Enhanced Story Comprehension for Large Language Models through Dynamic Document-Based Knowledge Graphs

Berkeley Andrus, Yeganeh Nasiri, Jay Cui, Ben Cullen, and Nancy Fulda
Computer Science Department
Brigham Young University
Provo, Utah, USA
[bandrus5, ynasiri, shilong, bcullen2, nfulda]@byu.edu

Abstract

Large transformer-based language models have achieved incredible success at various tasks which require narrative comprehension, including story completion, answering questions about stories, and generating stories *ex nihilo*. However, due to the limitations of finite context windows, these language models struggle to produce or understand stories longer than several thousand tokens. In order to mitigate the document length limitations that come with finite context windows, we introduce a novel architecture that augments story processing with an external dynamic knowledge graph. In contrast to static commonsense knowledge graphs which hold information about the real world, these dynamic knowledge graphs reflect facts extracted from the story being processed. Our architecture uses these knowledge graphs to create information-rich prompts which better facilitate story comprehension than prompts composed only of story text. We apply our architecture to the tasks of question answering and story completion. To complement this line of research, we introduce two long-form question answering tasks, LF-SQuAD and LF-QUOREF, in which the document length exceeds the size of the language model's context window, and introduce a story completion evaluation method that bypasses the stochastic nature of language model generation. We demonstrate broad improvement over typical prompt formulation methods for both question answering and story completion using GPT-2, GPT-3 and XLNet.

1 Introduction

Large language models such as GPT-2 (Radford et al. 2019) have been used for a variety of story-related tasks including story completion (Xu et al. 2020; Guan et al. 2020), reading comprehension (Radford et al. 2019; Raffel et al. 2020), and story generation (Fan, Lewis, and Dauphin 2018; Roemmele 2016). However, previous works have mostly focused on extremely short stories that fit within the limited context windows of transformer-based language models. For example, several of these works use the popular ROCStories dataset (Mostafazadeh et al. 2016) which is composed of five-sentence stories. Using GPT- 2's tokenization rules, the stories in this dataset have an average token length of 53.5, much smaller than most transformer context windows. While language models with finite context windows are capable of generating stories of arbitrary length through iterative text predictions, these longer stories are prone to introduce information that abandons or contradicts information present earlier in the story. Much work has been done to increase story coherence through external knowledge bases (Xu et al. 2020; Guan et al. 2020), but these works have focused on making stories consistent with the real world rather than on making stories consistent with themselves.

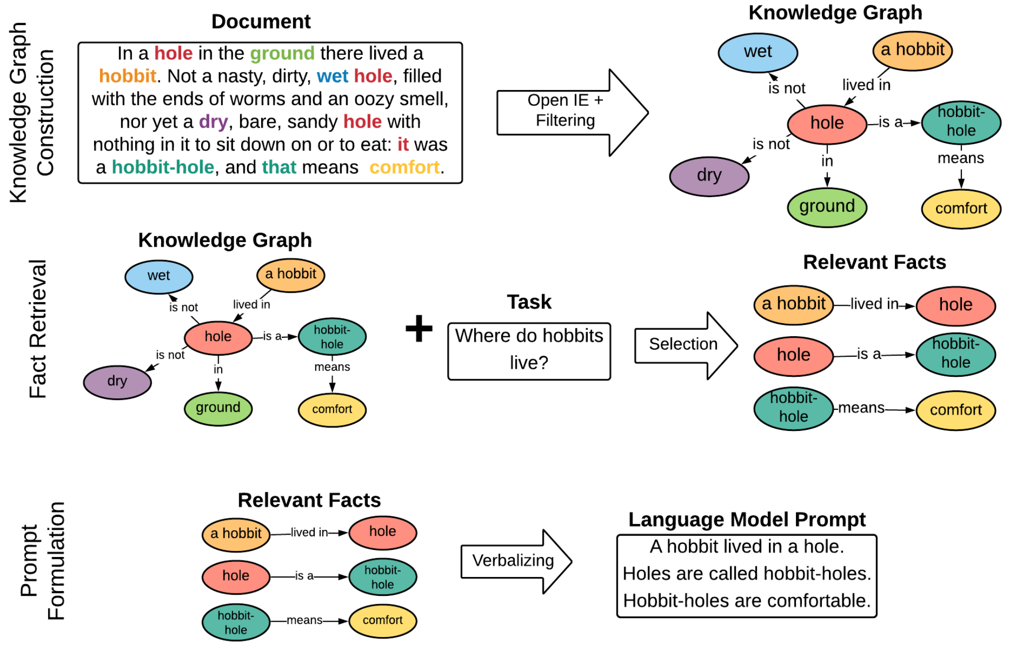
A language model's ability to generate internally consistent text relies on a property which we term "story comprehension". This ability is a prerequisite to successfully completing various tasks including story completion, story generation, document summarization, question answering, etc. It stands to reason that language models with finite context windows, including transformer-based language models, cannot comprehend more text than fits within their context windows, putting an upper limit on the story comprehension ability of these models. This poses a problem for long-form text generation applications such as *AI Dungeon*[1] in which end users have a vested interest in the coherency of generated stories.

