TellMeWhy with Small Models: Fine-Tuning Through Contextual Injection

Jaret McManus, Dane Meister

December 7, 2024

Abstract

This project explores how context injection impacts small transformer models like T5. It focuses on answering "why" based questions from narrative texts—a task that requires causal reasoning and commonsense knowledge. We introduce a method of injecting this external context, generated by the Gemini language model, into the TellMeWhy dataset. Our findings show a slight improvement in T5's performance, reflected in higher human evaluation scores. While automated metrics like BLEURT and ROUGE-L assess semantic alignment, they often misalign with human evaluations, particularly in causal reasoning tasks. Context injection benefited scenarios with implicit answers but introduced challenges in misinterpreting details and sequence processing due to limited training data. These results suggest that maximizing context injection requires larger datasets, refined evaluation metrics, and more efficient strategies, with future work focusing on long-sequence models and domain-specific fine-tuning.

1 Project Overview

Our project goal is to tackle answering "why" questions based on narratives, a usual task for applications like conversational AI and educational tools. This problem requires inferring causal relationships, understanding implicit assumptions (Pichotta and Mooney, 2014), and integrating external knowledge (Schank and Abelson, 1975), capabilities that exceed simple pattern recognition. These challenges are compounded by the lack of datasets with contextual information and the computational demands of large models. We narrow our focus to the TellMeWhy dataset, which contains "why" questions about narratives. The goal is to enhance the accuracy of a T5 model—a transformer-based text-to-text model—by injecting external commonsense context into the dataset with a larger language model—Gemini—so when the smaller model is training, it hopefully should perform better at answering questions about narratives, even when the answer isn't explicit in the text. Our goal is showing improved performance without the help of resource-intensive LLMs in real-time applications using the smaller fine-tuned models.

Existing question-answering methods, such as GPT, BERT, and T5, excel at fact-based tasks but struggle with "why" questions requiring causal reasoning and commonsense knowledge (Schank and Abelson, 1975; Pichotta and Mooney, 2014). Datasets like SQuAD and BoolQ focus on factual answers, overlooking explanation-based reasoning (Rajpurkar et al., 2016). COMET integrates commonsense knowledge graphs, and datasets like TellMeWhy are used to fine-tune models for reasoning tasks (Lal et al., 2021). However, these methods are limited by the scope of contextual data and the computational demands of large models, leading to poor generalization when required knowledge lies beyond the text. The TellMeWhy dataset, with over 30,000 "why" questions, highlights the challenge, as even fine-tuned models like T5 and GPT-2 struggle to generate plausible answers for questions requiring implicit reasoning (Lal et al., 2021). Addressing these gaps requires injecting training datasets with high-quality commonsense knowledge and developing efficient models capable of causal reasoning without heavy computational overhead.

To address the issues stated for answering "why" questions, our project uses a context-injection approach, with the goal of improving a smaller and more efficient model—T5—during training by supplying external commonsense knowledge. Specifically, we employ the LLM Gemini, to generate context using an engineered prompt. This context integrated into the TellMeWhy dataset, will ideally enable the T5 model to perform better than without context training. By incorporating this pre-processed context during training, our approach minimizes the reliance on resource-intensive models during inference. We will compare how a pretrained T5 model, a finetuned T5 model, and a finetuned T5 model with injected context perform given answerable questions with necessary context present, and narratives with questions asked without the context present during evaluation. Our evaluation approach will utilize a comprehensive set of metrics tailored to capture different aspects of performance as well as human evaluation.

Our primary idea, context injection, enhances the dataset with commonsense knowledge. Language models struggle with implicit knowledge, such as causal relationships or social norms, due to their reliance on statistical patterns and biases in training corpora focused on factual data (Schank and Abelson, 1975; Pichotta and Mooney, 2014). To overcome this, we use the Gemini language model to automatically generate context incorporating causal links, implicit assumptions, and external knowledge relevant to the TellMeWhy dataset. These contexts are appended to the training data, equipping the smaller T5 model with broader information during fine-tuning. This automated process leverages Gemini's pretrained capabilities to upgrade the dataset efficiently while maintaining quality, producing a ready-to-use JSON file for downstream tasks.

