story grammar theories have been criticized for their limited ability to understand stories [23, 24], as they focus on the syntax of a story rather than its semantics and cannot be applied to stories with conflicting goals or multiple protagonists. In response, the story points theory was developed, which views a story as a chain of causally connected events working towards a specific goal [181]. Cook's [46] Plotto: The Master Book of All Plots, which contains a structured collection of plot fragments with instructions on how to combine them to create complete plots, is based on the story points theory and has been used as a source for computational narratives. However, the Plotter system proposed by Eger et al. [53], which generates plots from Plotto's fragments, has limitations in terms of story consistency due to its reliance on the current state of the story rather than its past states. Using AI planning algorithms in automatic story generation involves providing an initial state and a goal for a reasoner to infer actions that will lead to the story goal, potentially with the addition of a directing process to improve the quality of the generated story.

### 2.1.4 ML Models

In machine learning (ML), a story can be viewed as a sequence of events, and the probability distribution between these events can be learned from a collection of stories, also known as a corpus.

#### 2.1.4.1 Story Abstraction

In order to improve the efficiency of learning and inference in machine learning (ML), it is useful to create a simplified representation of stories, known as a story abstraction, which focuses on the main events and entities in the story and increases the potential overlap between stories [105]. There are several approaches to creating a story abstraction, each with its own trade-offs. Chambers and Jurafsky [36, 37] and Jans et al. [76] represented stories as a chain of events, using (verb, dependency) pairs to associate each event with the grammatical role played by the protagonist. While this representation is able to capture the relationships between verbs and their arguments, it can lead to inconsistent subject-verb-object tuples and is limited to representing the actions of a single protagonist.

In contrast, Balasubramanian et al. [18, 19] used an Open IE system to extract Rel-grams schemas, which are relational triples in the form (arg1, Relation, arg2). This representation is able to express interactions between entities, but it may be more sparse than other approaches. Pichotta and Mooney [131] proposed a 4-tuple event representation in the form verb (subject, object, prepositional), which can express interactions between entities and was later improved by the inclusion of prepositions [133]. Martin et al. [105] and Tambwekar et al. [167] also used a 4-tuple event representation in the form (subject, verb, object, modifier), while Ammanabrolu et al. [6] used a 5-tuple representation in the form (subject, verb, preposition, object, M). Yao et al. [186]

#### 2.1.4.2 Script Learning and Generation

The process of script learning and generation involves using statistical models to analyze the relationships between events in a story and predict new events that could be part of a given chain of events. This process typically involves using a dependency parser to extract verb and argument information, a coreference resolver to identify expressions that refer to the same entity, and techniques such as Pointwise Mutual Information or skip n-grams to build a statistical model of event chains. This model can then be used to generate ranked lists of possible events that could fit into a given chain of events.

Chambers and Jurafsky [36] used coreference relationships and Pointwise Mutual Information (PMI) to extract the chain of events and pairwise relationships between events for a single protagonist. They then used a support vector machine to classify the temporal relationship between two events, as described in Chambers et al. [38], and generated a ranked list of possible events that could be part of the chain. This work improved upon previous methods by using coreference relationships to extract the chain of events for a single protagonist and using PMI to extract pairwise relationships between events.

Jans et al. [76] proposed using skip n-grams to learn about event chain statistics by pairing each event with the three following events in the chain, which improved upon the PMI method [36] by decreasing data sparsity and improving the training process. The bigram probabilities ranking function used in this approach scored events based on their position in the chain by considering the preceding and following events, which improved upon previous methods by modeling an event chain in the order it was observed.

Balasubramanian et al. [18, 19] introduced the Rel-grams system, a Markov model similar to that of Jans et al. [76] but focusing on relationship co-occurrence rather than argument co-occurrence. This improvement allowed the system to predict one of the arguments if provided with the relationships and the other argument.

Pichotta and Mooney [131] proposed using structured events with multi-arguments to encode pairwise entity relationships between story events and model the interaction between various entities. This improvement enabled the system to generate an event chain for the whole story rather than separate entity-based event chains produced by verb-dependency pairs. However, the complexity of the event structure added to the complexity of the statistical model. This system demonstrated improved prediction accuracy compared to systems with verb-dependency pairs, but the complexity of the event structure added to the complexity of the statistical model.

Several studies have attempted to predict events using language models that involve compositional representations of events, in contrast to previous count-based techniques. Rudinger et al. [148] trained a Log-Bilinear model to predict story events, arguing that event prediction could be productively reframed as a language modeling task. Their discriminative language model showed improved performance compared to prior count-based methods. Pichotta and Mooney [133] used Long Short-Term Memory (LSTM) RNN to learn about stories statistically. Their model was able to predict nouns or coreference information concerning event arguments, and demonstrated better performance compared to several baseline systems. They also extended their work to predict events directly from raw text without using explicit event structures [132], and found that the difference between raw text models and structured events models was marginal, indicating that extracting event structures may not be necessary for event prediction, particularly in an encoder-decoder setup. Granroth-Wilding and Clark [63] compared several approaches for deriving vector representations of event predicates and argument nouns, and used these representations in a compositional neural network model that predicts how probable it is that two events will appear in one event chain by performing a non-linear composition of their predicates and arguments.

Mostafazadeh et al. [114] introduced the Story Cloze Test (SCT) and built the ROCStories Corpora to test the ability of machines to select a correct story ending given the story context. Some researchers used ROCStories to build classification models that can choose the correct ending of stories based on various aspects, including feature-based models and neural models. Chaturvedi et al. [39] proposed a script learning model based on three semantic aspects: event-sequence, emotional trajectory, and topical consistency. While their work outperformed previous approaches, the researchers suggested using a thorough analysis of human behavior and societal norms to improve script learning. Lin et al. [95] proposed a similar feature-based model. Mostafazadeh et al. [116] trained a simple embedding model to predict the correct story ending based on the story context's embedding and the two alternative endings. Wang et al. [175] used generative adversarial networks, where the generative model generates a fake sample conditioned on the story context, and the discriminative model discriminates the real sample from the fake one. The discriminator had three models: an LSTM-RNN model to represent the sentence, an attention-based LSTM-RNN model to represent the document, and a bilinear model to calculate the context document and target sentence similarity. More recent studies [41, 74, 91, 93] have reported notable enhancements on SCT results when training on large datasets.

#### 2.1.4.3 Story Completion

Unlike previous research that predicts new story events by scoring known events, story completion aims to complete the plot when given a story context [65]. Most systems in this category conclude stories by generating a story ending based on previous story events.

Roemmele et al. [145] used the Children’s Book Test (CBT) dataset as a story corpus. Story generation starts by taking an initial story that contains 20 sentences as input and generating the next sentence based on CBR. Then, RNN is used to generate the last sentence word-by-word comparing with the original 21st sentence as a gold standard. This study’s main contribution is that it used several linguistic metrics to automate the evaluation of the generated stories. In addition, Hu et al. [72] proposed a context-aware hierarchical LSTM model that can predict future subevents given previous subevents. This model generates a sequence of words describing the future subevent. It considers two levels of the event sequence: the sequence of words and the temporal sequence of events. It also considers the story topic as an additional contextual feature.

Li et al. [92] proposed a Seq2Seq model trained using adversarial training to generate diversified story endings. They argued that traditional Seq2Seq models, trained purely by maximum likelihood estimation, are suitable for generation tasks where a gold standard exists. Nevertheless, this is not the case in story ending generation, where every proper ending is acceptable. To improve the quality of generated endings, the generator is encouraged to create endings similar to story endings written by humans. Therefore, a discriminator (a binary classifier) is trained to label the output as human generated or machine generated. This classification is used as a reward for the generator in the reinforcement learning algorithm. Zhao et al. [189] improved the accuracy and fluency of generated story endings by applying the copy and coverage mechanism to the traditional Seq2Seq model proposed by See et al. [156]. To avoid the out-of-vocabulary (OOV) problem, the copy mechanism is used to generate story endings directly from previous story events via pointing. The coverage mechanism is used to overcome the repetitive words problem by maintaining a coverage vector that keeps track of the attention history to adjust future attention. A new objective function of semantic relevance loss was added to maximize the semantic relevance between the generated ending and the story. It is calculated as the cosine similarity between the plot semantic vector and the semantic vector of the generated ending. Although the semantic vector of the generated ending is the encoder’s last hidden output, the plot semantic vector is calculated as proposed by Ma and Sun [102]. The generator was trained with a reinforcement learning algorithm that uses different evaluation metrics as reward functions to simulate the process of story generation by humans.

Guan et al. [65] proposed a neural model that generates a story ending considering two perspectives: story consistency and story implicit knowledge. All story events, attributes, and causal relationships between events play a role in story consistency. Therefore, story context clues were implemented by incremental encoding to maintain consistency. To mimic the human brain, which tends to understand a story and infer information based on its background knowledge, this model employs ConceptNet as a source of implicit knowledge and controls this knowledge through multi-source attention. This model was able to generate consistent story endings.

Wang et al. [176] proposed a model based on GPT-2 [137] to generate the missing parts of an incomplete story by conditioning the generated sentence on a previous sentence and a next sentence. Their model was able to create coherent stories that adhere to the provided end. Similarly, Wang and Wan [178] proposed a model for generating the missing story plot at any position for an incomplete story. Unlike the model of Wang et al. [176], this model can generate a sentence at the end of the story. It was adapted from the Transformer [174] by using shared attention layers for the encoder and decoder. BERT (Bidirectional Encoder Representations from Transformers) was used as the coherence discriminator. BERT is a new language representation model that is designed to pre-train deep bidirectional representations from unlabeled text [51].

#### 2.1.4.4 Story Generation

Researchers were motivated to use Seq2Seq models to generate complete stories due to their success in various NLP tasks.

Jain et al. [75] combined two off-the-shelf systems to create a story generator that generates stories when given a sequence of independent short descriptions. First, Statistical Machine Translation (SMT) was used to translate phrases independently within a sentence, and then a deep RNN was implemented to encode each sentence as a unit and decode them into comprehensive stories. However, the resulting summaries were not fully semantically related to the input description, and the overall scores of the applied evaluation metrics were not very high.

Choi et al. [44] trained an RNN model to generate stories by predicting the next sentence. The model consisted of two sub-models: RNN Encoder-Decoder (RNNED), which maps a sentence into a vector representation and vice versa, and RNN for Story Generator (RNNSG), which uses previously learned vectors to predict the next vector in the vectors sequence. The model was able to generate sentences with correct grammar and overall content, but misused some words in the generated sentences. Harrison et al. [66] used RNN to guide Markov Chain Monte Carlo (MCMC) sampling in generating stories, similar to the two-step process used by Choi et al. [44].

Although RNNs have been successful in many Seq2Seq problems, they did not reach expectations in story generation, as they often failed to generate coherent stories after a few sentences. This is because a story is a sequence of consistent events that is longer than an RNN can maintain, and as Khandelwal et al. [81] showed, RNNs depend on a relatively small part of the previous tokens for their predictions. As a result, as the story generation progresses, RNNs lose the connection between the currently generated event and previous, far-off events, which affects the consistency and coherence of the generated story.

