Fine tuning Large Language Models for Entity Relationship Prediction

Badri Narayanan Murali Krishnan, Son Dam, Neha B Chadaga
University of Wisconsin - Madison
{bmuralikrish, sdam, nchadaga}@wisc.edu

Abstract

In this project, we explore the application of generative large language models (LLMs) for relation extraction by fine-tuning a state-of-the-art LLM on the REBEL (Relation Extraction By End-to-end Language Generation) dataset. REBEL, a sequence-to-sequence model, addresses relation extraction by generating relation triplets directly from raw text, effectively transforming the task into a generative process. Our approach involves reimplementing REBEL's experimental setup and evaluating its performance as a baseline. We then compare this to the performance of our fine-tuned LLM, examining its ability to capture relational information with precision and recall. Through this comparison, we aim to identify the strengths and limitations of generative LLMs for structured information extraction, with a focus on contexts that require accurate handling of complex entity relationships. This project will provide insights into the potential of LLMs to advance relation extraction tasks, particularly in domains where high accuracy and multi-relation handling are critical.

1 Overview of Project Idea

With volumes of data exponentially increasing, it is important to convert this large amount of data into valuable insights by establishing semantic relationships between entities present in the text. Our project aims to leverage the REBEL (Relation Extraction By End-to-end Language Generation) model as a foundational framework for relation extraction. REBEL is based on sequence-to-sequence language generation using BART, aimed at simplifying the Relation Extraction process by converting relational facts from text into triplet sequences. Unlike the traditional two-step RE pipelines that extract entities before classifying their relations, REBEL approaches RE as an end-to-end task. Owing to this design, high accuracy and flexibility is achieved across diverse text types and domains, while supporting 200 different relations. Fine-tuned on multiple RE and Relation Classification benchmarks, REBEL demonstrates state-of-the-art performance with reduced training time, adapting well to new domains and tasks with minimal fine-tuning.

Our approach aims to fine-tune a recent state-of-the-art large language model (LLM) on the REBEL dataset. The objective is to compare the performance of a modern LLM in terms of precision, recall, and F1 score with the established REBEL model on relation extraction tasks. This exploration is motivated by the potential of generative LLMs to enhance relation extraction accuracy and handle complex, multi-entity relationships without traditional, multi-step pipelines. We hypothesize that a SOTA LLM, when fine-tuned on this structured task, may outperform or complement REBEL's ability to capture detailed relational data, especially for large, domain-specific datasets. In addition to the baseline implementation, we plan to perform a comparative analysis by reproducing REBEL's reported scores and contrasting them with the results from our fine-tuned model. This will include a fine-grained analysis to assess whether fine-tuning LLMs offers measurable improvements in maintaining high precision and recall for relation extraction. Our analysis will also explore any limitations and areas where the generative capabilities of large LLMs align or deviate from the structured generation approach inherent in REBEL.

The input to the model will be text and the aim is to extract all the possible relationships present in the text in the format <Entity 1, Relationship, Entity 2>. Giving an example, suppose the input text is *"John E Blaha, born in San Antonio on 1942-08-26, worked as a fighter pilot."* The extracted relationship and entities will

be of the form `(John_E_Blaha, birthDate, 1942_08_26) (John_E_Blaha, birthPlace, San_Antonio) (John_E_Blaha, occupation, Fighter_pilot)`.

2 Literature Survey

2.1 Methods for Entity-Relationship extraction

Author Roos M Bakker [1] investigated relation extraction methods to automate knowledge graph construction, on a dataset based on news reports.Different techniques were utilized to assess practical knowledge graph extraction. The study includes methods like Co-occurrences, KeyBERT, REBEL, KnowGL, and generative language models (GPT-3.5 and GPT-4), evaluating performance on both complex and simplified texts. Their study results indicate that while GPT models generally yield higher precision, recall, and F1-scores on simplified texts, models such as REBEL and KnowGL demonstrate strengths in utilizing external knowledge to enrich the extracted graphs. This research highlights the complexity of relation extraction in diverse contexts and the value of integrating generative models with external knowledge for robust knowledge graph extraction [1].