Using the augmented version of the TellMeWhy dataset, with context generated by the Gemini model, we will fine-tune the T5-small model to answer "why" questions. The dataset will be preprocessed into Hugging Face format, with inputs combining a narrative, an optional context, and a question, and the target being the corresponding answer. Tokenization included truncating and padding to a maximum length of 128 tokens, with padding in labels replaced by -100 to ignore them during loss computation. Separate train-test splits will be created for context-injected and no-context datasets to enable comparative analysis. The fine-tuning process will utilize a Trainer object from the Hugging Face Transformers library, employing gradient accumulation and mixed-precision (fp16) training for efficient resource use. The fine-tuned models will be evaluated using BLEURT, ROUGE-L F1, Exact Match, and F1 scores to measure semantic relevance, sequence overlap, and prediction accuracy. We will necessitate human evaluation for a comprehensive understanding of model performance.

To evaluate our approach, we address key questions: How does context injection improve the T5 model's ability to answer "why" questions? Does the enriched dataset enhance causal reasoning and semantic relevance? Using the T5-small model fine-tuned on the TellMeWhy dataset and commonsense context generated by Gemini, we employed metrics such as BLEURT (semantic similarity), ROUGE-L F1 (sequence overlap), Exact Match (EM), and F1 Score (precision and recall). Given BLEURT's limitations in assessing causal reasoning, we conduct human evaluations on 100 samples per model, scoring outputs on a scale of [-2, 2] based on narrative quality, question alignment, and correctness. Poorly labeled questions or answers will be reassessed for proper evaluation. Additionally, we analyze training and validation loss trends, token efficiency, and compare the context-injected model's performance against a baseline T5 model, offering a comprehensive view of our method's strengths and areas for future refinement.

Our findings demonstrate that context injection slightly improves the T5 model's ability to answer "why" questions, as reflected by higher human evaluation scores (0.643 with context vs. 0.457 without, on a [-2,2] scale). While automated metrics like BLEURT and ROUGE-L captured semantic and token alignment, they often misaligned with human evaluations, revealing their limitations in assessing causal reasoning. Context injection enhanced reasoning quality in scenarios where the target answer

lacked clarity or explicit relevance. However, the model sometimes misinterpreted nouns or referenced incorrect text sections, likely due to limited training data and the complexity of injecting exhaustive commonsense knowledge. Increased token length from context injection also introduced sequence processing constraints, highlighting the need for larger models or more efficient strategies. These results suggest that maximizing the benefits of context injection requires larger datasets, refined metrics, and adaptive strategies, with future work focusing on long-sequence models and domain-specific fine-tuning to enhance reasoning and alignment with human judgment.

2 Ideas

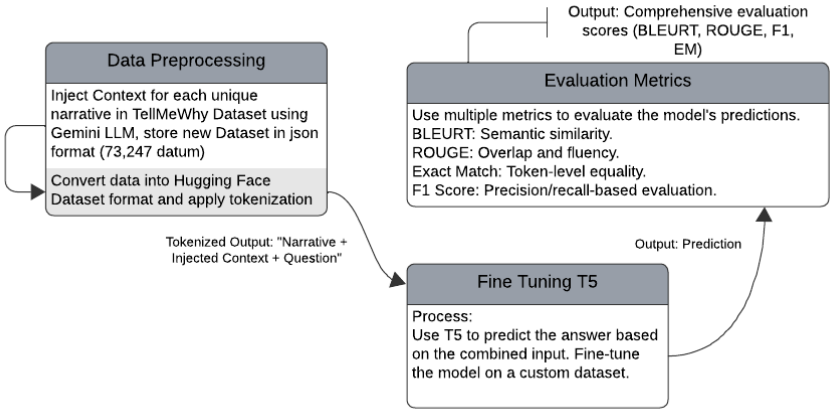


Figure 1: Method Details & Idea Block Diagram

2.1 Idea 1: Context Injection with External Commonsense Knowledge Using LLMs

The first idea in our project focuses on augmenting the TellMeWhy training dataset with externally generated commonsense context using a large language model (LLM), specifically Gemini (We had attempted to also test a LLAMA model, but could not due to time constraints). The goal of this approach is to improve the input narratives by providing relevant, concise, and informative background knowledge that enhances the reasoning ability of smaller models—in our case this is T5 small—when answering ”why” questions.