In this work we introduce a novel architecture for improving the story comprehension of large language models through external knowledge bases. Our approach involves extracting facts from a document and constructing a custom dynamic knowledge graph. Then, given a story comprehension task, we extract and verbalize relevant information from the knowledge graph and incorporate it into information-rich prompts for a language model.

The primary contributions of this work include:

- Defining an architecture which interfaces effectively with a large language model, providing the language model with fact-rich prompts that enhance story comprehension
- Introducing LF-SQuAD and LF-QUOREF, two novel evaluation tasks designed to measure long-form story comprehension
- Introducing a new evaluation metric for story completion that, unlike previous metrics such as BLEU, does not assume that a human-written response is the single correct answer

[1]https://play.aidungeon.io



Knowledge Graph Construction
Document
In a hole in the ground there lived a hobbit. Not a nasty, dirty, wet hole, filled with the ends of worms and an oozy smell, nor yet a dry, bare, sandy hole with nothing in it to sit down on or to eat: it was a hobbit-hole, and that means comfort.
Open IE + Filtering
Knowledge Graph
wet
a hobbit
is not
lived in
hole
is a
hobbit-hole
is not
in
means
dry
ground
comfort
Fact Retrieval
Task
Where do hobbits live?
Selection
Relevant Facts
Prompt Formulation
Verbalizing
Language Model Prompt
A hobbit lived in a hole.
Holes are called hobbit-holes.
Hobbit-holes are comfortable.

Figure 1: Visualization of the three tasks accomplished by our architecture. In Knowledge Graph Construction, we use an Open Information Extraction model with rule based post processing to convert a long-form text document into a document-specific knowledge graph. In Fact Retrieval we find which facts are most relevant to some story comprehension task. In Prompt Formulation we use few-shot learning with GPT-3 to verbalize extracted facts as natural language and incorporate them into a prompt for a large language model.

We apply our architecture to two story comprehension tasks, question answering and story completion, and evaluate its impact on the performance of three transformer-based language models, GPT-2 (Radford et al. 2019), GPT-3 (Brown et al. 2020), and XLNet (Yang et al. 2019). Our evaluation tasks require a language model to process and understand text that exceeds its context window size, and hence are not directly comparable to existing tasks and methodologies which rely on documents with limited lengths. We demonstrate consistent improvement over a system in which no knowledge base is used and language models must make predictions using traditional prompts.

2 Related Work

There have been several previous works exploring how knowledge graphs can augment language models. Guan et al. (2020) use commonsense knowledge graphs to generate data for fine-tuning GPT-2 to enhance the language model's ability to generate coherent, non-repetitive stories. Xu et al. (2020) also use a knowledge graph to fine-tune language models for story generation, adding a mechanism to extract relevant information from the knowledge graph based on predicted keywords as it was needed. Both works use static pre-constructed knowledge graphs.

Guu et al. (2020) use an external knowledge base to train a language model on open-domain question answering. It uses a collection of text snippets from Wikipedia as a knowledge base rather than a knowledge graph, and it trains a a neural knowledge retriever rather than retrieving based on predicted keywords.

In contrast to these works, our system uses a knowledge base that is dynamic and reflects facts that have appeared in the story rather than commonsense information. Our focus is not on helping the language model comprehend the real world but on helping the language model comprehend the document being processed. In addition, the documents we use to evaluate our system are many times longer than documents used in directly comparable systems, shifting our tasks to a new and largely unexplored domain.

Dynamic knowledge graphs have been used in Dialogue State Tracking applications to model participants in a conversation (He et al. 2017; Zhou and Small 2019). Bosselut, Bras, and Choi (2019) and Bosselut et al. (2019) also aim to generate knowledge graphs dynamically, but treat knowledge graph generation as a language modelling problem for discovering *implicit* commonsense relations between entities, rather than an extraction problem for recording *explicit* facts specific to document text.

There has also been much work over the years to automatically generate knowledge graphs from text documents. Wang et al. (2018) use a statistical model to predict entity relations from filtered domain-specific text. Distiawan et al. (2019) train a relation extraction model to add new relations between entities in an existing knowledge graph. Angeli, Premkumar, and Manning (2015) use dependency parses to extract tuples including entities and relations. Nayak and Ng (2020) focus on extracting overlapping relations, where entities have multiple relations reflected in a single text span. All of these works aim to extract knowledge about the world from the text, creating one large database that applies generally or that always describes a specific domain. Our system, conversely, aims to create a knowledge base that describes a single long-form document, including subjective facts that may not be meaningful or even intelligible out of context. Particularly relevant to this paradigm of document-specific knowledge graphs are (Ammanabrolu and Riedl 2019; Ammanabrolu et al. 2020; Adhikari et al. 2020), which use knowledge graphs to track agent progress and world state in text adventure games. This work uses similar mechanisms for generating knowledge graphs from text, but we use knowledge graphs to generate language model prompts rather than to inform reinforcement learning agents.