Teon Volkova [2] talks about recent advancements in optimizing Large Language Models (LLMs), which have focused on named entity matching with Monte Carlo Tree Search (MCTS) employed to enhance the performance of the Mistral model in identifying entities within complex datasets. Traditional techniques, including rule-based and probabilistic models, often struggle with scalability and adaptability. Fine-tuning models like these require careful hyperparameter optimization to mitigate overfitting and ensure effective generalization across diverse datasets. Integrating MCTS into the fine-tuning process allows for systematic exploration of the hyperparameter space, resulting in improved precision, recall, and F1-scores compared to conventional approaches. MCTS constructs a search tree where each node represents a possible state, balancing the exploration of novel configurations with the exploitation of previously successful ones. This approach enhances entity matching performance and improves the efficiency of the fine-tuning process by automating hyperparameter selection. Nonetheless, the complexity of the MCTS optimization process may pose challenges regarding interpretability and transparency [2].

2.2 Fine Tuning Optimizations of Large Language Models

Yiming Ju and his team [3] address training imbalances in fine-tuning large language models by merging models trained with different data sequences. The proposed "parameter-selection merging" technique goes beyond traditional weighted-average methods by selectively merging model parameters, thereby diminishing the adverse effects of data order on performance. Experimental results using Llama-2 models and benchmarks like AlpacaEval and GSM8K reveal that parameter-selection merging outperforms weighted-average merging across various tasks, with added resampling further enhancing model accuracy. This merging approach is especially beneficial for larger LLMs, underscoring its potential for efficient task-specific fine-tuning in large-scale applications. Additionally, the study's analysis shows that the observed performance improvements are primarily due to changes in sample positions rather than intra-batch diversity. This method offers a promising avenue for enhancing the fine-tuning process of LLMs, ensuring balanced and robust model performance across varied contexts [3].

Author Renrui Zhang discusses a recent innovation, the LLaMA-Adapter [4], which incorporates only 1.2 million additional parameters by implementing a zero-initialized attention mechanism. This mechanism uses a learnable gating factor to stabilize training, gradually integrating new prompts with preserved pre-trained knowledge. LLaMA-Adapter achieves comparable performance to fully fine-tuned models, significantly reducing training time. Additionally, it supports multi-modal functionality through the integration of an image encoder, enabling effective visual question answering. Its zero-initialized attention has been validated for fine-tuning across various domains, including vision and language models like ViT and RoBERTa, showcasing promising generalizability [4].

3 Re-implementation Details

The baseline and re-implementation were both trained based on pretrained weights of the Bart-Large model. They were both trained on one single epoch.

3.1 REBEL Architecture

The authors of REBEL [5] introduced a novel approach to relation extraction (RE), a crucial NLP task that involves identifying relationships between entities in raw text, such as the connection between people, places, or events. Traditional RE methods use multi-step pipelines that identify entities and classify relations between them. These methods, however, are complex, prone to errors, and often inflexible with limited relation types. REBEL aims to simplify RE by reimagining it as a sequence-to-sequence (seq2seq) generation task, leveraging an autoregressive model based on BART, a transformer designed for text generation tasks.

REBEL's innovation is representing relational triplets (e.g., "Talking Heads, performer, This Must Be the Place") as sequential text, which the model can output directly. The model takes raw text as input, where entities and relationships are extracted and transformed into linearized triplets using special tokens, such as $\langle triplet \rangle$ to mark a new triplet, $\langle subj \rangle$ for the subject, and $\langle obj \rangle$ for the object. This design allows REBEL to handle more than 200 relation types with flexibility, providing robust performance across various RE and relation classification (RC) tasks.

To train REBEL, the authors created a large pretraining dataset using a custom pipeline, which extracts relations from Wikipedia and validates them through a Natural Language Inference (NLI) model, reducing noise. This extensive pre-training enabled REBEL to achieve competitive performance with fewer epochs when fine-tuned on several RE benchmarks, including datasets like CONLL04 and DocRED. Results show that REBEL consistently performs at or above state-of-the-art levels across most tasks, underscoring the strength of its seq2seq design.

The paper also highlights REBEL's efficiency compared to existing methods. In contrast to typical RE models, which require prolonged training due to complex pipelines, REBEL achieves competitive performance in fewer epochs. This efficiency could make it more accessible for tasks where rapid adaptation to new domains or relation types is necessary. The seq2seq model's ability to output relation triplets directly also allows it to generalize well to document-level relation extraction without additional processing steps.