2.1.1 Implementation

The process of context injection involves the following steps:

1. **Dataset Preparation:** We started with the TellMeWhy dataset, which consists of narrative-question-answer triples. This dataset was loaded into a Python environment—hosted on Google Colab—using the Hugging Face `datasets` library.

2. **Prompt Engineering:** We tested several prompts on a small subset of examples from the dataset and researched the effects of delimiters and other factors in prompt engineering to carefully craft our prompt designed to guide Gemini in generating relevant commonsense context for each narrative.

3. **API Integration with Gemini:** Using the Google Generative AI SDK, Gemini was configured to generate context for each narrative. To avoid redundant API calls, narratives with the same content reused previously generated context. This optimization reduced the number of API requests and

associated costs. For each narrative in the dataset, Gemini-generated context was appended as an additional field.

Gemini Prompt for Context Injection

Given the following narrative sentences that describe a story, produce a sequence of concise and to-the-point sentences that bring in commonsense information and external world knowledge that is relevant. Be very verbose about commonsense knowledge and explain the reason why things are done.
Example: *Narrative:* Cam ordered a pizza and took it home. He opened the box to take out a slice. Cam discovered that the store did not cut the pizza for him. He looked for his pizza cutter but did not find it. He had to use his chef knife to cut a slice.
Context: Pizza is a food. People eat food when they are hungry. Pizza is usually already cut. Cam got the pizza from the store.
Task: Produce context sentences to the following narrative without any formatting, just as a sequence of 4 short, simple, and single-clause sentences. Do NOT reason through multiple sentences; each sentence should state commonsense information related to the narrative: **{narrative}**

Figure 2: Gemini Prompt for Context Injection

Example Narrative with Injected Context

Narrative: Bill's doctor told Bill that he needed to take care of his health. Bill decided to run the Boston Marathon. Bill trained every morning for nine months. Finally, the day of the marathon came and Bill ran it in 4 hours. Bill was finally healthy!
Injected Context:

- Running a marathon is a long and strenuous activity.
- Training is important so that people will be able to run a marathon without seriously injuring themselves.
- Marathons typically take several hours to complete.
- People who run marathons often celebrate their accomplishment and achievement.

Figure 3: Example Narrative with Injected Context using the Gemini Model

4. **Injected Context:** The injected context provided background knowledge relevant to the narrative and question. The final dataset was stored in JSON format for further preprocessing and training. Any errors during the API calls were logged, and partial results were saved for future use if necessary.

2.2 Idea 2: Fine-Tuning T5 on Specialized Dataset (TellMeWhy)

The second key idea in our project focused on fine-tuning the T5-small model—a transformer-based sequence-to-sequence model—on the TellMeWhy dataset. This is the model we train to generate well-reasoned and accurate answers to the "why" questions, by tailoring it to the nuances of causal reasoning and commonsense knowledge present in the augmented dataset. We fine-tuned the T5 model in two configurations: One using the baseline TellMeWhy dataset without additional context, and another trained on the supplemented dataset (as implemented in Idea 1). We will also test the performance of a pre-trained T5 model. By comparing the performance of these configurations, we could understand the impact of context injection on the model's reasoning and answer generation. BLEURT and ROUGE-L F1 scores were computed after each epoch for evaluation during training.

2.2.1 Implementation

Dataset Preprocessing: The TellMeWhy dataset was preprocessed to ensure compatibility with the T5 model and the Hugging Face Transformers framework:

1. **Dataset Formatting:** Each input was formatted as a concatenation of the narrative, context (when applicable), and a "why" question. The target was the corresponding answer.

2. **Tokenization:** Inputs and targets were tokenized using the T5 tokenizer, with truncation and padding to a maximum length of 128 tokens. Padding tokens in labels were replaced with -100 to ignore them during loss computation.

3. **Dataset Splits:** The processed data was split into training and test subsets. Two separate datasets were maintained (being the no-context dataset and the context-injected dataset).