Finally, there are many approaches to story related tasks that do not rely on prevailing transformer-based language models. These include the introduction of new variants of the transformer architecture (Ainslie et al. 2020), hierarchical representation and generation of stories (Li, Luong, and Jurafsky 2015; Fan, Lewis, and Dauphin 2018), and embedding facts with a neural network for question answering (Kumar et al. 2016; Xiong, Merity, and Socher 2016). We emphasize that our purpose is not to compete with these other methods but to maximize the performance of existing transformer-based language models on similar tasks in a way that does not require any training, fine-tuning, or labelled data.

Algorithm 1: Story Comprehension Pipeline

Inputs
D = natural language document to be processed
E = schemaless knowledge extractor (Open IE)
LM = text generation via language model (GPT-2, GPT-3, XLNet)
ψ = filter function for KG facts
Φ = text similarity metric (uses either SBERT or Levenshtein)
Υ = fact verbalization function
k = language model context length
$query$ = the question to be answered, or the final paragraph of the story to be completed, $query \not\subset D$
$framing\_text$ = Additional task-specific text inserted into prompt to focus the language model

Pipeline

```
 1: $G$ = []
 2: for $sentence$ in $D$ do
 3:     $T = E(sentence)$
 4:     $T$ = [t for t in $T$ if $\psi(t)$]
 5:     $G = G + T$
 6: end for
 7: $similarities$ = []
 8: for $f$ in $G$ do
 9:     $similarities$.append($\Phi$ ($f$, $query$))
10: end for
11: $indices$ = argmax($similarities$,n=3)
12: $facts = G[indices]$
13: $verbalized\_facts$ = [$\Upsilon(f)$ for $f$ in $facts$]
14: $k_D = k$ - len($framing\_text + verbalized\_facts + query$)
15: $prompt = D[-k_D:] + verbalized\_facts + framing\_text + query$
16: $output\_text = LM(prompt)$
```

3 Architecture

We use a dynamic knowledge graph to enhance the coherence of language model output when making document-based predictions as shown in Figure 1.

Our story comprehension pipeline consists of three steps: (1) *Knowledge Graph Construction*, in which we construct a knowledge graph G that contains key information extracted from a natural language document D, (2) *Fact Retrieval* from the knowledge graph, in which we retrieve from G facts that are relevant to some document comprehension task T, and (3) *Prompt Formulation*, which includes both the synthesis of knowledge graph facts into free-form text and the concatenation of those facts with an excerpt from the story text and some framing text to form a prompt P. Further details can be found in Algorithm 1. Our complete architecture implementation is available in the supplementary materials for this paper.

3.1 Knowledge Graph Construction

The first task for our architecture is to generate a knowledge graph G that contains information extracted from a text document D containing a story or narrative text. Once constructed, G is composed of facts of the form $[head, relation, tail]$, where $head$ and $tail$ are entities from the document and $relation$ describes how those entities are related. We construct G by, for each sentence $s \in D$, extracting 0 or more facts from s and adding them to G. We extract facts using the Stanford Open Information Extraction (Open IE) model (Angeli, Premkumar, and Manning 2015), which separates natural language sentences into short clauses and then segments clauses into a knowledge graph triple.

Many applications that use knowledge graphs for natural language tasks use a schema, such as the knowledge graphs Wikidata (Vrandečić and Krötzsch 2014) and ConceptNet (Liu and Singh 2004). Schemas often predefine a set of possible relations between entities, which facilitate automatic data analysis or text generation. In our application facts in G will only ever be processed by a language model, which eliminates the need for a predefined schema. We choose the more flexible approach of using a schemaless knowledge graph where the $head$, $relation$, and $tail$ of each fact can be composed of any natural language found in the text. $relation$ is typically a verb phrase and $head$ is typically a noun phrase. See Table 3 for examples of extracted knowledge graph facts.

To mitigate any mistakes made by the Open IE model, we apply several filtering rules before adding each fact to G. The purpose of these rules is to eliminate any facts that lack a recoverable meaning. We do not add a fact of the form $[head, relation, tail]$ to G if it follows any of the following patterns:

- $tail$ is identical to $head$ (e.g. [you, just sit, you])
- $head$ does not contain a noun (e.g. [screamed, howling like, wolf])

- the first word in $tail$ is identical to the last word in $relation$ (e.g. [i, enjoy playing, playing soccer])
- $head$ or $tail$ starts with a pronoun or conjunction (e.g. [who, lurking, behind a wire], [but, should, be careful])
- there are no verbs present in the fact (e.g. [breakfast, lunch, dinner])

We use the NLTK interface for WordNet (Miller 1995) to determine the part-of-speech of words in each fact.