Tambwekar et al. [167] created a controllable RNN story generator that is given a start and end state and uses reinforcement learning to guide the RNN from the start to the end. The reward function was designed based on two components: the distance between the next event and the final event, and the frequency of the next event occurring before the final event in the story corpus. Ammanabrolu et al. [6] applied the policy gradient deep reinforcement learner from Tambwekar et al. [167] to event generation and introduced four event-to-text models to improve the quality and interestingness of the generated story text. These models included a retrieve-and-edit model, a template filling model, a sequence-to-sequence with Monte Carlo beam decoding model, and a Seq2Seq with a finite state machine decoder. Results showed that an ensemble of these models performed better than the individual models. Fan et al. [54] decomposed story generation into two stages in order to improve the coherence of generated stories. They generated a story premise using a convolutional language model and transformed the premise into text using a Seq2Seq model with a fusion mechanism to improve the relevance between the story and premise and a self-attention mechanism to model long-range context. This approach inspired subsequent research on decomposing story generation into two stages.

Yao et al. [186] proposed a hierarchical story generator that uses both plot planning and text generation to create stories from given titles. They extract the storyline of each story in the corpus by identifying the most important words in each sentence using the RAKE algorithm [147] and then use two strategies to generate stories: the Dynamic Schema and the Static Schema. The Dynamic Schema generates the next word in the storyline and the next sentence in the story at each step, while the Static Schema generates the complete storyline and then translates it into text. They use a Seq2Seq model to translate the storyline into text, and the results show that planning the storyline leads to better stories in terms of fidelity, coherence, interestingness, and overall user preference, but the Static Schema produces more consistent and coherent stories than the Dynamic Schema.

Xu et al. [183] use a reinforcement learning method to learn the semantic dependency between sentences in a story by identifying the most critical phrases of a sentence, known as the skeleton, and then train a Seq2Seq model to generate coherent sentences based on the skeleton. However, coherence is negatively affected by the length of the input and the unfamiliarity of the input. Chen et al. [40] also generate an outline as an intermediate step before generating a story by using an off-the-shelf text summarizer to create high-level plots from the training corpus and then using natural language summaries to pre-train a planning model on how to generate outlines. They use a structured Seq2Seq model to generate a story given a title and an outline, and while this system outperforms similar previous systems [54, 183, 186], the authors suggest that more powerful mechanisms are needed to improve coherence at the story level.

Zhai et al. [188] proposed a hybrid model that can generate coherent stories from a small corpus by using an agenda generator to plan the story by sampling a path through a temporal script graph extracted from the story corpus, and a neural surface realization module to generate story text based on the story plan. They evaluated the global coherence of the stories in terms of inclusion, relevance, and order. Araz [10] proposed a transformer neural network for generating stories based on prompts, which produced novel and viable stories but also generated repetitions and grammatical errors and did not consider the prompts as closely as desired.

Large pre-trained language models have demonstrated strong abilities in processing natural language. Samples of text generated by the GPT-2 model [137] suggest that these models can produce text that is comparable to human writing. This has led researchers to use pre-trained models for story generation. See et al. [157] proposed two models: the pre-trained version of the Fusion model [125] and the smallest version of GPT2, known as GPT2-117 [137]. As in other works [10, 54], these models were trained to generate stories given prompts. Overall, the authors found that the GPT2-117 model performed better than the Fusion model in many aspects, but generated repetitive and less diverse text when using likelihood-maximizing decoding algorithms. The inefficiency of pre-trained models in story generation was also noted by Holtzman et al. [70], who observed that such models generate bland, incoherent, or repetitive text.

Guan et al. [64] argued that the weak performance of pre-trained models is due to a lack of information. Story generation, as an open-ended generation task, does not provide the model with a gold standard output to compare against, as in other generation tasks such as summarization. This lack of input hinders the learning process. Therefore, there is a need for a source of supporting knowledge. To address this, the authors proposed using commonsense knowledge from external knowledge bases to generate good stories, and also used multi-task learning to capture the causal and temporal dependencies between sentences in a story. Their model generated better stories compared to baseline models in terms of logic and global coherence. Inspired by Guan et al. [64], Xu et al. [184] proposed a controllable story generation framework that allows for the dynamic incorporation of commonsense knowledge into the language model. At each generation step, the model predicts a set of keywords based on the story context, and then uses these keywords to query a commonsense knowledge base for related concepts. The next sentence of the story is generated by the GPT-2 model, conditioned on both the story context and the top-ranked retrieved concepts.

Li et al. [94] proposed open-ended causal generation models based on Transformer. They used a causal relations corpus to train a cause model and an effect model, which generated high-quality and diverse causes and effects. To support diversity, they also developed an approach for disjunctive positive lexical constraints, which allows the decoder to select one of a set of provided words or phrases to be included in its output. This approach was used to choose among different morphological variants of the same lemma.

### 2.1.7 Story Evaluation

Automatic evaluation of stories is a challenging task in the field of story generation due to the subjectivity, diversity of evaluation criteria, and high dimensionality of story components. Most systems rely on human judgments to evaluate generated stories, but this method is inflexible, time-consuming, subjective, and lacks a gold standard for comparison. Human evaluators may also use their own knowledge and imagination to complete and rate inconsistent stories more highly than they deserve. [108, 129, 90, 144, 126]

There are several ways to evaluate the performance of story generation systems. One method is the narrative cloze test, which was proposed by Chambers and Jurafsky [36] to evaluate unsupervised script learning and generation. In this test, a sequence of story events is given with one event removed, and the system is asked to generate a ranked list of guesses for the missing event based on the seen events. The SCT, proposed by Mostafazadeh et al. [114], is a variation of the narrative cloze test that is designed for supervised learning approaches. It presents a system with a four-sentence story and two alternatives for the fifth sentence, labeled as a "right ending" and a "wrong ending," and measures the system's ability to choose the correct ending for each story. Granroth-Wilding and Clark [63] proposed the Multiple Choice Narrative Cloze (MCNC) test, which gives a system five randomly ordered events to choose the missing event from, allowing the system to use richer information about the context and the list of choices and better compare different story generation systems.

Once a benchmark or standard is established for comparing the various elements of a generated story, various metrics can be applied to assess the performance of the story generator.

**Statistical models.** These models predict story events based on several statistical criteria:

* **N-gram overlap**: As in other NLG tasks, story generation quality can be assessed by calculating the n-gram overlap between the predicted and expected events. This includes metrics such as BLEU, METEOR, CIDEr, and ROUGE. However, BLEU is the most widely used metric for story generation, e.g., see [7, 65, 105, 183, 186].
* **Perplexity**: Perplexity is an evaluation metric commonly used to assess the quality of language models. It measures the prediction ability of a model given the previous context where lower perplexity indicates better prediction accuracy. Perplexity [6] is defined as follows:



where x is a token in the text, and,



where Y is the vocabulary. Many ML models use perplexity, e.g., [7, 54, 105].

* **Pointwise Mutual Information**: PMI is used when an event is selected from several alternatives. It relies on word co-occurrence counts and chooses the event whose co-referring entity has the highest average PMI score within the story chain (see Section 5.2). Initially proposed by Chambers and Jurafsky [36], others have now adopted it, e.g., [76, 148].

**Embeddings models.** Embeddings models predict the story events based on embeddings, either at the word level or the sentence level. Different embedding-based metrics can be used, including Skip Thoughts Cosine Similarity (STCS), Embedding Average Cosine Similarity (EACS), Vector Extrema Cosine Similarity (VECS), and the Greedy Matching Score (GMS). The Average Maximum Similarity model proposed by Roemmele et al. [146] is a word-level embedding model that calculates the mean of the highest similarity embedding for each word of the ending, then selecting the ending with the highest mean. The Deep Structured Semantic model is another structured embedding model applied by Mostafazadeh [114] for the SCT. The Conditional Generative Adversarial Networks model was also applied in story generation, where the discriminator is used to choose the correct story ending [92].

There are several drawbacks to evaluating generated stories by comparing them to a reference. One issue is that treating story generation as a classification problem may result in systems that are good at classification but do not understand the semantics of a story and may not be creative enough in their generation. Additionally, the creative nature of generating a story means that it may not be interesting if it is overly predictive. Furthermore, there is no single "correct" answer in story generation, which can be at odds with evaluation methods that require systems to choose a specific answer or be penalized for not doing so.

There are several ways to evaluate story generation systems, including comparing the generated stories to a reference, analyzing the linguistic properties of the stories, using customized statistical evaluation measures, assessing the interestingness of the stories based on cognitive theories or social media metrics, and measuring the suspense level in the stories. Roemmele et al. [145] used lexical cohesion, style matching, and entity co-reference as linguistic evaluation measures for generated stories, while Purdy et al. [136] evaluated grammaticality, temporal ordering, local contextuality, and narrative productivity. Kartal et al. [80] calculated the believability of a generated story by multiplying the believability of each action in the story, while León and Gervás [89] used 13 variables such as interest, tension, causality, and hypotheses to guide the generation of stories based on accumulation of contributions, the appearance of patterns, and inference. Yao et al. [186] used inter- and intra-story repetition to measure diversity in generated stories. Wang et al. [177] used upvotes from social media as an approximate measure of story quality, and Sagarkar et al. [151] crowdsourced interestingness evaluations of story continuations. Behrooz et al. [21, 22] evaluated story interestingness based on unexpectedness and the ability to generate predictive inference, while O'Neill and Riedl [122] measured suspense in stories by calculating the cost of escape plans at different time-slices and finding the area under the resulting suspense curve.

### 2.1.8 Challenges in Automatic Story Generation

Despite the significant progress that has been made in the field of automatic story generation, the field has not seen as much progress as one might expect, and it still faces a number of challenges and limitations.

**Dispersion**. The lack of a common set of domain knowledge and evaluation criteria makes it difficult to compare the performance of different story generation systems. This can lead to the success of a system being attributed to its domain knowledge rather than its generation capabilities, and to the evaluation of a system being based on personal human opinion rather than objective metrics. Standardizing these elements would allow for a more accurate assessment of the strengths and weaknesses of different models, and facilitate the building upon and improvement of previous work. However, despite the long history of automatic story generation, no existing corpora or evaluation metrics have become widely accepted as standards. In recent years, some researchers have begun to reuse existing story corpora and commonsense knowledge bases, but more work is needed to establish a standard set of resources and metrics for evaluating story generation systems.

**Domain knowledge**. Story generation systems have traditionally relied on manually crafted domain models to produce closed-domain stories, but this has limited their ability to be extended to other domains. Recently, advances in data science and machine learning have led to the development of open-domain story generators that use various sources of knowledge, including story corpora, crowdsourced data, commonsense knowledge, and semantic relations corpora. While these sources can be useful for enhancing the knowledge of both planning-based and machine learning story generation systems, they also present challenges, such as the lack of standardization and the limited use of semantic relations corpora.