Fine-Tuning: We used the Hugging Face Transformers library to fine-tune the T5 model (See3.1)

2.2.2 Unimplemented Ideas:

We explored the idea of integrating BLEURT scores directly into the model's loss function to guide the T5 model toward generating outputs with higher semantic similarity and reasoning quality. However, BLEURT is not differentiable, making it incompatible with the gradient-based optimization used in cross-entropy loss. Implementing this would require additional techniques, such as reinforcement learning or differentiable approximations of BLEURT, which were beyond the computational resources and scope of our project. Consequently, we focused on using BLEURT as an evaluation metric rather than embedding it into the training process.

2.3 Idea 3: Comprehensive Evaluation Framework

To evaluate the effectiveness of our approach, we implemented a robust evaluation framework comprising both automated metrics and human evaluation. We evaluated 100 outputs per model, scoring reasoning, relevance, and correctness on a scale of -2 to +2, with a focus on implicit understanding. Automated metrics like BLEURT, ROUGE-L F1, and Exact Match complemented human assessments, providing a comprehensive performance overview.

3 Experimental Setup

3.1 Models

For this project we used pretrained T5 transformers for question answering. Specifically we used "T5-small" through HuggingFace's transformers API on a Google Colab Notebook and went on to finetune the models to the target task. We used HuggingFace's `Trainer` in order to finetune the models. T5-small is a transformer-based text-to-text model known for its ability to handle various natural language processing tasks through sequence-to-sequence learning. The T5-small model comprises 6 encoder and 6 decoder layers, with a hidden size of 512 and 8 attention heads per layer.

1. **Model Configuration:**
    - Model: T5-small.
    - Loss function: Cross-entropy loss with label smoothing for better generalization.
    - Tokenizer: T5 tokenizer.
2. **Training Strategy:**
    - Batch Size: Due to memory constraints, a batch size of 1 was used, with gradient accumulation over 4 steps to simulate a larger batch size.
    - Mixed-Precision Training (FP16): Enabled to reduce memory usage and speed up computations.

- Training Epochs: 4 epochs, determined empirically to balance training time and performance.
- Optimizer and Scheduler: AdamW optimizer with a linear learning rate scheduler.
- Learning Rate: $5e-5$

The training utilized an evaluation accumulation strategy to process larger validation sets, and logging steps were set to every 60 updates to track performance trends. The fine-tuning process emphasized efficient resource use and balanced computational constraints with model performance goals.

3.2 Dataset

We are using the TellMeWhy dataset for our project, a large-scale crowd-sourced dataset of over 30,000 "why" questions based on narratives. Each instance consists of a narrative (a short story or description of an event), a corresponding "why" question, and a text-based explanation as the answer. After augmentation, each instance includes an additional context field generated using external commonsense knowledge. Our custom dataset comprises 73,247 samples, with input lengths ranging from 55 to 256 tokens (mean=131.13, most frequent=130) and target lengths between 3 and 71 tokens (mean=11.07, most frequent=9). For training, we used 8,500 instances, while 1,500 were reserved for evaluation testing. Each input consists of a narrative, optional context, and a question, with the output being the answer.

3.3 Evaluation Metrics

We evaluated our model using both automatic metrics and human evaluation to compare the performance of different versions of the model.

- **BLEURT**: measures semantic similarity between the generated answers and the reference answers. It leverages pre-trained language models to evaluate the fluency, coherence, and semantic relevance of outputs. Higher BLEURT scores indicate better alignment with the reference answers.

- **ROUGE**: measures the overlap between the generated text and the reference text. Specifically, we used ROUGE-L F1, which focuses on the longest common subsequence (LCS), capturing sequence-level alignment. It evaluates the precision, recall, and F1 score of the generated answers.

- **F1**: scores the harmonic mean of precision and recall at the token level. It provides insights into how well the generated output captures the reference text in terms of word-level accuracy, balancing false positives and false negatives.

- **Exact Match**: calculates the percentage of generated answers that exactly match the reference answers. It is a strict measure of correctness, useful for assessing output alignment in straightforward QA tasks.