3.2 Fact Retrieval from Knowledge Graph

Given a story comprehension task T and a generated knowledge graph G, our next step in producing a suitable prompt P is to find which facts in G are most relevant to T. We use two relevance metrics, one for question answering and one for story completion. For question answering, where we are interested in finding discrete information about specific story entities, we determine relevance using the string edit distance between knowledge graph facts $f \in G$ and the question T. The most relevant fact for question answering can be defined as

$$\arg\min_{f \in G} lev(f, T) \tag{1}$$

where lev represents the Levenshtein distance function.

For story completion we are more concerned with semantic similarity than literal string similarity, so we determine relevance using Sentence-BERT (SBERT) (Reimers and Gurevych 2019), a variation on the BERT network (Devlin et al. 2018) designed to generate embeddings which facilitate semantic comparisons. We find the cosine similarity between the SBERT embeddings of T (in this case the most recent paragraph of the story) and each fact $f \in G$. The most relevant fact for story completion can be defined as

$$\arg\max_{f \in G} \cos(SB(f), SB(T)) \tag{2}$$

where $\cos$ represents the cosine similarity function and SB is the application of the SBERT embedding model.

We also try applying SBERT for the question answering task and Levenshtein distance for the story completion task, but find that these settings yield inferior performance.

For both question answering and story completion we select the top 3 most relevant facts from G. We also experiment with larger numbers but find a negligible change in overall system performance.

3.3 Prompt Formulation

When a knowledge graph schema is used, facts can be converted to sentences using template rules (Xu et al. 2020; Guan et al. 2020). Because we have a schemaless knowledge graph with open-ended relations, we instead use few-shot learning with GPT-3 (Brown et al. 2020) to verbalize knowledge graph facts as sentences that can be incorporated into language model prompts. We use the prompt shown in Figure 2 to coax GPT-3 into generating well-formed sentences.

We note that while GPT-3 is not currently publicly available, other methods of verbalizing knowledge graph facts exist. For example, Agarwal et al. (2021) demonstrate success fine-tuning T5 (Raffel et al. 2020) for knowledge graph verbalization, and in many cases even simple concatenation of fact constituents produces adequate verbalizations.

She , no went to , the market. : She didn't go to the market.
Rowling , is author of, Harry Potter. : Rowling is the author of Harry Potter.
SLC airport, located , Salt Lake City : SLC airport is located in Salt Lake City.
knight, fought with, dragon : The knight fought with the dragon.
[Fact Head], [Fact Relation], [Fact Tail] **:**

Figure 2: Prompt used to verbalize knowledge graph facts through few-shot learning with GPT-3. [Fact Head], [Fact Relation], and [Fact Tail] are replaced with the three constituents of the knowledge graph fact being verbalized. Several example inputs and outputs are shown in Table 3.

4 Experiments

To test whether our system improves the performance of large language models, we evaluate it using two tasks that require document comprehension: question answering and story completion. We hypothesize that our dynamic knowledge graph will aid the language model in generating more coherent text by capturing facts that would not normally fit within the language model's context window if the language model were working merely on unfiltered document text.

We repeat experiments using three language models: GPT-2, GPT-3, and XLNet. These models are selected due to their prevalence as generative language models and their high performance on other document comprehension tasks (Radford et al. 2019; Brown et al. 2020; Yang et al. 2019). Other large language models such as BERT (Devlin et al. 2018) and T5 (Raffel et al. 2020) are not trained to perform autoregressive text generation and are thus not natural fits for both the question answering and story completion tasks.

4.1 Question Answering

For the question answering task our goal is to correctly answer questions about documents. The task has been referred to as 'Reading Comprehension' in other works and is distinct from the open-domain question answering task, in which no document containing the correct answer is provided.

We evaluate our system's performance on two new question answering datasets, LF-SQuAD and LF-QUOREF (with LF standing for long-form), which we make available in the supplementary materials. These datasets are adapted from SQuAD (Rajpurkar et al. 2016) and QUOREF (Dasigi et al. 2019) respectively. The primary difference in these new datasets is that rather than associating questions with *paragraphs* we associate them with *documents*. For each question in SQuAD and QUOREF, we reconstruct the original source document and pair the question with the entire document rather than with a specific paragraph. This substantially increases the difficulty of the question answering task, particularly for transformer-based language models that can fit paragraphs in their context windows but not entire doc-

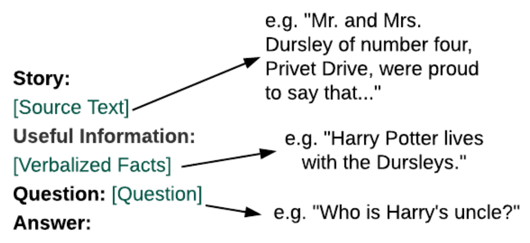

Story:
[Source Text]
e.g. "Mr. and Mrs. Dursley of number four, Privet Drive, were proud to say that..."
Useful Information:
[Verbalized Facts]
e.g. "Harry Potter lives with the Dursleys."
Question: [Question]
Answer:
e.g. "Who is Harry's uncle?"

