## Project Report

# Long Document Summarization: Augmenting Unlimiformer with Knowledge Graphs[1]

Patrick O'Callaghan (patocal) Sheel Sansare (ssansa2) Tristan Wang (aawang99)

## To-Do List (Remove When Done)

1. Explain objective function we are optimizing during training
2. Submit test results to Scrolls.
3. Present key result that length of summary is strongly dependent on input: LD < KG + LD < KG. Explain why this is.
4. Upload models to Hugging Face.
5. Figures
6. Shakespeare image with KG / dramatis personae.
7. The Mirror and the Light (Hilary Mantel).
8. Sheel's KG.
9. Plot distribution of LD, KG, and summary sizes for the 3 splits.
10. Graph convergence of summary length (number of tokens) to 90 for LDs, 750 for combined, 800+ for KGs.
11. Training / loss and other graphs from the training. we need to discuss training in more detail eg what is the loss function? or how does the trainer work? The relevant module is `src/utils/custom_seq2seq_trainer.py` I think we need to summarize this either in the colab or the blog. This custom trainer has minor modifications of `https://huggingface.co/docs/transformers/v4.25.1/en/main_classes/trainer#transformers` Trainer This means that it uses a standard cross-entropy loss function ... adamw optimizer
12. Table of results comparing R1, R2, RL, BERTScore F1 for the 3 experiments. Bold the best performers.

[1] December 13, 2023

# 1 Introduction

In this blog post, we explore how knowledge graphs (KGs) can be applied to improve the accuracy of the `unlimiformer` long-range transformer for the task of long document (LD) summarization.

## Problem Statement

Long documents (LDs) are often difficult to understand and summarize. This is especially true for technical documents such as government reports, where entities often include obscure institutions and lesser-known individuals. In literature, one way of dealing with this complexity is to introduce a *dramatis personae*, or cast of characters, at the beginning of the text. Famous examples include complicated historical texts, such as Hilary Mantel's "The Mirror and the Light", and the Shakespearean plays. Our conjecture is that, much like how a *dramatis personae* can aid a reader in understanding a complicated novel by highlighting key relationships between characters, KGs can help large language models (LLMs) generate better summaries of LDs.

<Insert photo here>

## Knowledge Graphs of Long Documents

Standing in contrast with LDs, KGs are structured and concise, forming a significant reduction of the document to relations between entities. We use the REBEL end-to-end relation extractor to generate our KGs.

## Two New Knowledge Graph Datasets

In our project, we generate two new KG datasets, one for each example in the GovReport dataset. This is a significant undertaking for two reasons:

1. There are approximately 19,500 documents in GovReport.
2. The is signficant variance in the length of GovReport documents, which leads to major hardware management issues.

There are significant design choices relating to how relations are specified and passed to the language model to generate summaries. We specify each KG as a single sequence of subsequences, with one subsequence for each relation triplet in the KG. We then integrate the collection of KGs with GovReport. The first dataset replaces each LD in GovReport with a KG. The second dataset replaces each LD with a string that is the concatenation of the KG and LD.

### Training BART+Unlimiformer

`Unlimiformer` [**bertsch2023unlimiformer**], a recent retrieval-based method for augmenting LLMs at the decoder level, is the first long-range transformer to support unlimited length inputs. The key innovation of `unlimiformer` is to create a datastore of encodings which correspond to each token in the original document, and use the $k$-nearest neighbors ($k$-NN) algorithm to select the $k$ most relevant tokens in the datastore during decoding.

<Shouldn't this part say something about BART+unlimiformer instead of just unlimiformer?>

### Experiments

Our experiments compare the summary outputs across three datasets: the original GovReport, the GovReportKG, and the GovReportKG+LD. However, our initial findings reveal significant differences in length between the summaries. Specifically, the default BART model produces summaries of approximately 130 tokens, with a typical range of 100 to 150 tokens. In contrast, the KG and KG+LD models generated much longer summaries of approximately 900 tokens, with a typical range of 600 to 1,100. The length of golden summaries for GovReport are closer to the latter, with the average number of tokens being 600, with a typical range between 400 and 1,000.