**Seq2Seq models**. There are open-domain story generators that use machine learning to create stories, but their performance has been disappointing and has lagged behind many story generators with manually crafted knowledge. Three main limitations have contributed to the modest performance of sequence-to-sequence story models: losing consistency over the course of the story, generating repetitive words, and the problem of out-of-vocabulary (OOV) words. To address these limitations, researchers have proposed hierarchical models that decompose story generation into a multi-level problem and have applied reinforcement learning for plot generation, as well as the coverage model and heuristics to reduce repetition and the pointer mechanism to address the OOV problem.

**Pre-trained language models**. Pre-trained language models have made significant advancements in various fields of natural language processing research, but have not yet achieved the same level of success in natural language generation tasks, including story generation. These models tend to struggle with repetition, logical conflicts, and a lack of long-range coherence. They also have difficulty with commonsense inference. While there have been attempts to use pre-trained language models in story generation, their performance has not been as impressive as expected. One reason for this may be that these models view stories as simply textual pieces, while stories are much more complex structures that require the integration of various subtasks. Combining deep learning models with other generation approaches or extending them with knowledge resources may be able to produce better stories.

**Story interestingness**. To create an interesting story, it is essential to consider both its consistency and structure. While some models focus on one of these aspects, there are only a few systems that aim to balance both. Hierarchical models, which generate a storyline first and then generate the story based on that storyline, have the potential to create stories that are interesting, structured, and consistent. These models can also incorporate elements from cognitive science and literature to increase the interestingness of the generated story. However, most existing story generators do not adequately address all of these factors.

**Objective evaluation**. One of the main reasons that generated stories are often low quality is the lack of effective automatic evaluation metrics, particularly in machine learning (ML) approaches where evaluation is important for guiding the learning process. Early ML story generators often used common natural language generation (NLG) metrics, such as BLEU, ROUGE, and perplexity, which are not suitable for open-domain generation because they require a gold standard for comparison and do not align with the creative nature of story generation. They also do not correlate well with human judgments [96]. While various approaches have been used to evaluate generated stories, more recent models have attempted to emulate human judgments by using criteria such as causality and suspense. It seems that the adoption of more cognitive science-based theories of interestingness would be helpful for implementing automatic story evaluation algorithms.

## 2.2 Prompt Based Learning

### 2.2.1 Brief History of NLP

Fully supervised learning has long been the fundamental paradigm in Natural Language Processing (NLP), and machine learning in general (Kotsiantis, 2007). Under this paradigm a model is trained specifically for a target task using a set of source-target samples. As fully supervised datasets proved insufficient for learning high-quality models, early NLP models relied heavily on feature engineering (Zhang and Nivre, n.d.; Lafferty et al., 2001; Guyon et al., 2002; Och and Ney, 2004). NLP researchers or engineers utilised their domain expertise to identify and extract important features from raw data and provide models with the appropriate inductive bias to learn from. As neural networks started being applied to NLP problems, feature selection and engineering became part of the model training itself (Collobert et al., 2011; Bengio et al., 2013). This led to the next paradigm of architecture engineering, where inductive bias was provided through careful design of network architecture (Hochreiter and Schmidhuber, 1997; Bahdanau et al., 2014; Chung et al., 2014; Kalchbrenner et al., 2014; Vaswani et al., 2017).

Towards the end of 2010’s, the standard for model training in NLP shifted to pre-train and fine-tune paradigm (Openai et al., 2018; Peters et al., 2018; Dong et al., 2019; Yang et al., 2019; Lewis et al., 2020; Zhang et al., 2020). This paradigm introduced Language Models (LM), models with (almost) fixed architecture, trained to predict the probability of observed textual data. As the raw textual data required to train LMs is widely available, they can be trained on large datasets and learn general-purpose features of the language they are modelling. This led to objective engineering, where pre-trained LMs were fine-tuned for downstream tasks using task-specific objective functions. For example, (Zhang et al., 2020) show that a text summarization model can be trained by adding a loss function for predicting important sentences from a document.

As of 2021, a new approach to natural language processing called "pre-train, prompt, and predict" has emerged. Instead of adapting pre-trained language models to specific tasks through engineering, this approach reformulates downstream tasks to be more similar to those encountered during the original training of the language model, using a textual prompt. For example, a prompt like "I felt so" can be used to ask a language model to fill in the blank with an emotion-bearing word in response to the social media post "I missed the bus today." Similarly, a prompt like "English: I missed the bus today. French:" can be used to ask a language model to translate the phrase into French. This approach allows a single, unsupervised language model to solve a wide range of tasks by selecting the appropriate prompts. However, finding the most effective prompts for a given task requires careful engineering. (Radford et al., 2019; Petroni et al., 2019; Brown et al., 2020; Raffel et al., 2020; Schick and Schütze, 2021b; Gao et al., 2021; Sun et al., 2021)

### 2.2.2 Introduction to Prompting

One of the main challenges of supervised learning is the need for large amounts of labeled data in order to train a model to perform a specific task. NLP approaches that use prompts aim to address this issue by training a language model (LM) to model the probability of text itself, rather than the probability of a specific output given input. This allows the LM to be used to predict the desired output without requiring a large amount of labeled data for the specific task.

#### 2.2.2.1 Prompt Addition

To generate a prompt, a prompt-function is applied to the input text, resulting in a modified version of the text. This function usually follows a two-step process: (1) it applies a template, which is a string with placeholders for the input text and an intermediate generated answer that will eventually be mapped to the desired output; and (2) it fills the placeholder for the input text with the actual input. For example, in sentiment analysis, the template might be "Overall, [X] was a [Z] movie," and the input text "I love this movie." would be transformed into the prompt "I love this movie. Overall, it was a [Z] movie." In machine translation, the template might be "Finnish: [X] English: [Z]," where the input and answer are labeled with language headers. These prompts can be either "cloze" prompts, with a placeholder in the middle of the text, or "prefix" prompts, with the input text coming before the placeholder. In some cases, the placeholder may be represented by a numerical id or a continuous vector, rather than a natural language token. The number of [X] and [Z] placeholders can be adjusted as needed for different tasks. (Kumar et al., 2016; McCann et al., 2018; Radford et al., 2019; Schick and Schütze, 2021a)

#### 2.2.2.2 Answer Search

Next, the goal is to find the highest-scoring text, denoted as ẑ, that maximizes the score of the language model (LM). To do this, we define a set of permissible values for the intermediate generated answer, denoted as Z. This set can either be the entire language for generative tasks, or a smaller subset of words for classification tasks (e.g. defining Z = { "excellent", "good", "OK", "bad", "horrible" } to represent the classes in Y = { ++, +, ~, -, -- }). We then define a function, ffill, that fills in the placeholder in the prompt with a potential answer from Z, resulting in a "filled" prompt. If the filled prompt includes the true answer, we refer to it as an "answered" prompt. Finally, we search over the set of potential answers in Z by calculating the probability of the corresponding filled prompts using the pre-trained LM. This search can be either an argmax search that finds the highest-scoring output, or sampling that generates outputs randomly according to the probability distribution of the LM.

#### 2.2.2.3 Answer Mapping

Finally, the goal is to convert the highest-scoring answer, ẑ, into the highest-scoring output, ŷ. In some cases, the answer itself is the output (e.g. in language generation tasks like translation), but in other cases, multiple answers may correspond to the same output. For example, multiple sentiment-bearing words (e.g. "excellent", "fabulous", "wonderful") may represent a single class (e.g. "++"). In these cases, it is necessary to map the searched answer to the desired output value.

#### 2.2.2.4 Design Considerations for Prompting

In this section, we outline several design considerations that go into the development of a prompting method for NLP tasks:

* Pre-trained Model Choice: There are many pre-trained language models (LMs) that can be used as backbone.
* Prompt Engineering: The choice of prompt has a significant impact on the accuracy and nature of the task that the model performs.
* Answer Engineering: Depending on the task, it may be necessary to design the set of permissible values for the intermediate generated answer, Z, and the mapping function between answers and outputs differently.
* Expanding the Paradigm: The equations presented earlier represent a basic framework for prompting methods, but there are many ways to expand upon and modify this paradigm to improve results or adapt it to different tasks.
* Prompt-based Training Strategies: There are also methods for training the parameters of the prompt, the LM, or both.

### 2.2.3 Pre-trained Language Models

There are already several comprehensive surveys available on the impact of pre-trained language models (LMs) on natural language processing (NLP) in the "pre-train and fine-tune" paradigm (Raffel et al., 2020; Qiu et al., 2020; Xu et al., 2021; Doddapaneni et al., 2021). This section presents a systematic view of various pre-trained LMs that focuses on aspects relevant to prompting methods.

#### 2.2.3.1 Training Objectives

The main training objective of a pre-trained language model (LM) is typically to predict the probability of text x. Standard language model (SLM) objectives focus on optimizing the probability P (x) of text from a training corpus (Radford et al., 2019). The text is typically predicted in an autoregressive fashion, meaning that the model predicts the tokens in the sequence one at a time, usually from left to right (though other orders are possible). An alternative to SLM objectives are denoising objectives, which apply some noising function x̃ = fnoise (x) to the input sentence and try to predict the original input sentence given the noised text P (x|x̃). There are two common types of denoising objectives: corrupted text reconstruction (CTR) objectives, which restore only the noised parts of the input sentence to their uncorrupted state, and full text reconstruction (FTR) objectives, which reconstruct the entire text, whether noised or not (Lewis et al., 2020a).

#### 2.2.3.2 Noising Functions

In reconstruction-based training objectives, the type of corruption applied to create the noised text x̃ can affect the performance of the learning algorithm. Additionally, prior knowledge can be incorporated by controlling the type of noise. For example, noise could be applied to entities within a sentence to encourage the model to be better at predicting entities. There are several types of noising functions that can be used, as detailed in Table 4. Masking (e.g. Devlin et al., 2019) involves replacing a token or multiple-token span with a special token like [MASK]. Masking can be either random or specifically designed to introduce prior knowledge. Replacement (e.g. Raffel et al., 2020) is similar to masking, but the token or multiple-token span is replaced with another token or piece of information, such as an image region (Su et al., 2020). Deletion (e.g. Lewis et al., 2020a) involves removing tokens or multiple-token spans from the text without adding a [MASK] or any other token. This operation is often used in conjunction with the full text reconstruction loss. Permutation (e.g. Liu et al., 2020a) involves dividing the text into different spans (tokens, sub-sentential spans, or sentences) and rearranging them into a new text.

#### 2.2.3.3 Directionality of Representations

The directionality of representation calculation is an important factor to consider when understanding pre-trained LMs. In general, there are two common ways to calculate such representations: left-to-right, where the representation of each word is based on the word itself and all previous words in the sentence; and bidirectional, where the representation of each word is based on all words in the sentence, including words to the left of the current word. It is also possible to combine these strategies or condition the representations in a random permuted order, but these methods are less commonly used. The directionality of representation calculation is often implemented through attention masking, which masks out the values in an attention model, such as the Transformer architecture (Vaswani et al., 2017).