- **Human Evaluation**: We conducted human evaluations on 100 randomly selected outputs from each model, including context-injected and baseline versions, to assess performance on a scale from -2 to +2. A score of +2 indicated a perfectly acceptable answer, while -2 reflected completely incorrect or nonsensical outputs. Each dataset was deterministically shuffled, and results were manually reviewed, emphasizing reasoning quality, narrative relevance, answer correctness, and semantic accuracy. We considered the clarity of the narrative, the alignment of answers to questions, and the quality of target values, offering insights into how models performed with and without injected context, particularly for implicit reasoning tasks. The average score across all ratings was used as the final evaluation metric.

4 Results

Below we have both our models' performance during training:

Finetuning a T5 *without* context			
Epoch	**Train Loss**	**Validation Loss**	**BLEURT score**
1	0.170300	0.141289	-0.905466
2	0.148500	0.138184	-0.881379
3	0.142700	0.137338	-0.868918
4	0.139200	0.136923	-0.861303

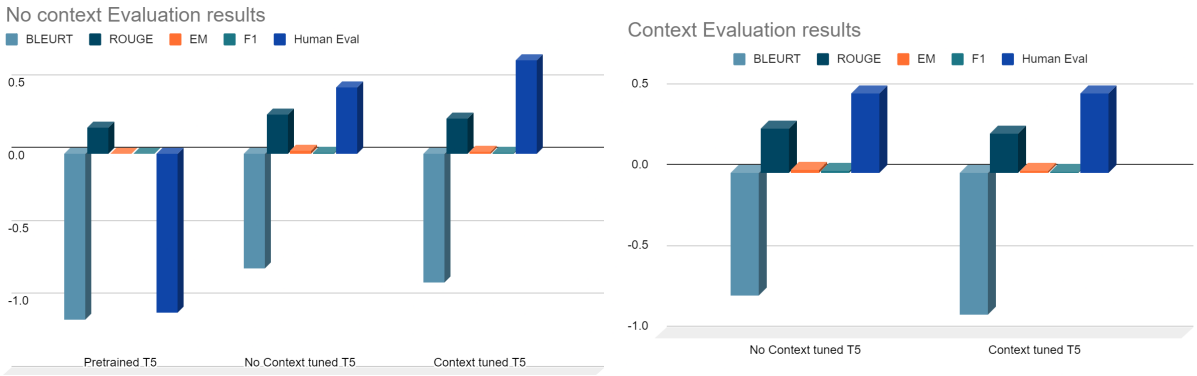
Finetuning a T5 *with* context			
Epoch	**Train Loss**	**Validation Loss**	**BLEURT score**
1	0.166000	0.150395	-0.983019
2	0.156400	0.146696	-0.958194
3	0.147800	0.144871	-0.948719
4	0.146500	0.144614	-0.944824

Below is a chart showing the evaluation of our models along with an untrained T5, to fit everything in the table we will use shorthands for our models and datasets:

T5: denotes a pretrained T5 model, **C-T5**: (Context T5) denotes our T5 model finetuned on data *with* context **NC-T5**: (No Context T5) denotes our T5 model finetuned on data *without* context **C-data**: (Context Data) data whose input contains context **NC-data**: (No Context Data) data whose input *does not* contain context

	BLEURT	ROUGE-L F1	Exact Match	F1 score	Human score [-2, +2]
T5 eval on NC-data	-1.1427	17.55%	0.00%	0.00%	-1.09
NC-T5 eval on NC-data	-0.7853	27.10%	1.59%	0.72%	0.457
C-T5 eval on NC-data	-0.8840	24.37%	1.07%	0.45%	0.643
NC-T5 eval on C-data	-0.7631	26.87%	1.72%	0.78%	0.487
C-T5 eval on C-data	-0.8772	24.02%	1.09%	0.47%	0.491

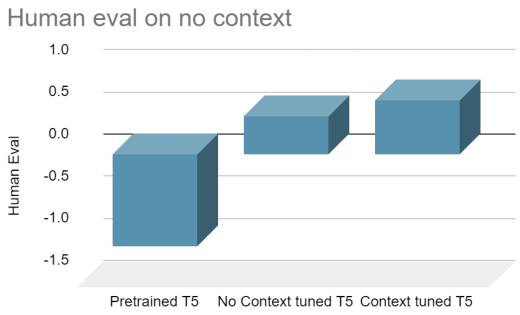
Below are some bar charts showing how these models compared:



(a) Evaluation results from giving models data without context
(b) Evaluation results from giving models data with injected context

5 Analysis and Discussion

Overall our context trained model performed only slightly better in no context situations compared to the model trained without. Our models however fail in a few places. To hypothesize these shortcomings, failure seems prominent when: the answer isn't directly in the text, and when the narrative/context isn't descriptive/comprehensive enough. However it tends to commonly succeed when the answer is nearly



Comparing models given no context, just human evaluation

verbatim in the narrative.