<Table with number of tokens similar to unlimiformer Scrolls?>

We then explore the cause of these differences and refine our experiments to equalize summary length. We do so by re-initializing training with a model that is fine-tuned to produce longer summaries consistently, so we can fairly compare the performance across the three datasets.

### Overview of Results

Once we control for summary length, our final results are in line with our initial hypothesis. We summarize these results in ??.

## 2 Methodology

We compare and contrast the LD summaries generated by three transformer-based LLM models. Firstly, we train the `facebook/bart-base` model using the `unlimiformer` augmentation, which operates on the entire LD and employs the $k$-NN algorithm. Secondly, we repeat the previous exercise but with KGs as inputs instead of LDs. Thirdly, we repeat the previous exercise with string inputs of concatenated KGs and LDs (in this order).

# 3 Creating the KGs and Datasets

Our baseline dataset is the Hugging Face version of GovReport [**huang2021efficient**], a well-established LD summarization dataset with many practical applications. To generate the required datasets, we use REBEL [**huguet2021rebel**], a pre-trained, end-to-end relation extraction model that can be found on Hugging Face here[2], to perform named-entity recognition (NER) and relation extraction (RE).

## GovReport

GovReport is a well-established LD summarization dataset that is both publicly available and ready-to-use. We use it because it is a large and popular dataset with many real-world applications. The Hugging Face GovReport[3] dataset has an approximate 90/5/5% split of approximately $19,500$ document-summary pairs.

## REBEL

We use REBEL because it is end-to-end (it finds entities and relations simultaneously), open-source, and easy to implement using Hugging Face. Additionally, as per the DocRED paper by Yao et al [**yao2019DocRED**], pretrained REBEL currently yields the best joint entity and relation extraction (NER and RE) results compared with the benchmark among all models sampled, achieving a relation F1 score of 47.1[4].

Since the pre-trained REBEL model has a token limit, we split the LD into 128-token chunks before extracting 3 head-relation-tail triplets from each chunk. We split the text into 128 token chunks as it is approximately the length of one paragraph. Through visual inspection, we find that there are typically 3 triples in each paragraph. Moreover, since REBEL employs beam search, the number of triples must be less than or equal to the number of beams. We determine that the optimal number of beams, based on runtime, is 3 beams, which means that the maximum triples per chunk would be 3.

Once the triplets are extracted, we use NetworkX to create a directed graph, and use MatPlotLib to visualize and plot the results. Below is a sample image of a KG produced from a gold summary.

<Insert image / plot of KG>

<Why extract triplets, and not extract triplets typed?>

## Alternatives to REBEL Considered

Other means of performing NER and RE we considered include spaCy-LLM, DyGIE++, and LlamaIndex. spaCy-LLM is a package that integrates LLMs into natural language processing (NLP) pipelines provided by spaCy, an industry-standard NLP library. In

[2]https://huggingface.co/Babelscape/rebel-large

[3]https://huggingface.co/datasets/ccdv/govreport-summarization

[4]https://paperswithcode.com/sota/joint-entity-and-relation-extraction-on-3

particular, its built-in `spacy.REL.v1`[5] component supports RE with both zero-shot and few-shot prompting, but relies on an upstream NER component for entity extraction.

DyGIE++ [**wadden2019dygiepp**] is an RE component that refines and scores text spans designed to capture both intra-sentence and cross-sentence context. We cloned the code from the official GitHub repository and attempted to replicate the process of training a model for RE, but were unsuccessful due to technical difficulties.

Finally, LlamaIndex, a framework for connecting data sources for LLMs, has a class called `KnowledgeGraphIndex` which is compatible with FAISS, the datastore that `unlimiformer` uses to conduct $k$-NN searches of top-level hidden state encodings, which would simplify our task of NER and RE.

# 4 Training

## Unlimiformer

<Why unlimiformer, and what is it?>

**Retrieval-Augmentations of LLMs** Unlimiformer stands out for its novel integration of retrieval mechanisms directly into the Transformer architecture. This integration allows the model to dynamically access large-scale, a document-specific external (FAISS) datastore during inference. This datastore is populated with encoded representations of the full input text. During training, The key advantage of this approach is that it enables the model to augment its language generation capabilities with contextually relevant, externally stored information. This is particularly useful for tasks requiring deep, specific knowledge or for improving the model's ability to stay updated with recent information.

