Project Milestone

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

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

1 Extracting Knowledge Graphs from Long Documents

REBEL

We have built a simple knowledge graph for one instance of a long document. Specifically, we built a knowledge graph for the first row of the GovReport validation set. We did this by using an open-source, pre-trained, end-to-end relation extraction model called REBEL [HN21] which can be found on Hugging Face here.[2] This model takes an input string and extracts head-relation-tail triplets. A sample of extracted relations is shown below (see figure 1).

```
{'head': 'Peterson Air Force Base', 'type': 'operator', 'tail': 'Air Force', 'meta': {'spans': [[1125, 1253]]}}
{'head': 'Military Information Support Operations', 'type': 'inception', 'tail': '2018', 'meta': {'spans': [[1250, 1378]]}}
{'head': 'Joint Staff', 'type': 'parent organization', 'tail': 'DOD', 'meta': {'spans': [[1250, 1378]]}}
```

Figure 1: Three example triplets extracted from REBEL.

We chose to use REBEL because it finds entities and relations at the same time (end-to-end), it is open source, and it is easy to implement using Hugging Face. One downside of REBEL is that it is limited in the number of relations that it can extract (approximately 200). As a result, we are still looking into alternatives. Since the pre-trained REBEL model has a token limit, we wrote some code that splits the long document into 128-token chunks. Triplets were then extracted for each chunk. Next, we used NetworkX to assemble the head-relation-tail triplets into a directed graph to form the knowledge graph. The plotted knowledge graph for the first document in GovReport is shown below (see figure 2). Our code for the REBEL implementation can be found here [3] and our code for the knowledge graph creation and plotting can be found here [4].

DyGIE++

Another relation extraction method we explored for building KGs is DyGIE++ which refines and scores text spans designed to capture both intra-sentence and cross-sentence

[1] November 10, 2023

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

[3] https://github.com/patrickocal/unlimiformer/blob/main/rebel.py

[4] https://github.com/patrickocal/unlimiformer/blob/main/main.ipynb

context. We have cloned the code from the official GitHub repository (`https://github.com/dwadden/dygiepp`) and attempted to replicate the process of training a model for relation extraction on the SciERC dataset, but have not yet been successful due to technical difficulties. To resolve this issue, we aim to raise a GitHub issue and continue to debug as needed, in anticipation that DyGIE++ may be used later on in our project.

LlamaIndex

`LlamaIndex` is a framework for connecting data sources for Large-Language Models. In particular `LlamaIndex` has a class called `KnowledgeGraphIndex`. We have established that the latter is compatible with `FAISS` which is the datastore that `unlimiformer` uses to conduct a k-nearest neighbor search of top-level hidden state encodings that are most relevant to the query. `https://github.com/run-llama/llama_index/issues/8767`. This means that we can store the Knowledge Graph we create using LlamaIndex as a FAISS datastore. The idea is to create a secondary channel for the attention mechanism. The potential advantage is characterised in the following notebook: KG-datastore-notebook. If a nearest neighbor search over a KG produces fewer, but more relevant results, this may help focus the cross-attention mechanism on what is most important (see figure 3).

2 Understanding and Implementing Unlimiformer

Unlimiformer [Ber+23] is a nontrivial way to augment Large-Language Models. It involves capturing hidden states from the encoder and placing them in a datastore. We have successfully run basic `inference_example.py` for our baseline `unlimiformer` model. We have also successfully trained the baseline Bart model on the GovReport dataset. We report our initial results in section C. We encountered a number of issues during this step and we raised Github issues with the authors/developers of `unlimiformer` (see `https://github.com/abertsch72/unlimiformer/issues/33`).

We have also taken significant steps towards understanding the `unlimiformer` methodology and codebase. We are in detailed discussions with the authors in this respect (see `https://github.com/abertsch72/unlimiformer/issues/55`). We have made detailed comments to the codebase to make it more readable and understandable to ourselves and others (see documents with "rich comments" updates in the forked repository `https://github.com/patrickocal/unlimiformer/tree/main/src`)

The goal is to be able to suitable modify and augment the `unlimiformer` codebase with Knowledge Graphs. We have learnt that `unlimiformer` only uses the FAISS datastore for inference at the testing stage. Thus, if we are to create a secondary FAISS datastore (using the fact that a LlamaIndex KnowledgeGraphIndex can be stored as using FAISS), then the same applies. This presents a potential opportunity for us in that we may be able to replace FAISS and the `faiss.knn_gpu` method with related PyG methods such as `pool.knn_graph` which would potentially allow us to extract relations from the k-nearest entities. This presumes we are able to establish and implement an isomorphism between

entities and bags of tokens. ****We would appreciate staff feedback and a discussion/meeting in relation to whether we should pursue this or whether we should pursue other less technical approaches to integrate KGs with the unlimiformer approach.****

We discuss these in the next section.