Figure 3: Prompt used for the Question Answering task. [Source Text] is replaced with as much of the end of the document text as will fit in the context window. [Verbalized Facts] is replaced with the most relevant facts from the knowledge graph verbalized as sentences. [Question] is replaced with the question from the dataset. The entire prompt is limited to a 1000 token context length. In the baseline approach, the [Verbalized Facts] field is omitted, which makes room for more document text in the [Source Text] field.

uments. In constructing LF-SQuAD and LF-QUOREF we also filter out any questions that have multiple correct answers listed to simplify the evaluation process and avoid situations in which answers could be expressed multiple ways. In these experiments we use an abridged version of each dataset to accommodate our compute-heavy task. We first choose the document-question pairs associated with the longest documents and then randomly select 600 document-question pairs from LF-SQuAD and 568 document-question pairs from LF-QUOREF. We use the same subset of the datasets in all experiments.

We answer questions using the process defined in Algorithm 1 and the language models listed above. We compare each model's answer to the human labelled answer using an F1 score, which measures the amount of overlap in tokens used in each answer as in (Rajpurkar et al. 2016). Specifically, we calculate F1 by finding the precision p (percentage of tokens in model answer also found in human answer) and recall r (percentage of tokens in human answer also found in model answer) and defining F1 as $100 \times 2pr/(p+r)$.

We compare our results to a more traditional approach in which no facts are provided in the prompt and the same language models are used. This approach typically provides more of the actual document text to the language model, as the maximum prompt length remains the same. We note that it is similar to the question answering approach used in (Radford et al. 2019), except that we do not prime the model with multiple question-answer pairs in order to fit more story text. We were unable to identify any other baseline approaches that could be meaningfully applied to the task of answering questions about long form documents via a pretrained language model.

4.2 Story Completion

We make a distinction between story *completion*, in which a system reads part of a human-written story and composes a plausible end or next portion of the story, and story *generation*, in which a system composes a story *ex nihilo*. The mechanics of the two tasks are similar and we believe our methods are applicable to both, but in this work we evaluate only on story completion due to a lack of satisfactory automated metrics for story generation.

We evaluate system performance using a dataset of Sparknotes summaries of popular novels[2]. We use summaries rather than stories because they contain highly factual and detail-rich sentences without extra prose or dialogue, making them well suited for comprehension tasks. Although the summaries are shorter than the corresponding stories, they are still considerably longer than the stories used in comparable works of which we are aware. For example, the commonly used ROCStories dataset (Mostafazadeh et al. 2016) contains an average of 53.5 tokens per story using GPT-2's tokenizer, while the Sparknotes summaries contain an average of 1166.3 tokens per story. For reference, the maximum total length of a combined prompt and response is 1024 tokens with GPT-2 and 2048 tokens with GPT-3. In these experiments we use a prompt length of 800 to allow sufficient space for the model's response.

We evaluate by processing the first paragraph of a human-written story and using it to predict the text of the second paragraph, then processing the first two paragraphs and using them to predict the text of the third paragraph, etc. until every paragraph has been predicted based on the processed preceding paragraphs. At each step we construct a knowledge graph from all of the story paragraphs seen so far and select knowledge graph facts based on what is relevant to the most recently processed paragraph. We generate a prompt consisting of document text with the verbalized knowledge graph facts inserted before the last story paragraph. The prompt is limited to a maximum of 800 tokens, and the amount of document text is scaled appropriately. As with question answering, we compare our results to a traditional approach in which no facts are provided in the prompt, which typically causes more document text to be provided.

We use two automatic methods to evaluate the efficacy of our enhanced prompts. Given a language model L, a prompt p, and a human-written completion c, the first evaluation method is to generate a new completion $\hat{c} = \text{generate}(L, p)$ and measure the BLEU score between c and $\hat{c}$. This method is currently the typical evaluation method for story completion, but it relies on the faulty assumption that c is the "correct answer" and that increased n-gram overlap between c and $\hat{c}$ correlates with generation quality. In reality, there are many different ways to appropriately complete any given story portion, and most will have little n-gram overlap with c. Additionally, $\hat{c}$ is stochastically generated by the language model, making it difficult to replicate BLEU results. To overcome these issues with BLEU evaluation, we introduce a second story-completion evaluation metric based on per-

[2]We use the summaries for the 606 novels found at https://www.sparknotes.com/lit/ at the time of writing. For each novel we collect the text at https://www.sparknotes.com/lit/[story ID]/summary/. In total we make predictions on over 4000 paragraphs of text, and we make all summaries available with the supplementary materials.