**Comparison with Other Methods (Datastore Access)** Unlike traditional methods where datastores are accessed externally or through separate mechanisms, Unlimiformer integrates the datastore access internally within its architecture. This internal integration facilitates a more seamless and efficient interaction between the model's language processing capabilities and the external knowledge sources. In contrast, other models might rely on separate retrieval steps or external systems to incorporate knowledge from datastores, which can introduce complexity and inefficiency. Unlimiformer's approach, therefore, represents a significant advancement in making retrieval-augmented models more streamlined and effective.

These points highlight Unlimiformer's innovative approach to enhancing LLMs with retrieval-augmented capabilities, particularly its unique internal mechanism for accessing and integrating external datastores.

[5] https://github.com/explosion/spacy-llm/tree/main/usage_examples/rel_openai

## BART

We focus on training the `facebook/bart-base` model. Although there are more advanced models, many of which are compatible with unlimiformer (e.g. Llama), BART provides the main benchmark in the `unlimiformer` paper [**bertsch2023unlimiformer**]. In addition, each model treats special tokens slightly differently, which is important to the resulting training on the KGs.

## Using BART for Training

Like other transformer-based models, BART is adept at handling structured inputs due to several key features of its architecture and design. *Structured inputs* refer to data that is organized in a predictable, often hierarchical manner, with clear relationships between different parts. This contrasts with unstructured data, like free-form text, where the organization and relationships are not as explicitly defined. Examples of structured inputs include XML and JSON data, databases, and tables, where elements are nested and have defined relationships.

<How the training actually works, and how it relates to KGs.>

### 4.0.1 Background on BART

**Structured Inputs **Why BART Handles Structured Inputs Well**

1. **Self-Attention Mechanism**: BART's transformer architecture uses a self-attention mechanism, which allows it to consider the entire input sequence at once. This enables the model to understand relationships between different parts of the input, essential for structured data.

2. **Contextual Understanding**: BART can capture context from both left and right of each token in the input sequence. This bi-directional context is crucial for understanding structured inputs, where the meaning often depends on the surrounding elements.

3. **Layered Encoding**: The layered structure of transformers enables them to capture and encode different levels of abstraction, which is beneficial for understanding hierarchical and nested structures in the input.

4. **Pre-training on Diverse Data**: BART is pre-trained on a wide range of data, including structured formats. This pre-training helps it to learn patterns and structures that are common in various types of data.

5. **Flexibility in Input Representation**: BART can handle sequences with special tokens and delimiters, allowing it to adapt to different types of structured inputs. For example, it can process inputs where parts of the data are segmented or highlighted using special tokens.

6. **Adaptability to Task-Specific Structures**: With fine-tuning, BART can adapt to specific types of structured inputs relevant to a particular task, enhancing its ability to process and generate meaningful outputs based on that structure.

In summary, BART's ability to process and understand the entire input sequence contextually, along with its adaptability and pre-training on diverse data, makes it well-suited for handling structured inputs. This capability allows it to effectively process and generate outputs based on inputs like knowledge graphs, tables, or other structured data forms. We chose to use the beginning of sequence (BOS, '<s>') and end of sequence (EOS, '</s>') tokens to separate triples in our knowledge graphs (KGs) with the intent of aligning BART's understanding of sequence boundaries, this approach has specific implications:

1. **Clear Segmentation of Information**: Using BOS and EOS tokens to delimit triples in the KG makes each triple a distinct segment from the model's perspective. This is beneficial since we want the model to treat each triple as an independent unit of information since we expect our GovReport KGs to be such that the relationships within triples contain key information.

2. **Facilitating Attention Across Segments**: This segmentation should help the model's attention mechanism focus on each triple individually, potentially enhancing the model's ability to capture the nuances of each relationship within the KG.