#### 2.2.3.4 Typical Pre-training Methods

Four popular pre-training methods are mentioned below.

Left-to-Right Language Model

Left-to-right language models (L2R LMs) are a type of auto-regressive language model that predict the next word or assign a probability to a sequence of words (x1, x2,..., xn) (Jurafsky and Martin, 2021). The probability is calculated using the chain rule in a left-to-right fashion: P (x) = P (x1) × P (x2 |x1) ×...× P (xn |x1...xn-1). L2R LMs have been in use since they were first proposed by Markov in 1913 (Markov, 2006) and have been used in both count-based (Goodman, 2001) and neural forms (Bengio et al., 2003; Mikolov et al., 2010; Radford and Narasimhan, 2018). Examples of modern pre-trained L2R LMs include GPT-3 (Brown et al., 2020) and GPT-Neo (Black et al., 2021). These LMs are popular for use as the backbone of prompting methods (Radford et al., 2019; Brown et al., 2020) because they are often large and difficult to train, or not available for public use, making it impractical to use them in a pre-train and fine-tune regimen.

Masked Language Models

MLMs, or masked language models, are a type of bidirectional objective function that are used widely in representation learning. They aim to predict masked pieces of text based on their surrounding context, as in the example P(xi|x1,...,xi−1,xi+1,...,xn) which represents the probability of the word xi given its surrounding context. MLMs are often used in prompting methods for tasks related to natural language understanding or analysis, such as text classification, natural language inference, and extractive question answering. Examples of pre-trained models using MLMs include BERT (Devlin et al., 2019) and ERNIE (Zhang et al.,2019; Sun et al., 2019b). MLMs have also been used in combination with fine-tuning in some prompting methods.

Prefix and Encoder-Decoder

There are two common architectures for pre-trained models that are used for conditional text generation tasks, such as translation and summarization, where an input text x is given and the goal is to generate target text y. These architectures both involve using an encoder with a fully-connected mask to encode the source text x, and then using a left-to-right language model to decode the target text y in an autoregressive manner. The first architecture, called the prefix language model (UniLM 1-2 (Dong et al., 2019; Bao et al., 2020) and ERNIE-M (Ouyang et al., 2020)), decodes y while being conditioned on a prefixed sequence x, which is encoded using the same model parameters but with a fully-connected mask. The second architecture, called the encoder-decoder model (T5 (Raffel et al., 2020), BART (Lewis et al., 2020a), MASS (Song et al., 2019) and their variants), uses a separate encoder and decoder for the input and output text, with the parameters of the encoder and decoder not being shared. These models can be naturally used for text generation tasks with or without prompting, and have also been applied to tasks such as information extraction, question answering, and text generation evaluation through the use of appropriate prompts. Prompting methods can broaden the applicability of these generation-oriented models and also allow for unified modeling across different tasks.

### 2.2.4 Prompt Engineering

Prompt engineering is the creation of a function to optimize performance on a specific downstream task. This can involve finding the best prompt template for a given task through manual or automated methods, taking into consideration the shape of the prompt.

#### 2.2.4.1 Prompt Shape

The choice of prompt can affect the performance of the pre-trained model on a downstream task. Cloze prompts, which involve filling in blanks in a text, work well with tasks solved using masked language models (Petroni et al., 2019; Cui et al., 2021). On the other hand, prefix prompts, which involve continuing a string, are better suited for tasks involving generation or auto-regressive language models (Li and Liang, 2021; Lester et al., 2021). Full text reconstruction models can be used with either type of prompt. When dealing with tasks that require multiple inputs, such as text pair classification, prompt templates must include space for multiple inputs.

#### 2.2.4.2 Manual Template Engineering

One way to create prompts is by manually creating templates based on human intuition. This method has been used in the creation of the LAMA dataset (Petroni et al., 2019), which provides cloze prompts for probing knowledge in language models. Manually crafted prefix prompts have also been used in a variety of tasks, including question answering, translation, and common sense reasoning (Brown et al., 2020). Pre-defined templates have also been used in few-shot learning settings for tasks such as text classification and conditional text generation (Schick and Schütze, 2020, 2021a,b).

#### 2.2.4.3 Automated Template Engineering

There are several issues with the manual creation of prompt templates, including the time and experience required to do so, and the potential for even experienced designers to fail in finding optimal prompts (Jiang et al., 2020c). To address these problems, automated methods have been developed to design prompt templates. These methods can be divided into those that use discrete prompts, which are actual text strings, and those that use continuous prompts, which are described directly in the embedding space of the underlying language model. Additionally, prompt functions can be either static, using the same template for each input, or dynamic, generating a custom template for each input.

Discrete Prompts

There are several methods for automatically discovering prompts in a discrete space, usually corresponding to natural language phrases. These methods include:

* Prompt mining, where frequent words or dependency paths between inputs and outputs are used as templates Jiang et al. (2020c).
* Prompt paraphrasing, where an existing seed prompt is paraphrased into a set of other candidates and the one that achieves the highest training accuracy on the target task is selected (Jiang et al., 2020c) (Yuan et al., 2021b) (Haviv et al., 2021).
* Gradient-based search, where short sequences are found to trigger the desired target prediction using an iterative search process over tokens in the prompt Wallace et al. (2019a) Shin et al. (2020)
* Prompt generation, where natural language generation models are used to generate template tokens for downstream tasks Gao et al. (2021) Ben-David et al. (2021)
* Prompt scoring, where a set of hand-crafted templates are scored using a language model and the one with the highest probability is selected for each individual input Davison et al. (2019).

Continuous Prompts

The purpose of prompt construction in natural language processing (NLP) is to find a method that allows a language model (LM) to perform a task effectively. One approach to prompt construction is to use continuous prompts, also known as soft prompts, which perform prompting directly in the embedding space of the model rather than using human-interpretable natural language. Continuous prompts remove two constraints: they allow the embeddings of template words to be the embeddings of words in any language, rather than just natural language, and they allow the template to have its own parameters that can be tuned based on data from the downstream task rather than being parameterized by the pre-trained LM's parameters. There are several methods that use continuous prompts in NLP.

* **Prefix Tuning** Prefix Tuning is a method proposed by Li and Liang (2021) for prompt construction in natural language processing (NLP) that involves prepending a sequence of continuous task-specific vectors to the input while keeping the language model (LM) parameters frozen. This approach has been found to be more sensitive to different initializations in low-data settings compared to the use of discrete prompts with real words. Lester et al. (2021) also use a prefix approach by prepending the input sequence with special tokens and tuning the embeddings of these tokens directly, which adds fewer parameters compared to Li and Liang's method. Tsimpoukelli et al. (2021) use a vision encoder to encode images into a sequence of embeddings that can be used to prompt a frozen auto-regressive LM to generate a caption, and demonstrate that this approach can perform few-shot learning for vision-language tasks. The prefix used in Tsimpoukelli et al.'s method is sample-dependent, meaning it is a representation of input images rather than a task embedding, unlike the prefixes used in the previous two works.
* **Tuning Initialized with Discrete Prompts** There are also methods for prompt construction in natural language processing (NLP) that initialize the search for a continuous prompt using a prompt that has already been created or discovered using discrete prompt search methods. For example, Zhong et al. (2021b) first define a template using a discrete search method such as AUTO PROMPT (Shin et al., 2020) and then fine-tune the embeddings of virtual tokens based on this discovered prompt to improve task accuracy. They find that initializing with manual templates can provide a better starting point for the search process. Qin and Eisner (2021) propose learning a mixture of soft templates for each input, where the weights and parameters for each template are jointly learned using training samples. The initial set of templates they use can be either manually crafted or obtained using the "prompt mining" method. Hambardzumyan et al. (2021) also introduce the use of a continuous template, but the shape of this template follows a manual prompt template.
* **Hard-Soft Prompt Hybrid Tuning** There are also methods for prompt construction in natural language processing (NLP) that insert some tunable embeddings into a hard prompt template rather than using a purely learnable prompt template. Liu et al. (2021b) propose "P-tuning," which involves learning continuous prompts by inserting trainable variables into the embedded input and representing prompt embeddings using a bidirectional long short-term memory (BiLSTM) network. P-tuning also introduces the use of task-related anchor tokens within the template, such as "capital" in relation extraction, which are not tuned during training. Han et al. (2021) propose "prompt tuning with rules" (PTR), which uses manually crafted sub-templates and logic rules to compose a complete template. To enhance the representation ability of the resulting template, they also insert virtual tokens whose embeddings can be tuned together with the pre-trained LM's parameters using training samples. The template tokens in PTR include both actual tokens and virtual tokens, and experimental results show the effectiveness of this prompt design method in relation classification tasks.

### 2.2.5 Answer Engineering

Answer engineering is the process of searching for an answer space and a map to the original output that results in an effective predictive model, in contrast to prompt engineering which involves designing appropriate inputs for prompting methods. In order to perform answer engineering, it is necessary to consider two dimensions: the shape of the answer and the method used to design the answer.

#### 2.2.5.1 Answer Shape

The shape of an answer refers to its granularity and can take several forms in natural language processing (NLP) tasks. One option is to use tokens, which can be individual words from the pre-trained language model's vocabulary or a subset of the vocabulary. Another option is to use spans, which are short multi-token spans that are often used with cloze prompts. Sentences or documents can also be used as answers, which are common with prefix prompts. The choice of answer shape depends on the task being performed. Token or text-span answers are often used in classification tasks such as sentiment classification (Yin et al., 2019) and relation extraction (Petroni et al., 2019), as well as named entity recognition (Cui et al., 2021). Phrasal or sentential answers are commonly used in language generation tasks (Radford et al., 2019) and multiple-choice question answering (Khashabi et al., 2020), where the scores of multiple phrases are compared.

#### 2.2.5.2 Answer Shape Design Methods

In order to effectively perform answer engineering, it is necessary to consider not only the shape of the answer, but also how to design the appropriate answer space and the mapping to the output space if the answers are not used as the final outputs.

Manual Answer Search

There are two main strategies for designing the answer space and mapping it to the output space in manual design: unconstrained spaces and constrained spaces. In the case of unconstrained spaces, the answer space is often the space of all tokens (Petroni et al., 2019), fixed-length spans (Jiang et al., 2020a), or token sequences (Radford et al., 2019), and the answer is directly mapped to the final output using the identity mapping. On the other hand, constrained spaces are often used for tasks with a limited label space such as text classification or entity recognition, or multiple choice question answering. For example, Yin et al. (2019) and Cui et al. (2021) manually design lists of words or labels to be used as the answer space for text classification and entity recognition tasks, respectively. In multiple-choice question answering, it is common to use a language model to calculate the probability of an output among multiple choices (Zweig et al., 2012).