3 Next Steps

One next step would be fine-tuning the REBEL model on the long-document summarization datasets we chose, namely the Hugging Face versions of GovReport [Hua+21] and BookSum [Kry+21]. The REBEL model is proven to be relatively flexible, having been fine-tuned on at least 4 widely-used relation extraction (RE) datasets of diverse contexts (CONLL04, DocRED, NYT, and ADE) and performed well on most of them. However, we bear in mind that the 4 aforementioned datasets differ significantly from the one we propose. DocRED and ADE entities are words or short phrases, while CONLL04 and NYT entities are sentences from news articles. On the other hand, the entities in our dataset will be summaries. Fortunately, DocRED benchmark data `https://paperswithcode.com/sota/joint-entity-and-relation-extraction-on-3` suggests that, when pre-trained, REBEL performs well on joint entity and relation extraction on DocRED, which is similar to what we are trying to accomplish.

As such, we plan to train REBEL on GovReport and BookSum using the method outlined in part 4 of the original REBEL paper [HN21]. We note that even though REBEL as a standalone model can extract more than 200 different relation types, this may still prove insufficient for summarizing long documents. Should this be the case, we plan to implement a 2-stage extraction process. One such option would be to use DyGIE for the entity extraction, then use DREEAM for the relation extraction.

Our long-document summarization project is dependent on the fact that KGs can encode the key semantic relationships between entities in a document in a more concise manner than a full datastore of the entire long document. We aim to evaluate the quality of our KGs by feeding them into a language model to obtain a hidden encoding of the the KG. Then, we will evaluate them against the gold summaries provided in the GovReport and BookSum datasets using ROUGE-1 (unigram), ROUGE-2 (bigram), ROUGE-L (sub-sequence), and BERTScore. The ROUGE metrics compare summarization via lexical overlap between the model-generated and gold summaries, while BERTScore compares the semantic similarities of the two using the BERT embeddings.

If the KGs produced by REBEL are too large to be converted directly into summaries, we will use the graph-to-graph (G2G) method detailed in part 6.2 of [Wu+20]. In short, the G2G method encodes the input KG with a GAT and uses the resulting node representations to make a binary salience prediction and generate a summary subgraph. We can then feed this smaller subgraph into ChatGPT and obtain a summary.

References

[Wu+20] Zeqiu Wu et al. "Extracting summary knowledge graphs from long documents". In: *https://arxiv.org/pdf/2009.09162.pdf* (2020).

[Hua+21] Luyang Huang et al. "Efficient Attentions for Long Document Summarization". In: *https://arxiv.org/pdf/2104.02112.pdf* (2021).

[HN21] Pere-Lluís Huguet Cabot and Roberto Navigli. "REBEL: Relation Extraction By End-to-end Language generation". In: *Findings of the Association for Computational Linguistics: EMNLP 2021*. Ed. by Marie-Francine Moens et al. Punta Cana, Dominican Republic: Association for Computational Linguistics, Nov. 2021, pp. 2370–2381. DOI: 10.18653/v1/2021.findings-emnlp.204. URL: https://aclanthology.org/2021.findings-emnlp.204.

[Kry+21] Wojciech Kryściński et al. "BookSum: A Collection of Datasets for Long-form Narrative Summarization". In: *https://arxiv.org/abs/2105.08209* (2021).

[Ber+23] Amanda Bertsch et al. "Unlimiformer: Long-Range Transformers with Unlimited Length Input". In: *https://arxiv.org/pdf/2305.01625v1.pdf* (2023).

A Example of Extracted Knowledge Graph

Figure 2: KG for the first document of the GovReport validation set: extracted using REBEL. Node labels are in black and edge labels are in red.

B KG-Datastore Example

Comparison of results

Let's put results together with their LLM tokens during the query process:

> Tell me events about NASA.