LM	Method	LF-S F1	LF-Q F1
GPT-2	Traditional Prompts	8.1	24.0
	Our Pipeline	**18.6***	**25.4**
GPT-3	Traditional Prompts	15.7	22.4
	Our Pipeline	**20.0***	**25.7**
XLNet	Traditional Prompts	6.6	14.1
	Our Pipeline	**12.9***	**18.5***

Table 1: Results of question answering experiments described in Section 4.1. LF-S and LF-Q are LF-SQuAD and LF-QUOREF respectively. F1 scores in this context measure the precision and recall between bag-of-words tokens used in a human-written and predicted answer. Bolded scores are better and * indicates a significant result (α = 0.01).

plexity (PPL). Given the same L, p, and c as above, we use the language model to measure the perplexity of c given p, or $PPL = L(\ c \mid p)$. Note that we are no longer assuming that c is the *only* high quality completion. Rather, we treat c as one sample from the population of high quality completions for a given story. Thus, if our enhanced prompts consistently improve the probability of generating c in the majority of cases, we infer that our method improves the probability of generating most high-quality completions and successfully increases story comprehension. PPL is also non-stochastic, which improves the reliability of results. Unlike previous metrics commonly used for evaluating language models, our PPL metric specifically measures the quality of prompts, making it effective at evaluating this and other automatic prompt engineering tasks.

Due to the nature of the OpenAI API used to access GPT-3, we make two adjustments to the typical perplexity calculation. First, because the API only gives access to the 100 most likely tokens at each generation step, we treat any token not appearing in the top 100 as having the same likelihood as the 100th most likely token. This occurs for only 4.4% of tokens with both traditional and enhanced prompts. Second, due to cost and time limitations we run perplexity tests on only a subset (roughly 65%) of the Sparknotes test cases. Perplexity calculations using the other two language models are performed in the usual manner.

4.3 Module Analysis

In order to better understand our system's performance we measure accuracy after each module. We evaluate by measuring the frequency with which the correct answer to each question-answer pair appears in our system at seven points in the pipeline: in the original document, after extracting knowledge graph facts, after applying filtering rules, after selecting the most relevant facts from the knowledge graphs, after verbalizing facts in natural language, after constructing a prompt, and after generating a response with the language model. We use GPT-2 as the language model in these experiments. We do not perform this analysis for story completion as the literal human-written completion will never appear in any portion of the system.

LM	Method	BLEU	PPL
GPT-2	Traditional Prompts	0.0620	2.42
	Our Pipeline	**0.0622**	**2.24***
GPT-3	Traditional Prompts	**0.0643***	10.93
	Our Pipeline	0.0455	**10.89**
XLNet	Traditional Prompts	**0.0306***	1.61
	Our Pipeline	0.0254	**1.56***

Table 2: Results of story completion experiments on the Sparknotes dataset described in Section 4.2. Bolded scores are better (higher for BLEU, lower for PPL) and * indicates a significant result (α = 0.01).

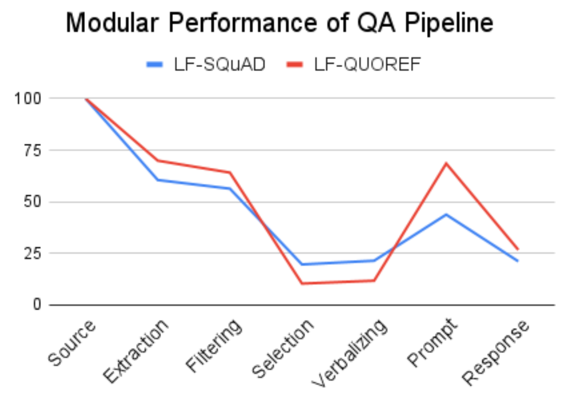


Figure 4: Percentage of correct answers retained by the system after each module of the QA Pipeline on two datasets. When desired information is lost, it usually happens during the fact selection step and when applying the language model to generate a response. The filtering step loses almost no correct answers and the verbalizing step adds correct answers. Constructing the prompt causes a spike in answer appearances because it re-incorporates as much of the document text as possible given context length constraints.

5 Results

Results for the question answering experiments are found in Table 1, and story completion results are found in Table 2. For question answering our prompt construction system outperforms traditional prompts on both datasets and using all three language models, and results are statistically significant (α = 0.01) in the majority of cases. LF-SQuAD has a longer average document length than LF-QUOREF, which accounts both for the higher overall accuracy on LF-QUOREF and our system's higher performance relative to the baseline on LF-SQuAD. This underscores the difficulty of long-form question answering by language models and validates our system's performance on the defined task.

For story completion our prompts outperform traditional prompts on the PPL metric for all three language models, including significant (α = 0.01) improvements using GPT-2 and XLNet. BLEU scores were less consistent; we achieve slightly better BLEU scores than traditional prompts using

Original Text	The age distribution was 6% aged 0–4 years, 14% aged 5–15 years, 4% aged 16–19 years…
KG Fact	[head:4% // relation:aged // tail:16-19 years]
GPT-3 Verbalized	4% of the population aged 16–19 years.
Original Text	…Milo then leads his friends towards the Valley of Sound, where they meet …Dynne, a monster made of smoke.
KG Fact	[head:a monster // relation:made // tail:of smoke]
GPT-3 Verbalized	The monster was made of smoke.
Original Text	As the two men eat borscht, a red Russian soup made of beets, Rainsford praises his host's …
KG Fact	[head:the two men // relation:eat // tail:borscht]
GPT-3 Verbalized	The two men ate borscht.
Original Text	Perez gets injured and decides to stay behind …
KG Fact	[head:Perez // relation:stay // tail:behind]
GPT-3 Verbalized	Perez stayed behind.
Original Text	Kovrin is restless and does not sleep much, but he talks a great deal …
KG Fact	[head:Kovrin not // relation:sleep // tail:much]
GPT-3 Verbalized	Kovrin doesn't sleep much.

Table 3: Examples of original document text, knowledge graph facts extracted from the text, and the corresponding sentences generated by GPT-3 via few-shot learning. Note that GPT-3 is capable of recovering meaning found in the original text but not the knowledge graph (e.g. the word "population" in the first example, which is implied in the original text and omitted in the knowledge graph).

GPT-2 and significantly worse ($\alpha = 0.01$) BLEU scores with GPT-3 and XLNet. PPL scores measure the likelihood of generating the human-written response (and by assumption other appropriate responses) given the constructed prompt, while BLEU scores compare a randomly generated response to the human-written one.

The results for module-specific experiments are in Figure 4. We find that the performance of each module is similar on LF-SQuAD and LF-QUOREF. Most of the information loss that occurs happens at two points: when selecting relevant facts from the knowledge graph and when generating a final answer using the language model. This is not particularly surprising, as these are the two phases where the most total information is pared out. The filtering step loses almost no correct answers, despite the fact that it does remove a lot of information, which means that our filtering rules are effective at removing only irrelevant knowledge graph facts. The verbalizing step actually *increases* the percentage of cases where our system retains the correct answer. This is likely due to cases when the correct *content* is contained in selected knowledge graph facts and GPT-3 adds important function words or formatting. We find similar trends when we measure the percentage of correct answer tokens retained by the system rather than measuring only the presence of the complete correct answer.

6 Conclusion

In this work we have demonstrated that a dynamic knowledge graph containing document-specific information can enhance prompt generation for large language models, thereby mitigating the limitations of finite context lengths used by transformer-based language models. We have evaluated our architecture on two story comprehension tasks, question answering and document completion, and believe that it will be successfully applied to other tasks which require comprehension of large documents. We are optimistic that these and similar techniques will allow for more practical and effective story generation systems in the future.

Each part of the presented architecture is independently trained for general purpose language processing, leaving many opportunities for potential domain-specific improvements. We are particularly interested in developing more tailored models for retrieving relevant facts from the knowledge graph and creating a feedback system for fine-tuning a domain-specific Open IE model. We also look forward to more sophisticated retrieval methods, including predicting which knowledge graph facts are likely to be relevant in the future based on learned storytelling patterns. Even as it currently stands with no domain-specific fine-tuning, our architecture is effective at enhancing the story comprehension of large language models.

We are encouraged by the rapid improvements that have been made in the area of neural story processing, including many significant developments in the past two years alone. As researchers collectively have pushed the limits of what large language models are capable of, context window sizes have proven to be a prohibitive obstacle. This work is an initial attempt at mitigating the weaknesses of finite-length context windows, and it indicates great potential for this line of research going forward.

Acknowledgements

This work has been funded in part by Latitude. We would like to thank Nick Walton and Ben Swanson at Latitude for their mentorship throughout this project, and we would like to thank Latitude and OpenAI for providing access to GPT-3 through the OpenAI API.

References

Adhikari, A.; Yuan, X.; Côté, M.-A.; Zelinka, M.; Rondeau, M.-A.; Laroche, R.; Poupart, P.; Tang, J.; Trischler, A.; and Hamilton, W. 2020. Learning Dynamic Belief Graphs to Generalize on Text-Based Games. In Larochelle, H.; Ranzato, M.; Hadsell, R.; Balcan, M. F.; and Lin, H., eds., *Advances in Neural Information Processing Systems*, volume 33, 3045–3057. Curran Associates, Inc.

Agarwal, O.; Ge, H.; Shakeri, S.; and Al-Rfou, R. 2021. Knowledge Graph Based Synthetic Corpus Generation for Knowledge-Enhanced Language Model Pre-training. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 3554–3565.

Ainslie, J.; Ontanon, S.; Alberti, C.; Cvicek, V.; Fisher, Z.; Pham, P.; Ravula, A.; Sanghai, S.; Wang, Q.; and Yang, L. 2020. ETC: Encoding Long and Structured Inputs in Transformers. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 268–284.

Ammanabrolu, P.; and Riedl, M. 2019. Playing Text-Adventure Games with Graph-Based Deep Reinforcement Learning. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, 3557–3565.