Discrete Answer Search

There is less research on automatic answer search compared to prompt search in natural language processing (NLP), but some methods have been developed for both discrete and continuous answer spaces. Answer paraphrasing involves expanding an initial answer space by generating paraphrased answers using a method such as back-translation (Jiang et al., 2020b). Prune-then-search methods first generate an initial pruned answer space and then use an algorithm to search within this space to select the final set of answers. Schick and Schütze (2021a) and Schick et al. (2020) identify frequent tokens with at least two alphabetic characters in an unlabeled dataset as the initial answer space, while Shin et al. (2020) use a logistic classifier based on the contextualized representation of the [Z] token. Gao et al. (2021) select top-k vocabulary words based on their generation probability and further prune the answer space based on zero-shot accuracy on training samples. Label decomposition, used in relation extraction tasks by Chen et al. (2021b), involves decomposing each relation label into its constituent words and using them as the answer, with the probability of the answer span being the sum of the probabilities of each token.

Continuous Answer Search

Hambardzumyan et al. (2021) are one of few works that have explored the use of continuous, or "soft," answer tokens that can be optimized through gradient descent. They assign a virtual token for each class label and optimize the token embedding for each class together with the prompt token embeddings. Unlike other methods, this approach does not make use of the embeddings learned by the language model and instead learns a new embedding from scratch for each label.

### 2.2.7 Training Strategies for Prompting Methods

The methods described in the previous sections provide a way to obtain appropriate prompts and corresponding answers. The next step is to consider training strategies that involve explicitly training models in concert with prompting methods.

#### 2.2.7.1 Training Settings

In many cases, prompting methods can be used without any explicit training of the language model for the downstream task by applying a pre-trained language model to fill in cloze or prefix prompts. This is known as the zero-shot setting, as there is no training data for the task of interest. However, there are also methods that use training data to train the model in conjunction with prompting methods. These methods fall into two categories: full-data learning, where a large number of training examples are used to train the model, and few-shot learning, where a small number of examples are used to train the model. Prompting methods are particularly useful in few-shot learning, as there are generally not enough training examples to fully specify the desired behavior and the prompt can be used to guide the model in the right direction. It is worth noting that, for many of the prompt engineering methods described in the previous section, although annotated training samples are not explicitly used in the training of the downstream task model, they are often used in the construction or validation of the prompts that the downstream task will use. This means that it is not true zero-shot learning with respect to the downstream task (Perez et al., 2021).

#### 2.2.7.2 Parameter Update Methods

In prompt-based downstream task learning, there are typically two types of parameters: those from the pre-trained model and those from the prompts. Deciding which parameters to update is an important design decision that can affect the model's performance in different scenarios. There are five tuning strategies based on (i) whether the parameters of the underlying language model are tuned, (ii) whether there are additional prompt-related parameters, and (iii) if there are additional prompt-related parameters, whether those parameters are tuned.

Promptless Fine-tuning

The pre-train and fine-tune strategy, which involves updating all or some of the parameters of a pre-trained language model using gradients induced from downstream training samples, has been widely used in natural language processing (NLP) before the popularization of prompting methods. This strategy is referred to as promptless fine-tuning to contrast with the prompt-based learning methods described in the following sections. Examples of pre-trained models that have been fine-tuned in this way include BERT (Devlin et al., 2018) and RoBERTa (Liu et al., 2019). While this is a simple and effective method, it may not perform well on small datasets and can result in catastrophic forgetting, where the language model loses the ability to do things it was previously able to do (McCloskey and Cohen, 1989). The advantages of promptless fine-tuning include simplicity and the ability to fit to larger training datasets, but it has the disadvantage of potentially overfitting or not learning stably on smaller datasets (Dodge et al., 2020).

Tuning-free Prompting

Tuning-free prompting generates answers without changing the parameters of the pre-trained language model based only on a prompt, as described in the simplest form of prompting in section 2. This can be combined with the augmentation of input with answered prompts, a method referred to as in-context learning (Brown et al., 2020). Examples of tuning-free prompting include LAMA (Peters et al., 2019) and GPT-3 (Brown et al., 2020). The advantages of tuning-free prompting include efficiency, the ability to avoid catastrophic forgetting, and applicability in zero-shot settings. However, this method requires heavy engineering to achieve high accuracy, and in the in-context learning setting, providing many answered prompts can be slow at test time and may not be able to use large training datasets.

Fixed-LM Prompt Tuning

Fixed-LM prompt tuning is a method that updates only the parameters of the prompts, which are additional parameters introduced for the task, using supervision from the downstream training samples, while keeping the pre-trained language model fixed. Examples of this include Prefix-Tuning and WARP [55]. This method has the advantage of retaining knowledge in the language model and being suitable for few-shot scenarios, but is not applicable in zero-shot scenarios. It can also have limited representation power in large-data settings and requires prompt engineering through the choice of hyperparameters or seed prompts. The prompts generated are usually not interpretable or manipulable by humans.

Fixed-prompt LM Tuning

In the scenario where both the pre-trained model's parameters and additional prompt-related parameters are updated, both the LM and prompts are fine-tuned in concert. This can lead to improvements in accuracy for certain tasks, particularly in few-shot scenarios. An example of this is the work by Logan IV et al. (2021), who observe that the prompt engineering can be reduced by allowing for a combination of answer engineering and partial LM fine-tuning. They define a simple template, called the null prompt, where the input and mask are directly concatenated without any template words, and find that this achieves competitive accuracy. Advantages of this approach include the ability to more completely specify the task through prompt or answer engineering, leading to more efficient learning in few-shot scenarios. However, it still requires some amount of prompt or answer engineering, and the fine-tuned LMs may not be effective on other tasks.

Prompt+LM Tuning

In this setting, both the parameters of the pre-trained models and the prompt-relevant parameters are fine-tuned using the supervision signal from downstream training samples. Representative examples include PADA [8] and P-Tuning [103]. This method is similar to the standard pre-train and fine-tune paradigm, but the addition of the prompt can provide additional guidance at the start of model training. This method is likely suitable for high-data settings, but it requires training and storing all parameters of the models, which may lead to overfitting on small datasets.

### 2.2.9 Prompt-relevant Topics

In this section, the connections between prompt learning and other learning methods are discussed.

#### 2.2.9.1 Few-shot Learning

Few-shot learning is a method for learning machine learning systems in scenarios with a small number of training examples. There are many ways to achieve few-shot learning, including:

* Model agnostic meta-learning: (Finn et al., 2017b) Learning features rapidly adaptable to new tasks
* Embedding learning: (Bertinetto et al., 2016) Embedding each sample in a lower-dimensional space where similar samples are close together
* Memory-based learning: (Kaiser et al., 2017) Representing each sample by a weighted average of contents from the memory

Prompt augmentation is another way to achieve few-shot learning, also known as priming-based few-shot learning. It involves prepending labeled examples to the current sample in order to elicit knowledge from a pre-trained language model without any parameter tuning. (Finn et al., 2017b; Bertinetto et al., 2016; Kaiser et al., 2017; Wang et al., 2020; Kumar and Talukdar, 2021)

#### 2.2.9.2 QA-based Task Formulation

Previous work on QA-based task formulation involves conceptualizing various NLP tasks as question answering problems, with the goal of unifying multiple tasks within this framework. This approach is similar to prompting methods, which use textual questions to specify tasks to be performed and aim to effectively use the knowledge in pre-trained language models. However, previous work on QA formulation did not focus extensively on utilizing pre-trained language models. Examples of this type of work include (Kumar et al., 2016; McCann et al., 2018) for unifying multiple NLP tasks, (Li et al., 2020; Wu et al., 2020) for information extraction, and (Chai et al., 2020) for text classification.

#### 2.2.9.3 Controlled Generation

Controlled generation is a method that involves adding additional guidance beyond the input text to a generation model in order to control the generated text (Yu et al., 2020). This guidance can take the form of style tokens (Sennrich et al., 2016b; Fan et al., 2018), length specifications (Kikuchi et al., 2016), domain tags (Chu et al., 2017), keywords (Saito et al., 2020), relation triples (Zhu et al., 2020), key phrases or sentences (Grangier and Auli, 2018; Liu et al., 2021c) to decide content, or other types of information. Some prompting methods can be seen as a form of controlled generation, as they use a prompt to specify the task and add extra information to the input text. Both controlled generation and prompting methods involve adding additional, learnable parameters to the input text, and when controlled generation is used with a seq2seq-based pre-trained model, it can be considered a form of prompt learning with input-dependent prompts and the "prompt+LM fine-tuning" strategy, where both the prompt's and the pre-trained LM's parameters can be fine-tuned. Examples include GSum (Dou et al., 2021).

There are some key differences between controlled generation and prompt-based text generation. First, controlled generation often focuses on controlling the style or content of the generated text, while keeping the underlying task the same (Fan et al., 2018; Dou et al., 2021), and does not necessarily require a pre-trained model. In contrast, the main purpose of using prompts for text generation is to specify the task itself and make better use of a pre-trained model (Li and Liang, 2021). Second, most current research on prompt learning in text generation uses prompts at the dataset or task level (Li and Liang, 2021), rather than input-dependent prompts (Tsimpoukelli et al., 2021), which is a common approach in controlled text generation and could be an area for future research on prompt learning.

#### 2.2.9.4 Data Augmentation

Data augmentation involves modifying existing data to increase the amount of data available for training (Fadaee et al., 2017; Ratner et al., 2017). Scao and Rush (2021) found that adding prompts to a dataset can lead to similar accuracy improvements as adding hundreds of additional data points across classification tasks, suggesting that using prompts for a downstream task is similar to implicitly conducting data augmentation.

### 2.2.10 Challenges in Prompting

Despite the promising results of prompt-based learning across various tasks and scenarios, there are still several challenges that need to be addressed. Some of these challenges are discussed below.

#### 2.2.10.1 Prompt Design

Most research on prompt-based learning has focused on text classification or generation tasks, with less attention given to information extraction and text analysis tasks. This is likely because the design of prompts is more challenging for these types of tasks. In the future, it may be necessary to reformulate these tasks as classification or text generation problems or to use effective answer engineering to express structured outputs in a suitable textual format. Another challenge is how to express structured information in prompts, such as tree, graph, table, or relational structures. Chen et al. (2021b) used additional marks to encode lexical information and Aghajanyan et al. (2021) used structured prompts based on HTML for web text generation, but more complicated structures have not been widely explored. Additionally, there is the question of how to simultaneously search or learn for the best combination of template and answer, as the performance of a model depends on both. Some research has focused on selecting answers before templates (Gao et al., 2021; Shin et al., 2020), while Hambardzumyan et al. (2021) have shown the potential for simultaneously learning both.

#### 2.2.10.2 Answer Engineering

There are two main challenges for answer engineering in classification tasks. First, when there are many classes, it can be difficult to select an appropriate answer space. Second, when using multi-token answers, it is not clear how to best decode multiple tokens using language models, although some methods have been proposed (Jiang et al., 2020a). For text generation tasks, qualified answers can be semantically equivalent but syntactically diverse, and most research has focused on using a single answer, with only a few exceptions (Jiang et al., 2020c). There is still an open research question of how to better guide the learning process with multiple references in text generation tasks.

#### 2.2.10.3 Selection of Tuning Strategy

There are various methods for tuning the parameters of prompts, language models, or both. However, the field lacks a comprehensive understanding of the tradeoffs between these methods. It would be helpful to conduct systematic explorations similar to those performed in the pre-train and fine-tune paradigm to understand the tradeoffs between different strategies (Peters et al., 2019).

#### 2.2.10.4 Multiple Prompt Learning

Prompt ensembling involves using multiple prompts, which can increase space and time complexity. There is a lack of research on how to distill knowledge from different prompts. Schick and Schütze (2020, 2021a,b) used an ensemble model to annotate a large dataset to distill knowledge from multiple prompts, but it is not clear how to select ensemble-worthy prompts or how to apply ensemble learning in text generation tasks. Prompt composition and decomposition involve breaking down a complicated task input into multiple sub-prompts, but it is not clear how to choose between them. Empirically, prompt decomposition is suitable for token or span prediction tasks, while prompt composition is better for span relation prediction tasks. Prompt augmentation methods are limited by input length, so there is a need to research how to select and order informative demonstrations. Finally, prompt sharing involves applying prompt learning to multiple tasks, domains, or languages, but there is little research in this area. It is not clear how to design individual prompts for different tasks or how to modulate their interaction with each other.

#### 2.2.10.5 Selection of Pre-trained Models

There are many pre-trained language models to choose from for prompt-based learning, but it is not clear how to select the best one. While there are conceptual introductions on how to choose pre-trained models for different NLP tasks, there are few systematic comparisons of the benefits of prompt-based learning for different pre-trained language models.

#### 2.2.10.6 Theoretical and Empirical Analysis of Prompting

While prompt-based learning has shown success in many situations, there is a lack of theoretical analysis and guarantees for this approach. Wei et al. (2021) showed that soft-prompt tuning can relax the non-degeneracy assumptions needed for downstream recovery in text classification tasks, making it easier to extract task-specific information. Saunshi et al. (2021) verified that text classification tasks can be reformulated as sentence completion tasks, making language modeling a meaningful pre-training task. Scao and Rush (2021) found that prompting is often equivalent to adding hundreds of data points across classification tasks.

#### 2.2.10.7 Transferability of Prompts

It is important to understand the extent to which prompts are specific to a particular model and to improve the transferability of prompts. Perez et al. (2021) found that prompts selected under a tuned few-shot learning scenario (where there is a larger validation set to choose prompts) generally perform well across models of similar sizes, while prompts selected under a true few-shot learning scenario (where there are only a few training samples) do not generalize as effectively to models of similar sizes. The transferability of prompts is poor when the model sizes are very different in both scenarios.

#### 2.2.10.8 Combination of Different Paradigms

Prompt-based learning has been successful due in part to the use of pre-trained models developed for the pre-train and fine-tune paradigm, such as BERT. However, it is not clear if the pre-training methods that are effective for the pre-train and fine-tune paradigm can be directly applied to prompt-based learning or if new pre-training methods are needed to further improve accuracy or ease of use for prompt-based learning. This is an important research question that has not been widely explored in the literature.

#### 2.2.10.9 Calibration of Prompting Methods

Calibration refers to a model's ability to make good probabilistic predictions. When using the generation probability of a pre-trained language model to predict an answer, it is important to consider that the probability distribution may not be well calibrated. Jiang et al. (2020b) found that the probabilities of pre-trained models on QA tasks are well calibrated, but Zhao et al. (2021) identified three pitfalls (majority label bias, recency bias, and common token bias) that can lead pre-trained language models to be biased towards certain answers when provided with answered prompts. To overcome these pitfalls, Zhao et al. (2021) used context-free input to get an initial probability distribution and real input to get a second probability distribution, which can be used to get a calibrated generation probability distribution. However, this method has the overhead of finding appropriate context-free input and the underlying pre-trained language model's probability distribution is still not calibrated. Even if the probability distribution is calibrated, it is important to be cautious when assuming a single gold answer for an input, as different surface forms of the same object will compete for finite probability mass (Holtzman et al., 2021). To address this issue, one could perform answer engineering to construct a comprehensive gold answer set using paraphrasing methods (§5.2.2) or calibrate the probability of a word based on its prior likelihood in given context (Holtzman et al., 2021).

## 2.3 Summary

There has been a lot work done in the field of Automatic Story Generation. But most of the works use the previous generation GPT2 as the pre-trained language model for story generation. There are very few works that utilize the power of the latest generation language models such as GPT3 or its alternatives.

Even though Prompt-Learning is a relatively new paradigm, there has been significant amount of research done in this field as well. Prompt-learning has been used for various NLP applications including text generation. But there is lack of works that use prompt-learning for narrative story generation tasks.

This work tries to fill the above gaps by using prompt-learning for story generation and contributing benchmark scores with the latest generation language models.

# CHAPTER 3: RESEARCH METHODOLOGY

## 3.1 Introduction

This work focuses on the text generation and few-shot learning capabilities of PLMs. Given an outline as a control-mechanism, the model should generate a story conditioned on the outline.

## 3.2 Algorithms & Techniques

3.2.3 State-of-the-Art Language Models

Some of the latest State-of-the-Art Pre-trained Language Models available for text generation tasks are discussed below:

* GPT3
* OPT
* BLOOM
* LaMDA
* PaLM

3.2.3 n-Shot Learning

* Zero-Shot Learning
* One-Shot Learning
* Few-Shot Learning

## 3.3 Methodology

### 3.3.1 End-to-End Pipeline

The end-to-end pipeline for the methodology is shown in Figure 3.3.1.1.

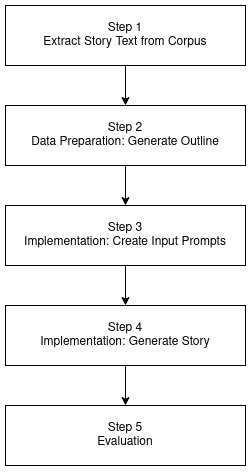


Figure 3.3.1.1 E2E Pipeline

3.3.2 Data Selection

This work makes use of two standard story generation datasets:

* **ROCStories**: Introduced by (Mostafazadeh et al., 2016) , this dataset contains ~98K 5-sentence long stories along with story titles. This dataset is widely used for short-form story generation tasks.
* **WritingPrompts**: Introduced by (Fan et al., 2018b) , this dataset contains ~300K human-written stories along with the starting prompt used to write the story. These stories were collected from the Reddit, an online social media forum. These stories are long-form multi-paragraph stories, and hence useful for more complex task of long-form story generation.

The datasets are used as a corpus for stories. This corresponds to Step 1 in Figure 3.3.1.1.

### 3.3.3 Data Preparation

The proposed method requires sample pairs of outline-instance to paragraph. While paragraph text can be derived from the ROCStories and WritingPrompts datasets, there is no dataset of outlines readily available. Hence, the outlines need to extracted from the story datasets and then mapped to corresponding paragraph text. These outline-paragraph pairs can then be sampled during the few-shot inference.

The outline instances can take one of two forms:

* **Summary** – Here the outline instance is a short extractive summary of the paragraph. The paragraph is expanded from the summary. For the summary extraction, TextRank (Mihalcea and Tarau, 2004) is proposed to be used to extract the most informative sentence from the paragraph.
* **Keywords/Keyphrases** - Here the outline instance is a set of keywords and phrases that are present in the paragraph. The paragraph text is generated conditioned on these keywords/keyphrases. For the outline extraction, RAKE (Rose et al., 2010) is proposed to be used to extract keyphrases from the paragraph.

This corresponds to Step 2 in Figure 3.3.1.1.

### 3.3.4 Implementation

The proposed implementation can be broadly separated into two major steps:

1. Create prompts for Few-Shot Learning – In this step, a dataset of few-shot sample pairs is created. Each sample pair consists of an outline (o) and corresponding text paragraph (t). The dataset takes the following form:

[(o1, t1), (o2, t2), (o3, t3), …, (on, tn)]

This corresponds to Step 3 in Figure 3.3.1.1.

1. Use the sample pairs as few-shot prompts to generate missing story paragraph for a new outline – The prompt, few-shot samples and the query outline are passed to the model as input for inference. The model returns the generated story paragraph corresponding to the query outline as prediction.

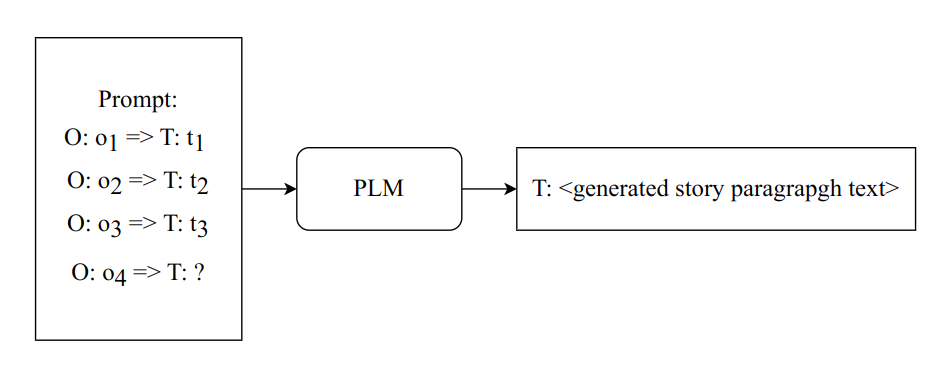


Figure 3.3.4.1 Prompting for Story Generation

This corresponds to Step 4 in Figure 3.3.1.1.

### 3.3.5 Evaluation

The generated stories are to be evaluated using multiple metrics. This work only focuses on evaluation using Automatic Metrics. Human-Evaluation of the generated stories is not within the scope of this work.

The proposed metrics for evaluation are:

* **Perplexity (PPL)** - Similar to (Fang et al., 2021; Jin et al., 2022) , PPL is used to compute word-level complexity.
* **DIST/distinct-n** (Li et al., 2015) - DIST measures generation diversity as a ratio of distinct n-grams to all generated n-grams.
* **BLEU** (Papineni et al., 2002) - Measures n-gram overlap between generated text and ground truth.
* **Self-BLEU** (Zhu et al., 2018) - Measures intra-story lexical diversity.
* **ROUGE** (Lin, 2004) - Includes Precision, Recall & F1, where ROUGE Precision has similar interpretation as BLEU score.

These metrics will be compared across different factors.

Based on length of generated story:

* Short-Form Story Generation – Stories with upto 5 sentences.
* Long-Form Story Generation – Stories with more than 5 sentences. This may include multiple paragraphs as well.

Based on Pre-trained Language Model used:

* GPT2
* GPT3
* BLOOM
* OPT

Based on the Inference type:

* Zero-shot Generation
* One-shot Generation
* Few-shot Generation

This work will be benchmarked against the following baselines:

* Outline-to-Story (**O2S**) (Fang et al., 2021)
* Summarize, Outline and Elaborate (**SOE**) (Sun et al., 2020)
* Prompt Transfer for Text Generation (**PTG**) (Li et al., 2022)

This corresponds to Step 5 in Figure 3.3.1.1.

## 3.3 Tools

## 3.5 Summary

# REFERENCES

Anon (2022) ROCStories and the Story Cloze Test. [online] Available at: https://cs.rochester.edu/nlp/rocstories/ [Accessed 26 Oct. 2022].

Bakhtin, A., Deng, Y., Ott, M., Ranzato, M.’ A. and Szlam, A., (2021) Residual Energy-Based Models for Text. Journal of Machine Learning Research, [online] 22, pp.1–41. Available at: http://jmlr.org/papers/v22/20-326.html. [Accessed 23 Oct. 2022].

Bisk, Y., Holtzman, A., Thomason, J., Andreas, J., Bengio, Y., Chai, J., Lapata, M., Lazaridou, A., May, J., Nisnevich, A., Pinto, N. and Turian, J., (2020) Experience Grounds Language. EMNLP 2020 - 2020 Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference, [online] pp.8718–8735. Available at: https://arxiv.org/abs/2004.10151v3 [Accessed 23 Oct. 2022].

Bosselut, A., Celikyilmaz, A., He, X., Gao, J., Huang, P. sen and Choi, Y., (2018) Discourse-Aware Neural Rewards for Coherent Text Generation. NAACL HLT 2018 - 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies - Proceedings of the Conference, [online] 1, pp.173–184. Available at: https://arxiv.org/abs/1805.03766v1 [Accessed 23 Oct. 2022].

Brown, T.B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D.M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., Mccandlish, S., Radford, A., Sutskever, I. and Amodei, D., (2020) Language Models are Few-Shot Learners. Advances in Neural Information Processing Systems, [online] 33, pp.1877–1901. Available at: https://commoncrawl.org/the-data/ [Accessed 23 Oct. 2022].

Chen, Z., Eavani, H., Chen, W., Liu, Y. and Wang, W.Y., (2019) Few-Shot NLG with Pre-Trained Language Model. [online] pp.183–190. Available at: https://arxiv.org/abs/1904.09521v3 [Accessed 25 Oct. 2022].

Clark, E., August, T., Serrano, S., Haduong, N., Gururangan, S. and Smith, N.A., (2021) All That’s ‘Human’ Is Not Gold: Evaluating Human Evaluation of Generated Text. ACL-IJCNLP 2021 - 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, Proceedings of the Conference, [online] pp.7282–7296. Available at: https://arxiv.org/abs/2107.00061v2 [Accessed 23 Oct. 2022].

Conneau, A. and Lample, G., (2019) Cross-lingual Language Model Pretraining. Advances in Neural Information Processing Systems, [online] 32. Available at: https://arxiv.org/abs/1901.07291v1 [Accessed 25 Oct. 2022].

Dathathri, S., Madotto, A., Lan, J., Hung, J., Ai, U., Frank, E., Molino, P., Yosinski, J. and Liu, R., (2019) Plug and Play Language Models: A Simple Approach to Controlled Text Generation. [online] Available at: https://arxiv.org/abs/1912.02164v4 [Accessed 25 Oct. 2022].

Devlin, J., Chang, M.W., Lee, K. and Toutanova, K., (2018) BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. NAACL HLT 2019 - 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies - Proceedings of the Conference, [online] 1, pp.4171–4186. Available at: https://arxiv.org/abs/1810.04805v2 [Accessed 23 Oct. 2022].

Dong, L., Yang, N., Wang, W., Wei, F., Liu, X., Wang, Y., Gao, J., Zhou, M. and Hon, H.-W., (2019) Unified Language Model Pre-training for Natural Language Understanding and Generation. Advances in Neural Information Processing Systems, [online] 32. Available at: https://github.com/microsoft/unilm. [Accessed 25 Oct. 2022].

Dou, Y., Forbes, M., Koncel-Kedziorski, R., Smith, N.A. and Choi, Y., (2021) Is GPT-3 Text Indistinguishable from Human Text? Scarecrow: A Framework for Scrutinizing Machine Text. [online] pp.7250–7274. Available at: https://arxiv.org/abs/2107.01294v3 [Accessed 23 Oct. 2022].

Dziri, N., Madotto, A., Zaiane, O. and Bose, A.J., (2021) Neural Path Hunter: Reducing Hallucination in Dialogue Systems via Path Grounding. EMNLP 2021 - 2021 Conference on Empirical Methods in Natural Language Processing, Proceedings, [online] pp.2197–2214. Available at: https://arxiv.org/abs/2104.08455v2 [Accessed 23 Oct. 2022].

Fan, A., Lewis, M. and Dauphin, Y., (2018a) Hierarchical Neural Story Generation. ACL 2018 - 56th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference (Long Papers), [online] 1, pp.889–898. Available at: https://arxiv.org/abs/1805.04833v1 [Accessed 23 Oct. 2022].

Fan, A., Lewis, M. and Dauphin, Y., (2018b) Hierarchical Neural Story Generation. ACL 2018 - 56th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference (Long Papers), [online] 1, pp.889–898. Available at: https://aclanthology.org/P18-1082 [Accessed 13 Sep. 2022].

Fan, A., Lewis, M. and Dauphin, Y., (2019) Strategies for Structuring Story Generation. ACL 2019 - 57th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference, [online] pp.2650–2660. Available at: https://arxiv.org/abs/1902.01109v2 [Accessed 23 Oct. 2022].

Fang, L., Li, C., Gao, J., Dong, W. and Chen, C., (2019) Implicit Deep Latent Variable Models for Text Generation. EMNLP-IJCNLP 2019 - 2019 Conference on Empirical Methods in Natural Language Processing and 9th International Joint Conference on Natural Language Processing, Proceedings of the Conference, [online] pp.3946–3956. Available at: https://arxiv.org/abs/1908.11527v3 [Accessed 25 Oct. 2022].

Fang, L., Zeng, T., Liu, C., Bo, L., Dong, W. and Chen, C., (2021) Outline to Story: Fine-grained Controllable Story Generation from Cascaded Events. [online] Available at: http://arxiv.org/abs/2101.00822 [Accessed 31 Aug. 2022].

Gao, J., Peng, B., Li, C., Li, J., Shayandeh, S., Liden, L. and Shum, H.-Y., (2020a) Robust Conversational AI with Grounded Text Generation. [online] Available at: https://arxiv.org/abs/2009.03457v1 [Accessed 23 Oct. 2022].

Gao, T., Fisch, A. and Chen, D., (2020b) Making Pre-trained Language Models Better Few-shot Learners. ACL-IJCNLP 2021 - 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, Proceedings of the Conference, [online] pp.3816–3830. Available at: https://arxiv.org/abs/2012.15723v2 [Accessed 25 Oct. 2022].

Goldfarb-Tarrant, S., Chakrabarty, T., Weischedel, R. and Peng, N., (2020) Content Planning for Neural Story Generation with Aristotelian Rescoring. EMNLP 2020 - 2020 Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference, [online] pp.4319–4338. Available at: https://arxiv.org/abs/2009.09870v2 [Accessed 23 Oct. 2022].

Guan, J., Huang, F., Zhao, Z., Zhu, X. and Huang, M., (2020) A knowledge-enhanced pretraining model for commonsense story generation. Transactions of the Association for Computational Linguistics, [online] 8, pp.93–108. Available at: https://direct.mit.edu/tacl/article/doi/10.1162/tacl\_a\_00302/43540/A-Knowledge-Enhanced-Pretraining-Model-for [Accessed 23 Oct. 2022].

Hu, Z., Yang, Z., Liang, X., Salakhutdinov, R. and Xing, E.P., (2017) Toward Controlled Generation of Text. Available at: https://proceedings.mlr.press/v70/hu17e.html [Accessed 25 Oct. 2022].

Hua, X. and Wang, L., (2020) PAIR: Planning and Iterative Refinement in Pre-trained Transformers for Long Text Generation. EMNLP 2020 - 2020 Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference, [online] pp.781–793. Available at: https://arxiv.org/abs/2010.02301v1 [Accessed 23 Oct. 2022].

Ji, H. and Huang, M., (2021) DiscoDVT: Generating Long Text with Discourse-Aware Discrete Variational Transformer. EMNLP 2021 - 2021 Conference on Empirical Methods in Natural Language Processing, Proceedings, [online] pp.4208–4224. Available at: https://arxiv.org/abs/2110.05999v1 [Accessed 23 Oct. 2022].

Ji, H., Ke, P., Huang, S., Wei, F., Zhu, X. and Huang, M., (2020) Language Generation with Multi-Hop Reasoning on Commonsense Knowledge Graph. EMNLP 2020 - 2020 Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference, [online] pp.725–736. Available at: https://arxiv.org/abs/2009.11692v1 [Accessed 23 Oct. 2022].

Jin, Y., Kadam, V. and Wanvarie, D., (2022) Plot Writing From Pre-Trained Language Models. INLG 2022. [online] Available at: https://arxiv.org/abs/2206.03021 [Accessed 24 Aug. 2022].

Keskar, N.S., Mccann, B., Varshney, L.R., Xiong, C., Socher, R. and Research, S., (2019) CTRL: A Conditional Transformer Language Model for Controllable Generation. [online] Available at: https://arxiv.org/abs/1909.05858v2 [Accessed 23 Oct. 2022].

Li, J., Galley, M., Brockett, C., Gao, J. and Dolan, B., (2015) A Diversity-Promoting Objective Function for Neural Conversation Models. 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL HLT 2016 - Proceedings of the Conference, [online] pp.110–119. Available at: https://arxiv.org/abs/1510.03055v3 [Accessed 29 Oct. 2022].

Li, J., Tang, T., Nie, J.-Y., Wen, J.-R. and Zhao, W.X., (2022) Learning to Transfer Prompts for Text Generation. NAACL 2022, [online] pp.3506–3518. Available at: https://arxiv.org/abs/2205.01543v2 [Accessed 11 Oct. 2022].

Li, J., Tang, T., Zhao, W.X., Wei, Z., Yuan, N.J. and Wen, J.R., (2021) Few-shot Knowledge Graph-to-Text Generation with Pretrained Language Models. Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, [online] pp.1558–1568. Available at: https://arxiv.org/abs/2106.01623v1 [Accessed 25 Oct. 2022].

Li, X.L. and Liang, P., (2021) Prefix-Tuning: Optimizing Continuous Prompts for Generation. ACL-IJCNLP 2021 - 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, Proceedings of the Conference, [online] pp.4582–4597. Available at: https://arxiv.org/abs/2101.00190v1 [Accessed 25 Oct. 2022].

Lin, C.-Y., (2004) ROUGE: A Package for Automatic Evaluation of Summaries. Available at: https://aclanthology.org/W04-1013 [Accessed 29 Oct. 2022].

Lin, Z. and Riedl, M.O., (2021) Plug-and-Blend: A Framework for Controllable Story Generation with Blended Control Codes. NAACL | NUSE | 2021, [online] pp.62–71. Available at: https://aclanthology.org/2021.nuse-1.7 [Accessed 31 Aug. 2022].