REBEL offers a flexible, efficient alternative for RE, with its seq2seq framework making it a strong candidate for broad NLP applications. Future directions for REBEL include extending the approach to multi-lingual datasets and exploring lighter model versions to improve training efficiency. This research provides valuable insights into how generative models can streamline relation extraction and adapt across varied NLP tasks and data settings.

3.2 REBEL Re-implementation Analysis

We re-implemented the following scripts from the original repository [5].

- generate_samples.py
- pl_data_modules.py
- pl_modules.py
- utils.py

Data Processing Module (pl_data_modules.py) Implements Base PyTorch Lightning Data module for handling dataset operations. This script handles data preprocessing with caching. Custom collation for sequence-to-sequence tasks to enable uniform batching. Support for configurable batch sizes and dataloader parameters was also added.

Model Architecture (pl_modules.py) Implements Base PyTorch Lightning module to handle model operations. Implemented label smoothing loss for training stability along with custom loss function handling as formulated by the original paper. Metrics such as precision, recall, and F1 were also computed. Support for the following optimizer schedulers was also implemented:

- Constant Schedule
- Constant Schedule with Warmup
- Cosine Schedule with Warmup
- Cosine Schedule with Hard Restart and Warmup
- Linear Schedule with Warmup
- Polynomial Decay Schedule with Warmup

Metric	Baseline Scores	Reimplementation Scores
Test F1 Micro Average	66.0052756868	63.4401843704
Test Loss	0.1323998719	0.1323998719
Test Precision Micro Average	68.5170838565	67.7532562357
Test Recall Micro Average	63.6711186138	59.6433754924

Table 1: Comparison of Baseline and Reimplementation Scores

Utility Functions (utils.py) Utils file used for model training and testing. Implemented a Trilinear attention mechanism as formulated by the original paper. Custom BART triplet head for relation extraction, token shifting, and triplet extraction logic by making use of the fundamentals depicted by the baseline paper. Type-aware triplet extraction for different datasets is also supported.

Generation & Sampling (generate_samples.py) Helper file to generate samples used by the architecture. Key features of this file include Batch-wise text generation, logging generation results to Weights & Biases. Support for handling special tokens and padding was also implemented.

4 Potential Methods

While there exist various experiments on relationship-entity extraction using models like pre-trained and fine-tuned GPT-3.5, pre-trained GPT-4, and even Llama 3, there has not been a study on fine-tuning the REBEL model by utilizing Llama 3.2 for the same use case. Hence, we would like to explore the possibility of implementing such a method by building upon the REBEL framework and incorporating the Llama 3.2 model. Few of the potential ways this can be achieved are as follows:

4.1 Enhanced Fine-Tuning with Curriculum Learning

To improve model convergence and robustness, curriculum learning could be explored as a fine-tuning strategy. Curriculum learning involves gradually increasing the complexity of the training data presented to the model. For example, we could start by training LLAMA 3.2 on simpler relation instances in the REBEL dataset (e.g., single-relation sentences) and progressively introduce more complex instances with multiple relations. This approach is expected to help the model build foundational patterns in simpler cases before tackling more challenging examples, which may result in better overall performance and generalization.

4.2 Multi-task Learning with Named Entity Recognition (NER)

LLAMA 3.2 could also be fine-tuned in a multi-task learning setting, where it learns both relation extraction and named entity recognition (NER) tasks simultaneously. By sharing representations between these tasks, the model's understanding of entities and their relationships might improve. An auxiliary NER objective can be integrated during training, which could potentially make the model more resilient to noisy or ambiguous inputs and result in improved precision and recall on relation extraction.

4.3 Knowledge Distillation for Lightweight Models

Given the large size of LLAMA 3.2, deploying the fine-tuned model in resource-constrained environments could be challenging. To address this, the usage of knowledge distillation can be proposed, to create a smaller, efficient model without significantly compromising performance. In this setup, the fine-tuned LLAMA 3.2 model would serve as the "teacher," and a smaller transformer-based model, such as DistilBERT or TinyBERT, would act as the "student." The student model would be trained to replicate the predictions of the teacher model, thereby inheriting its performance capabilities while requiring fewer computational resources. This distilled model could enable faster inference and easier deployment in real-world applications.

4.4 Data Augmentation with Synthetic Relation Triples

The REBEL dataset, though effective, may lack the diversity needed to capture rare or complex relations. To address this, the dataset can be augmented

with synthetic data generated through data augmentation techniques, such as back-translation and paraphrasing. Additionally, we can explore using an external LLM (e.g., GPT-4) to generate new relation triples based on existing patterns, thereby expanding the variety of relations and entities within the dataset. These synthetic examples would help LLAMA 3.2 generalize better, especially in scenarios where it encounters less frequent or out-of-domain relations during testing.

4.5 Adversarial Training for Improved Robustness

Relation extraction models are often vulnerable to adversarial examples or small perturbations in the input text. To make the model more robust, we can use adversarial training techniques during fine-tuning. Specifically, we could introduce minor changes to entity names, sentence structure, or phrasing in the REBEL dataset and ensure that LLAMA 3.2 remains accurate under these perturbations. This could make the model more resilient to input variations and enhance its generalization capability, particularly in real-world scenarios where entities and relations may not follow a rigid structure.

Overall, these proposed methods extend beyond the traditional training of the REBEL model on the REBEL dataset. By experimenting with curriculum learning, multi-task learning, knowledge distillation, data augmentation and adversarial training, we aim to push the boundaries of relation extraction performance. These methods, individually or in combination, could reveal valuable insights into how large language models like LLAMA 3.2 can be leveraged for complex information extraction tasks, and they may provide a foundation for future advancements in generative relation extraction models.

5 Potential Data Sets and Experimental Settings

5.1 Datasets

The REBEL project includes an end-to-end relation extraction dataset that was used to pre-train an initial REBEL model based on the Bart-Large model. This dataset was created by using wikiextractor to extract entities from Wikipedia abstracts, then using wikimapper to link these entities to Wikidata entities, and finally extracting all relations present between those entities. Since not all Wikidata relations are entailed within the text document, the authors of REBEL used RoBERTa to filter out relations not entailed in the document [5].

Each split (train, val, test) is stored in a jsonl file, in which each line contains a text document, the entities within the document, and triplets indicating the relationships between entities [5]. Because the original dataset is large, we reduced the train and test data to about 10% of the original sizes to run on our limited compute resources. We left the validation set as is because the validation process during training did not take too long. The training set consists of 300,000 documents, the validation set consists of 152,673 documents, and the test set consists of 15,000 documents.

Aside from the REBEL dataset, there are a few other datasets commonly used in relation extraction and relation classification tasks, and were also used in the REBEL project by fine-tuning based on the model pre-trained on the REBEL dataset. DocRED is similar to the REBEL dataset in that it is based on Wikipedia and Wikidata, but it focuses on longer spans of text. The CONNL04 and NYT-multi datasets are based on news articles, and the ADE dataset is about adverse effects from drugs. Re-TACRED is a relation classification dataset [5]. We will focus our studies on the REBEL dataset, but if we were to use these datasets in our study, we will have to pre-process them differently from the REBEL dataset since they have different formats.

5.2 Experimental Settings

For our baseline and reimplementation, we used the settings used in the REBEL repository. We used a batch size of 4 for training and 8 for evaluation, as this was the largest we could use to run on a RTX 2080Ti with 11GB of memory. For both the baseline and reimplementation, we started with the pretrained Facebook's Bart-Large model on HuggingFace and fine-tuned it for one epoch, which is equivalent to 10,362 steps. The REBEL project fine-tuned Bart-Large for six epochs on the REBEL dataset, but we only trained one epoch due to our limited compute resources.

The REBEL codebase requires many specific versions of Python packages, but some of them had dependency conflicts. Thus, for some packages we removed the version requirement and manually installed relevant dependencies according to any error messages displayed.

Initially, we trained and evaluated our model on a RTX 2080Ti with 11GB of memory. One epoch

of training took about five hours, and testing took about one hour. However, at one point all of the 2080Ti's were used by other users, so we trained on a A40 with 48GB of memory which also had other tasks running. One epoch of training took about four hours, and testing took about one hour.

5.3 Experiments to be Conducted

Our next experiment is to fine-tune the LLAMA 3.2 model with 8B parameters on the REBEL dataset, similar to how BART is fine-tuned in the baseline and reimplementation. First, we will build a pipeline to process the REBEL dataset into LLAMA similar to how it is processed into BART in the baseline. Then we will preprocess the data according to the LLAMA inputs. Then we will fine-tune LLAMA on several epochs and evaluate it on the test set. There are several methods for fine-tuning LLAMA, which we discussed in Section 4. Once we evaluate our fine-tuned model, we will perform a detailed analysis of how it compares to REBEL and report on areas for future improvement. Further details about experiments to be conducted can be found in Section 6.

6 Plan of Activities

Given that the primary goal of this project is to fine-tune the REBEL model on LLAMA 3.2, we have outlined the following tasks and divided the work as follows:

6.1 Data Preprocessing - Neha Chadaga

In this stage, we will focus on preparing the REBEL dataset for fine-tuning on LLAMA 3.2. This involves cleaning the data, removing any inconsistencies, and organizing the relation triplets in a structured format suitable for input into the LLM. Additional tasks include tokenization and ensuring that all entities are properly tagged. We will also assess the need for any data augmentation techniques to expand the dataset, potentially improving the model's ability to generalize.

6.2 Fine-tuning Setup - Badri Narayanan

Setting up the fine-tuning environment includes configuring LLAMA 3.2 to handle relation extraction tasks based on REBEL's structure. We will specify the model architecture, hyperparameters, and training configuration, drawing on best practices outlined in recent studies. The setup process will involve initializing LLAMA 3.2 with pre-trained weights, configuring hardware accelerations (like GPUs or TPUs), and installing necessary dependencies. This stage will also include setting checkpoints for tracking performance over multiple epochs.

6.3 Fine-tuning LLM - Son Dam

The fine-tuning process involves fine-tuning LLAMA 3.2 on the REBEL dataset with appropriate hyperparameters for optimal performance. This step will involve monitoring training metrics of loss, accuracy, precision and recall while adjusting parameters like learning rate, batch size, and dropout as necessary. We will use validation sets to avoid overfitting and to ensure the model is generalizing well. Given the computational intensity of training LLMs, we will use 10% of the original sizes to train and test the data.

6.4 Fine-tuning Evaluation - Badri, Son and Neha

After training, we will evaluate the performance of the fine-tuned LLAMA 3.2 model on a test set from the REBEL dataset. Evaluation metrics will include F1 score, precision, recall, and loss, enabling us to directly compare the fine-tuned model's performance with the baseline REBEL model. Additionally, we will conduct a qualitative analysis of the relation extraction output to understand areas where the model performs well and where it may struggle, especially in handling complex multi-relation contexts.

6.5 Final Report Writing - Badri, Son and Neha

The final report will document all stages of the project, including the methodology, experimental setup, and detailed analysis of the results. It will also cover the limitations encountered during the project and insights gained from the fine-tuning process.

6.6 Presentation Preparation - Badri, Son and Neha

In the final phase, we will prepare a presentation to summarize out project to enable us to present our methodology and findings in class.

Table 2: Timeline for the project

Task	Week 1	Week 2	Week 3	Week 4	Week 5	Week 6	Week 7	Week 8
Problem Statement Definition								
Literature Survey								
Environment Setup								
Baseline Result Replication								
Project Proposal								
Data Preprocessing								
Fine-tuning Setup								
Fine-tuning LLM								
Evaluation of Fine-tuned LLM								
Project Report								
Project Presentation								

References

[1] Roos M Bakker and Daan L Di Scala. “From Text to Knowledge Graph: Comparing Relation Extraction Methods in a Practical Context”. In: *First International Workshop on Generative Neuro-Symbolic AI, co-located with ESWC*. 2024.

[2] Teon Volkova, Evander Delacruz, and Thaddeus Cavanaugh. “A novel approach to optimize large language models for named entity matching with monte carlo tree search”. In: *Authorea Preprints* (2024).

[3] Yiming Ju et al. “Mitigating Training Imbalance in LLM Fine-Tuning via Selective Parameter Merging”. In: *arXiv preprint arXiv:2410.03743* (2024).

[4] Renrui Zhang et al. “LLaMA-adapter: Efficient fine-tuning of large language models with zero-initialized attention”. In: *The Twelfth International Conference on Learning Representations*. 2024.

[5] 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*. 2021, pp. 2370–2381.