Ammanabrolu, P.; Tien, E.; Hausknecht, M.; and Riedl, M. O. 2020. How to Avoid Being Eaten by a Grue: Structured Exploration Strategies for Textual Worlds. *arXiv preprint arXiv:2006.07409*.

Angeli, G.; Premkumar, M. J. J.; and Manning, C. D. 2015. Leveraging Linguistic Structure for Open Domain Information Extraction. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, 344–354.

Bosselut, A.; Bras, R. L.; and Choi, Y. 2019. Dynamic Neuro-Symbolic Knowledge Graph Construction for Zero-shot Commonsense Question Answering. *arXiv preprint arXiv:1911.03876*.

Bosselut, A.; Rashkin, H.; Sap, M.; Malaviya, C.; Celikyilmaz, A.; and Choi, Y. 2019. COMET: Commonsense Transformers for Automatic Knowledge Graph Construction. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 4762–4779.

Brown, T. B.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A.; et al. 2020. Language Models are Few-Shot Learners. *arXiv preprint arXiv:2005.14165*.

Dasigi, P.; Liu, N. F.; Marasović, A.; Smith, N. A.; and Gardner, M. 2019. QUOREF: A Reading Comprehension Dataset with Questions Requiring Coreferential Reasoning. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 5925–5932.

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2018. Bert: Pre-training of Deep Bidirectional Transformers for Language Understanding. *arXiv preprint arXiv:1810.04805*.

Distiawan, B.; Weikum, G.; Qi, J.; and Zhang, R. 2019. Neural Relation Extraction for Knowledge Base Enrichment. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 229–240.

Fan, A.; Lewis, M.; and Dauphin, Y. 2018. Hierarchical Neural Story Generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 889–898.

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*, 8: 93–108.

Guu, K.; Lee, K.; Tung, Z.; Pasupat, P.; and Chang, M. 2020. Retrieval Augmented Language Model Pre-Training. In *International Conference on Machine Learning*, 3929–3938. PMLR.

He, H.; Balakrishnan, A.; Eric, M.; and Liang, P. 2017. Learning Symmetric Collaborative Dialogue Agents with Dynamic Knowledge Graph Embeddings. *arXiv preprint arXiv:1704.07130*.

Kumar, A.; Irsoy, O.; Ondruska, P.; Iyyer, M.; Bradbury, J.; Gulrajani, I.; Zhong, V.; Paulus, R.; and Socher, R. 2016. Ask Me Anything: Dynamic Memory Networks for Natural Language Processing. In *International Conference on Machine Learning*, 1378–1387. PMLR.

Li, J.; Luong, M.-T.; and Jurafsky, D. 2015. A Hierarchical Neural Autoencoder for Paragraphs and Documents. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, 1106–1115.

Liu, H.; and Singh, P. 2004. ConceptNet—A Practical Commonsense Reasoning Tool-Kit. *BT technology journal*, 22(4): 211–226.

Miller, G. A. 1995. WordNet: A Lexical Database for English. *Communications of the ACM*, 38(11): 39–41.

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. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 839–849. San Diego, California: Association for Computational Linguistics.

Nayak, T.; and Ng, H. T. 2020. Effective Modeling of Encoder-Decoder Architecture for Joint Entity and Relation

Extraction. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(05): 8528–8535.

Radford, A.; Wu, J.; Child, R.; Luan, D.; Amodei, D.; and Sutskever, I. 2019. Language Models are Unsupervised Multitask Learners. *OpenAI blog*, 1(8): 9.

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*, 21: 1–67.

Rajpurkar, P.; Zhang, J.; Lopyrev, K.; and Liang, P. 2016. SQuAD: 100,000+ Questions for Machine Comprehension of Text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, 2383–2392.

Reimers, N.; and Gurevych, I. 2019. Sentence-BERT: Sentence Embeddings Using Siamese BERT-Networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Roemmele, M. 2016. Writing Stories with Help from Recurrent Neural Networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 30.

Vrandečić, D.; and Krötzsch, M. 2014. Wikidata: A Free Collaborative Knowledgebase. *Commun. ACM*, 57(10): 78–85.

Wang, C.; Ma, X.; Chen, J.; and Chen, J. 2018. Information Extraction and Knowledge Graph Construction from Geoscience Literature. *Computers & geosciences*, 112: 112–120.

Xiong, C.; Merity, S.; and Socher, R. 2016. Dynamic Memory Networks for Visual and Textual Question Answering. In *International Conference on Machine Learning*, 2397–2406. PMLR.

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. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2831–2845.

Yang, Z.; Dai, Z.; Yang, Y.; Carbonell, J.; Salakhutdinov, R. R.; and Le, Q. V. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. *Advances in neural information processing systems*, 32.

Zhou, L.; and Small, K. 2019. Multi-Domain Dialogue State Tracking as Dynamic knowledge Graph Enhanced Question Answering. *arXiv preprint arXiv:1911.06192*.