Liu, X., Zheng, Y., Du, Z., Ding, M., Qian, Y., Yang, Z. and Tang, J., (2021) GPT Understands, Too. [online] Available at: https://arxiv.org/abs/2103.10385v1 [Accessed 25 Oct. 2022].

Mao, H.H., Majumder, B.P., McAuley, J. and Cottrell, G.W., (2019) Improving Neural Story Generation by Targeted Common Sense Grounding. EMNLP-IJCNLP 2019 - 2019 Conference on Empirical Methods in Natural Language Processing and 9th International Joint Conference on Natural Language Processing, Proceedings of the Conference, [online] pp.5988–5993. Available at: https://aclanthology.org/D19-1615 [Accessed 31 Aug. 2022].

Mihalcea, R. and Tarau, P., (2004) TextRank: Bringing Order into Texts.

Mori, Y., Yamane, H., Shimizu, R. and Harada, T., (2022) Plug-and-Play Controller for Story Completion: A Pilot Study toward Emotion-aware Story Writing Assistance. [online] pp.46–57. Available at: https://aclanthology.org/2022.in2writing-1.6 [Accessed 1 Oct. 2022].

Mostafazadeh, N., Chambers, N., He, X., Parikh, D., Batra, D., Vanderwende, L., Kohli, P. and Allen, J., (2016) A Corpus and Cloze Evaluation for Deeper Understanding of Commonsense Stories. 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL HLT 2016 - Proceedings of the Conference, [online] pp.839–849. Available at: https://aclanthology.org/N16-1098 [Accessed 13 Sep. 2022].

Papineni, K., Roukos, S., Ward, T. and Zhu, W.-J., (2002) Bleu: a Method for Automatic Evaluation of Machine Translation. Proceedings of the 40th Annual Meeting on Association for Computational Linguistics - ACL ’02, [online] pp.311–318. Available at: https://aclanthology.org/P02-1040 [Accessed 29 Oct. 2022].

Pascual, D., Egressy, B., Bolli, F. and Wattenhofer, R., (2020) Directed Beam Search: Plug-and-Play Lexically Constrained Language Generation. [online] Available at: http://arxiv.org/abs/2012.15416 [Accessed 31 Aug. 2022].

Pascual, D., Egressy, B., Meister, C., Cotterell, R. and Wattenhofer, R., (2021) A Plug-and-Play Method for Controlled Text Generation. Findings of the Association for Computational Linguistics, Findings of ACL: EMNLP 2021, [online] pp.3973–3997. Available at: https://aclanthology.org/2021.findings-emnlp.334 [Accessed 31 Aug. 2022].

Peng, N., Ghazvininejad, M., May, J. and Knight, K., (2018) Towards Controllable Story Generation. [online] pp.43–49. Available at: https://aclanthology.org/W18-1505 [Accessed 25 Oct. 2022].

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D. and Sutskever, I., (2019) Language Models are Unsupervised Multitask Learners. [online] Available at: https://github.com/codelucas/newspaper [Accessed 23 Oct. 2022].

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W. and Liu, P.J., (2020) Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. Journal of Machine Learning Research, [online] 21, pp.1–67. Available at: http://jmlr.org/papers/v21/20-074.html. [Accessed 25 Oct. 2022].

Rashkin, H., Celikyilmaz, A., Choi, Y. and Gao, J., (2020) PlotMachines: Outline-Conditioned Generation with Dynamic Plot State Tracking. EMNLP 2020 - 2020 Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference, [online] pp.4274–4295. Available at: https://aclanthology.org/2020.emnlp-main.349 [Accessed 31 Aug. 2022].

Rashkin, H., Smith, E.M., Li, M. and Boureau, Y.L., (2018) Towards Empathetic Open-domain Conversation Models: a New Benchmark and Dataset. ACL 2019 - 57th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference, [online] pp.5370–5381. Available at: https://arxiv.org/abs/1811.00207v5 [Accessed 23 Oct. 2022].

Rose, S., Engel, D., Cramer, N. and Cowley, W., (2010) Automatic Keyword Extraction from Individual Documents. Text Mining: Applications and Theory, [online] pp.1–20. Available at: https://onlinelibrary.wiley.com/doi/full/10.1002/9780470689646.ch1 [Accessed 29 Oct. 2022].

le Scao, T. and Rush, A.M., (2021) How Many Data Points is a Prompt Worth? NAACL-HLT 2021 - 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Proceedings of the Conference, [online] pp.2627–2636. Available at: https://arxiv.org/abs/2103.08493v2 [Accessed 25 Oct. 2022].

See, A., Pappu, A., Saxena, R., Yerukola, A. and Manning, C.D., (2019) Do Massively Pretrained Language Models Make Better Storytellers? CoNLL 2019 - 23rd Conference on Computational Natural Language Learning, Proceedings of the Conference, [online] pp.843–861. Available at: https://arxiv.org/abs/1909.10705v1 [Accessed 23 Oct. 2022].

Shakeri, H., Neustaedter, C. and DiPaola, S., (2021) SAGA: Collaborative Storytelling with GPT-3; SAGA: Collaborative Storytelling with GPT-3. Companion Publication of the 2021 Conference on Computer Supported Cooperative Work and Social Computing. [online] Available at: https://doi.org/10.1145/3462204.3481771 [Accessed 1 Oct. 2022].

Shen, T., Lei, T., Barzilay, R., Jaakkola, T. and Csail, M., (2017) Style Transfer from Non-Parallel Text by Cross-Alignment. Advances in Neural Information Processing Systems, [online] 30. Available at: https://github.com/shentianxiao/language-style-transfer. [Accessed 25 Oct. 2022].

Shin, T., Razeghi, Y., Logan, R.L., Wallace, E. and Singh, S., (2020) AutoPrompt: Eliciting Knowledge from Language Models with Automatically Generated Prompts. EMNLP 2020 - 2020 Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference, [online] pp.4222–4235. Available at: https://arxiv.org/abs/2010.15980v2 [Accessed 25 Oct. 2022].

Song, K., Tan, X., Qin, T., Lu, J. and Liu, T.Y., (2019) MASS: Masked Sequence to Sequence Pre-training for Language Generation. 36th International Conference on Machine Learning, ICML 2019, [online] 2019-June, pp.10384–10394. Available at: https://arxiv.org/abs/1905.02450v5 [Accessed 25 Oct. 2022].

Su, Y., Wang, X., Qin, Y., Chan, C.-M., Lin, Y., Wang, H., Wen, K., Liu, Z., Li, P., Li, J., Hou, L., Sun, M. and Zhou, J., (2021) On Transferability of Prompt Tuning for Natural Language Processing. NAACL 2022 - 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Proceedings of the Conference, [online] pp.3949–3969. Available at: http://arxiv.org/abs/2111.06719 [Accessed 25 Oct. 2022].

Sun, X., Fan, C., Sun, Z., Meng, Y., Wu, F. and Li, J., (2020) Summarize, Outline, and Elaborate: Long-Text Generation via Hierarchical Supervision from Extractive Summaries. [online] Available at: https://arxiv.org/abs/2010.07074v2 [Accessed 23 Oct. 2022].

Tan, B., Yang, Z., Al-Shedivat, M., Xing, E.P. and Hu, Z., (2020) Progressive Generation of Long Text with Pretrained Language Models. NAACL-HLT 2021 - 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Proceedings of the Conference, [online] pp.4313–4324. Available at: https://arxiv.org/abs/2006.15720v2 [Accessed 23 Oct. 2022].

Vaswani, A., Brain, G., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł. and Polosukhin, I., (2017) Attention is All you Need. Advances in Neural Information Processing Systems, 30.

Vu, T., Lester, B., Constant, N., Al-Rfou, R., Cer, D. and Research, G., (2021) SPoT: Better Frozen Model Adaptation through Soft Prompt Transfer. [online] pp.5039–5059. Available at: https://arxiv.org/abs/2110.07904v2 [Accessed 25 Oct. 2022].

Xu, J., Ren, X., Zhang, Y., Zeng, Q., Cai, X. and Sun, X., (2018) A Skeleton-Based Model for Promoting Coherence Among Sentences in Narrative Story Generation. Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, EMNLP 2018, [online] pp.4306–4315. Available at: https://arxiv.org/abs/1808.06945v2 [Accessed 23 Oct. 2022].

Xu, P., Patwary, M., Shoeybi, M., Puri, R., Fung, P., Anandkumar, A. and Catanzaro, B., (2020) MEGATRON-CNTRL: Controllable Story Generation with External Knowledge Using Large-Scale Language Models. EMNLP 2020 - 2020 Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference, [online] pp.2831–2845. Available at: https://arxiv.org/abs/2010.00840v1 [Accessed 23 Oct. 2022].

Yao, L., Peng, N., Weischedel, R., Knight, K., Zhao, D. and Yan, R., (2019) Plan-and-Write: Towards Better Automatic Storytelling. Proceedings of the AAAI Conference on Artificial Intelligence, [online] 3301, pp.7378–7385. Available at: https://ojs.aaai.org/index.php/AAAI/article/view/4726 [Accessed 23 Oct. 2022].

Zellers, R., Holtzman, A., Rashkin, H., Bisk, Y., Farhadi, A., Roesner, F., Choi, Y. and Allen, P.G., (2019) Defending Against Neural Fake News. Advances in Neural Information Processing Systems, [online] 32. Available at: https://rowanzellers.com/grover [Accessed 23 Oct. 2022].

Zhang, Y., Sun, S., Galley, M., Chen, Y.-C., Brockett, C., Gao, X., Gao, J., Liu, J. and Dolan, B., (2019) DialoGPT: Large-Scale Generative Pre-training for Conversational Response Generation. [online] pp.270–278. Available at: https://arxiv.org/abs/1911.00536v3 [Accessed 23 Oct. 2022].

Zhao, J.J., Kim, Y., Zhang, K., Rush, A.M. and Lecun, Y., (2018) Adversarially Regularized Autoencoders. [online] pp.5902–5911. Available at: https://proceedings.mlr.press/v80/zhao18b.html [Accessed 25 Oct. 2022].

Zhu, Y., Lu, S., Zheng, L., Guo, J., Zhang, W., Wang, J. and Yu, Y., (2018) Texygen: A benchmarking platform for text generation models. 41st International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2018, pp.1097–1100.

Ziegler, Z.M., Melas-Kyriazi, L., Gehrmann, S. and Rush, A.M., (2019) Encoder-Agnostic Adaptation for Conditional Language Generation. [online] Available at: https://arxiv.org/abs/1908.06938v2 [Accessed 25 Oct. 2022].

Zou, X., Yin, D., Zhong, Q., Yang, H., Yang, Z. and Tang, J., (2021) Controllable Generation from Pre-trained Language Models via Inverse Prompting. Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp.2450–2460.

# APPENDIX A: RESEARCH PLAN

# APPENDIX B: RESEARCH PROPOSAL

# APPENDIX C: ETHICS FORMS