3. **Model Adaptation to Structured Inputs**: Given that BART is designed to handle structured text, using BOS and EOS tokens in this way could aid the model in better understanding and generating summaries based on the structured nature of KGs. It aligns with the model's pre-existing mechanisms for processing text.

4. **Potential for Contextual Integration**: While each triple is treated as a separate sequence, the overall structure still allows the model to integrate these segments contextually. The model can learn to understand the KG as a whole, even though it processes each triple individually.

5. **Efficient Processing of Smaller Units**: By breaking down the KG into smaller segments, the model might process each unit more efficiently, especially if the triples are concise and the relationships within them are straightforward.

In this context, the slower training times you observed might not be due to the tokenization strategy per se but could involve other factors such as the complexity of the relationships in the KGs, the adaptation of the model to this unique structuring of inputs, or other computational aspects related to how the BART model processes these inputs.

Your approach aligns with the design principles of transformer models like BART, which are adept at handling structured inputs. The key would be to ensure that the rest of your training pipeline, including data preprocessing and model fine-tuning, is optimized to leverage this structure effectively.

## Appropriateness of the BART Model

When training our model, we chose to feed the relational data of our KGs as tokens into `unlimiformer`, as opposed to embedding the KGs as separate relations into vector space. We believe that our approach is more appropriate as it allows us to better utilize the the `unlimiformer` framework, while preserving as much of the KG structure as possible within the dataset.

# 5 Results

<How did our model perform compared to the baseline? Explanation?>
<Why is the average summary 800 words and not 500 words?>

Interpreting the performance differences between models trained on long documents (LD) and knowledge graphs (KG) based on the provided metrics involves considering what each metric measures and how that relates to the nature of the inputs:

1. **ROUGE Scores**: - **ROUGE-1 (LD: 23, KG: 40)**: This measures the overlap of unigrams (individual words) between the generated summary and the reference summary. The substantially higher score for KG suggests that the KG-based model is better at capturing key content words. This could be because KGs, being structured and concise, might enable the model to focus on essential terms more effectively. - **ROUGE-2 (LD: 11.74, KG: 11.47)**: This metric evaluates bigram overlap, indicating how well the model captures phrases and specific content. The similar scores suggest that both models are nearly equally effective at capturing phrase-level information, though the LD model has a slight edge. - **ROUGE-L (LD: 14.7, KG: 17.7)**: ROUGE-L assesses the longest common subsequence, which reflects sentence-level structure and coherence. The higher score for KG indicates better preservation of sentence structure or flow from the KG inputs.

2. **BERTScore**: - **Precision (LD: 0.69, KG: 0.58)**: Precision measures how much of the content in the generated summary is relevant or present in the reference summary. The higher precision for LD implies that it might be better at generating content closely aligned with the reference, likely due to the richer context provided by the long document. - **Recall (LD: 0.52, KG: 0.57)**: Recall assesses how much of the reference summary is captured in the generated summary. The higher recall for KG suggests it is better at including essential points from the reference summary, possibly due to the distilled and focused nature of KGs. - **F1/Aggregated BERTScore (LD: 0.59, KG: 0.57)**: This balanced metric considers both precision and recall. The scores are quite close, indicating that overall, both models are similarly effective in terms of content relevance and coverage, though the LD model has a marginal advantage.

Given these results after 8000 training steps:

- The KG-based model excels in capturing key content words and maintaining sentence structure, likely due to the concise and structured nature of KGs. - The LD-based model, despite its lower performance in some ROUGE metrics, achieves higher precision in BERTScore, suggesting its strength in generating summaries that are more aligned with the content and style of the reference summaries. This might be due to the rich, contextual information available in long documents. - The marginal differences in ROUGE-2 and BERTScore/F1 indicate that both models have their strengths and weaknesses. The LD model seems to be better for generating precise content, while the KG model is more effective in covering essential information and maintaining structure.

In conclusion, each model has its advantages depending on the desired outcome of the summary: whether the focus is on precise content alignment (LD) or on covering key points and maintaining structure (KG). Continued training and further optimization could enhance the performance of both models, potentially narrowing these gaps.

# 6 Conclusion

<Do we recommend using KGs for LD summarization?>