-**Answer not explicit**: An example would be: `"Randy had a home snow-cone maker. It was a hot day and he really wanted a snow-cone. ... he crushed some ice up by hand. Then he ate his snow cone. Question: Why was it a hot day?"` The expected answer was that it was summer, our context trained model outputted ”Randy had a snow-cone maker.” Since the answer was external world knowledge that Gemini happened to not generate, it didn't succeed. (Gemini generated: Snow cones are made of ice. Snow cones are usually shaved. People use ice shavers to shave ice. Randy did not have an ice shaver.) A good example that small language models do poorly in logical deduction and assumption, and how irrelevant context can trap our model.

-**Long narratives**: An example would be: `"The television narrator explained what Congress done this year. I researched Congress accomplishments online and found nothing. I asked everyone if Congress Accomplished anything and got nothing. I called my Congressman and surprisingly he answered the phone. I soon learned Congress had voted themselves a raise this year."` The question was ”Why did I research Congress accomplishments online?”, the expected answer was ”The man on television explained it”, which isn't an effectively labeled answer. We generated ”I found nothing.” The reason we believe our model struggles whenever there is too much narrative and context is simply because it is more input to process. This would exhort more parameters to quickly learn, as we used a T5-small that we could have ideally trained with more data for longer.

-**Verbatim answers**: However our model did very well whenever the answer was verbatim or near verbatim in the narrative. For example: `"I woke up with a really bad sore throat. I called the doctor and... Question: Why did I wake up?"` The answer was ”my throat hurt.”, and we generated ”I had a bad sore throat”, which is an acceptable answer, yet different phrasing. This category might do well since not only is the answer explicit, but formatted nearly exactly the way the model needs to feed it back to the output.

6 Code

Link to Google Colab Notebook used
Link to Google Drive folder with our models, our data as a JSON, README

7 Learning Outcomes

Through this project we learned the complexities of answering ”why” questions, emphasizing the importance of causal reasoning and implicit understanding. Context injection showed moderate improve-

ments in narrative alignment and answer relevance, particularly in cases requiring external knowledge. However, we discovered significant shortcomings in the metrics used. BLEURT and ROUGE-L F1, designed to assess semantic similarity and sequence overlap, failed to accurately reflect reasoning quality and contextual understanding, often misaligning with human evaluations. These metrics performed poorly in the very tasks they were intended for, highlighting the limitations of relying solely on automated evaluation. Human evaluations provided more reliable insights, showcasing the need for robust, domain-specific evaluation strategies. Overall, the project underscored the potential of context augmentation and the critical role of refined evaluation methods for advancing natural language understanding.

8 Contributions

Jaret McManus: *(1)* script for context injection into data and gathering into a file, *(2)* script for running T5 trainers, *(3)* script for setting up Human evaluation spreadsheet

Dane Meister: *(1)* script for linking google drive/recovering TellMeWhy data, *(2)* script for tokenizing dataset, *(3)* script for evaluating models' performance

References

[1] Schank, R. C., & Abelson, R. P. (1975). *Scripts, plans, goals, and understanding: An inquiry into human knowledge structures*. Hillsdale, NJ: Erlbaum.

[2] Pichotta, K., & Mooney, R. J. (2014). *Using narrative events for commonsense reasoning*. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL 2014)*.

[3] Rajpurkar, P., Zhang, J., Lopyrev, K., & 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 (EMNLP 2016)*.

[4] Lal, S., Kumar, S., & Kalyan, A. (2021). *TellMeWhy: A dataset and model for commonsense question answering from stories*. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing (EMNLP 2021)*.