	VectorStore	Knowledge Graph + VectorStore	Knowledge Graph
Answer	NASA scientists report evidence for the existence of a second Kuiper Belt, which the New Horizons spacecraft could potentially visit during the late 2020s or early 2030s. NASA is expected to release the first study on UAP in mid-2023. NASA's Venus probe is scheduled to be launched and to arrive on Venus in October, partly to search for signs of life on Venus. NASA is expected to start the Vera Rubin Observatory, the Qitai Radio Telescope, the European Spallation Source and the Jiangmen Underground Neutrino. NASA scientists suggest that a space sunshade could be created by mining the lunar soil and launching it towards the Sun to form a shield against global warming.	NASA announces future space telescope programs on May 21. **NASA publishes images of debris disk on May 23. NASA discovers exoplanet LHS 475 b on May 25.** NASA scientists present evidence for the existence of a second Kuiper Belt on May 29. NASA confirms the start of the next El Niño on June 8. NASA produces the first X-ray of a single atom on May 31. NASA reports the first successful beaming of solar energy from space down to a receiver on the ground on June 1. NASA scientists report evidence that Earth may have formed in just three million years on June 14. NASA scientists report the presence of phosphates on Enceladus, moon of the planet Saturn, on June 14. NASA's Venus probe is scheduled to be launched and to arrive on Venus in October. NASA's MBR Explorer is announced by the United Arab Emirates Space Agency on May 29. NASA's Vera Rubin Observatory is expected to start in 2023.	NASA announced future space telescope programs in mid-2023, published images of a debris disk, and discovered an exoplanet called LHS 475 b.
Cost	1897 tokens	2046 Tokens	159 Tokens

Figure 3: This is an example taken from KG-datastore-notebook that shows how the different structure of a Knowledge graph produces very different results in the associated, yet different, context of querying a document or collection of documents.

C Training Bart on GovReport using Unlimiformer

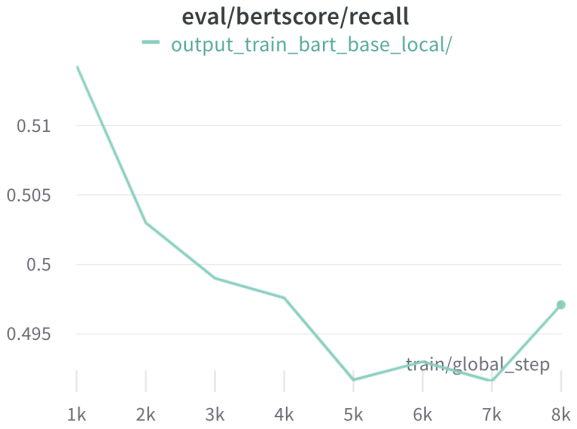

Figure 4:

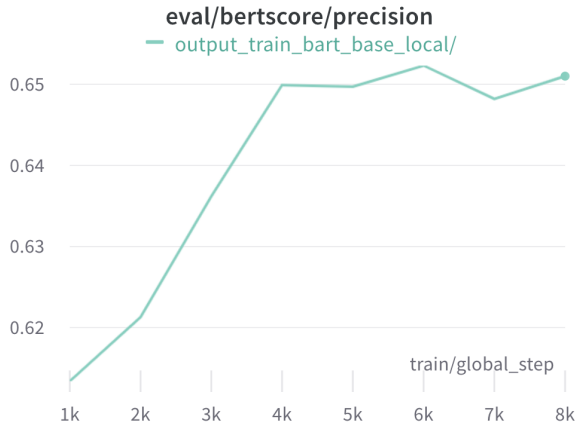

Figure 5:

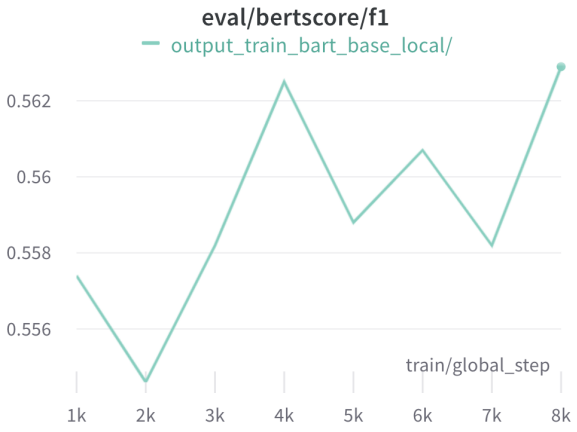

Figure 6:

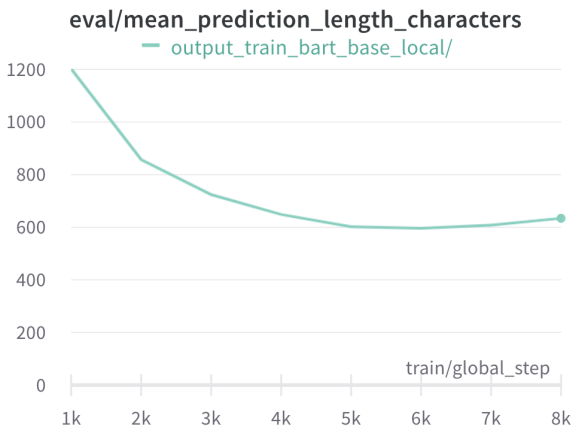

Figure 7:

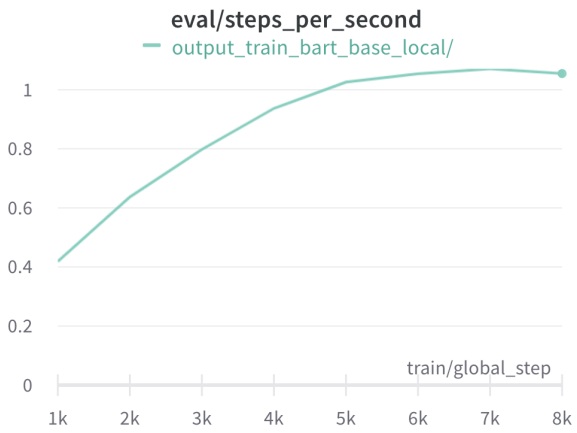

Figure 8:

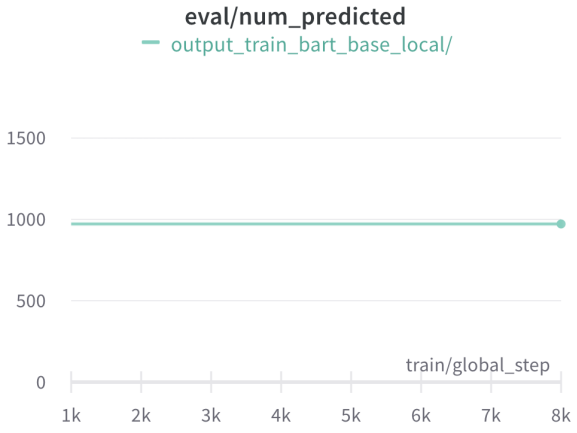

Figure 9:

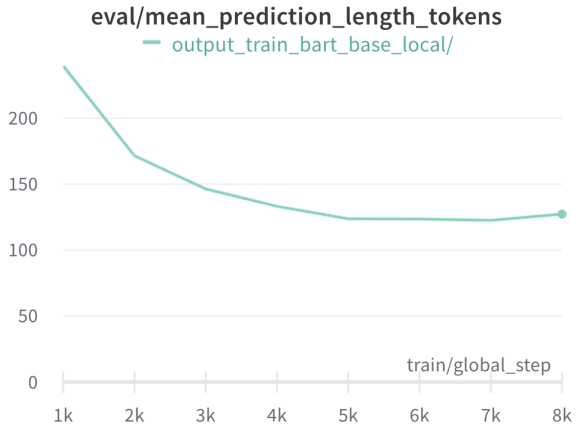

Figure 10:

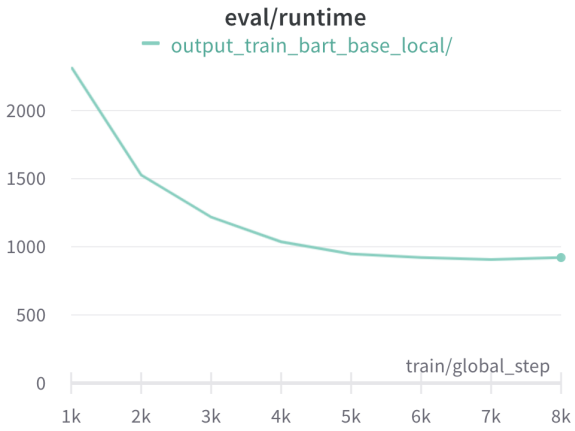

Figure 11:

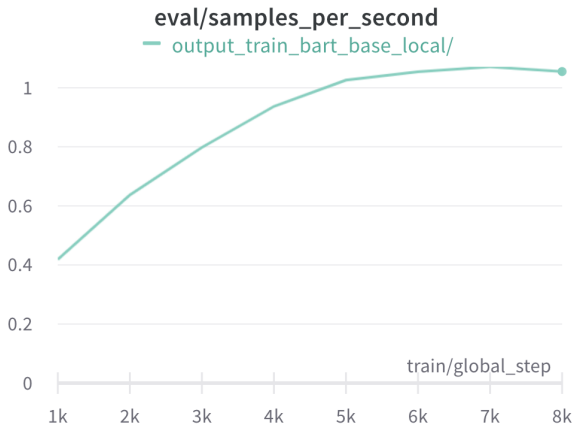

Figure 12:

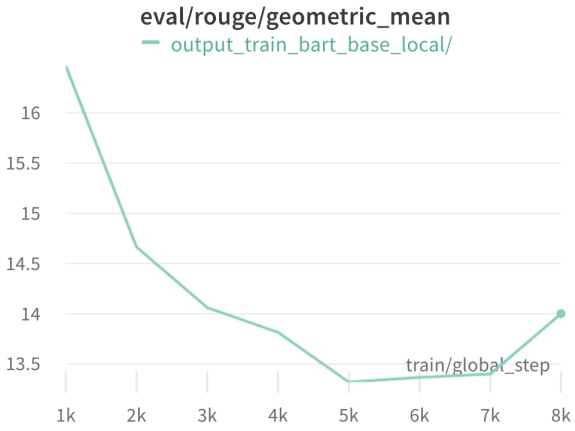

Figure 13:

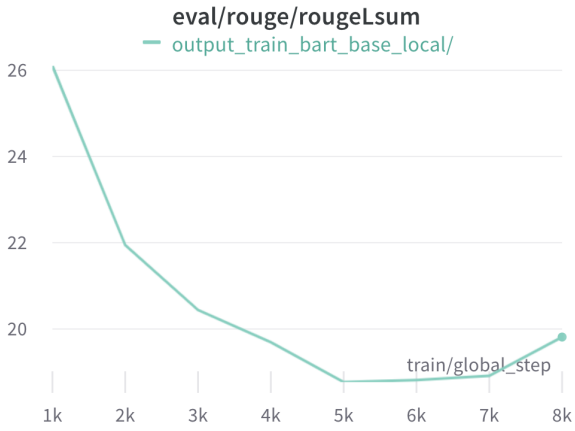

Figure 14:

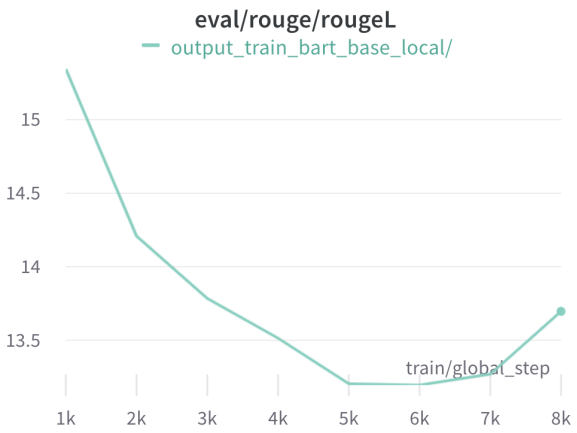

Figure 15:

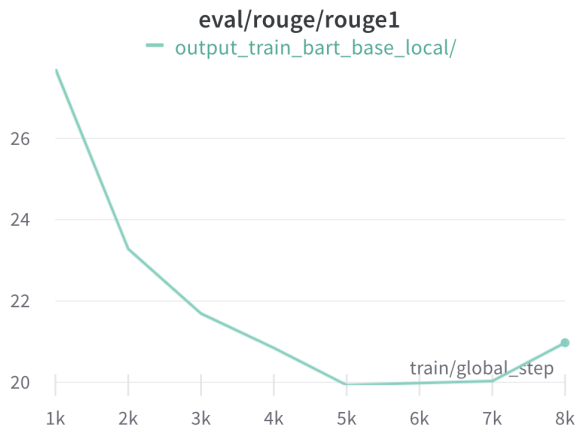

Figure 16:

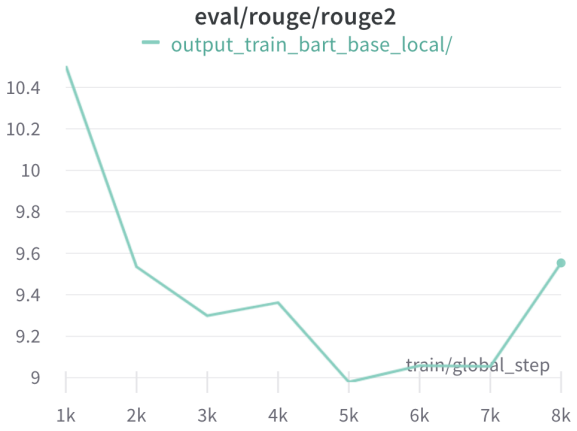

Figure 17:

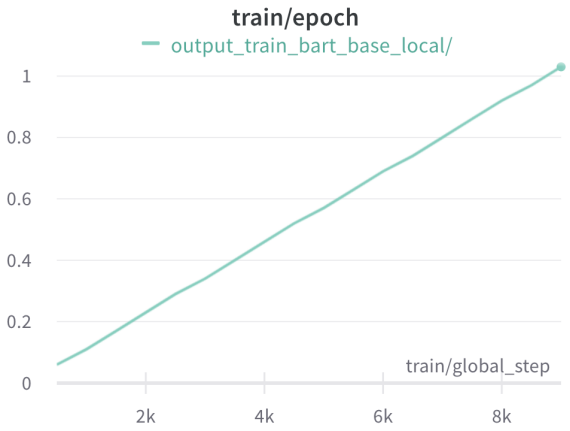

Figure 18:

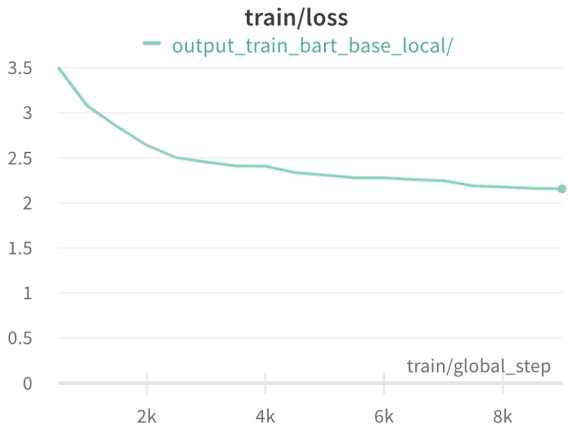

Figure 19:

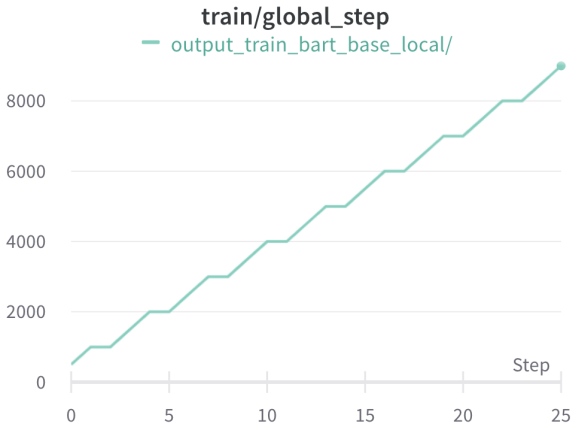

Figure 20:

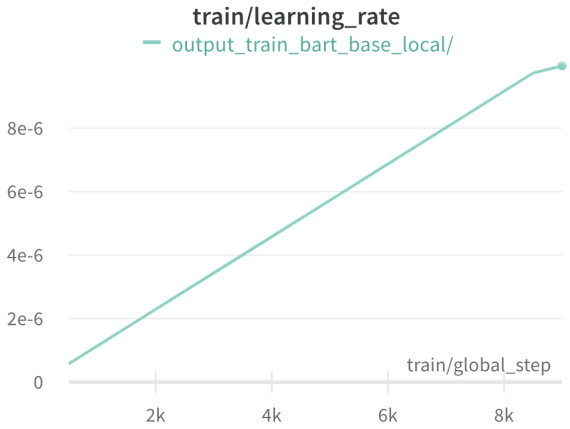

Figure 21: