Fact-Checking Book Claims: A Dual-Phase Approach with Enhanced Retrieval and Entailment Models

Jun Hu
Cornell University
Ithaca, NY
jh2829@cornell.edu

Abstract

This report presents our work on creating and evaluating datasets for training retriever models that, given a claim about the book, retrieve the relevant chunk from the same book. We explore the effectiveness of different retriever models and the impact of retrieval fine-tuning on entailment performance.

1 Introduction

The goal of our work is to develop an effective method for fact-checking claims about books. Since entire books are difficult to fit into a context window, we envision a two-step fact-checking pipeline: retrieval followed by entailment. First, the retriever model identifies and extracts the relevant chunks from the book based on the claim, where each claim is extracted from a corpus of book summaries generated by language models. Then, an entailment model assesses whether the retrieved chunk supports or refutes the given claim. This approach aims to enhance the performance of entailment models by improving the accuracy of retriever models. The approach we decided to take is to fine-tune the Contriever model. If we can demonstrate that our approach to fine-tune the retriever model works for Contriever and we can automate this training approach, it will become the de-facto method for training other retrieval models, thereby enhancing the entailment model's ability to correctly identify whether a claim is supported or not.

2 Notation

Our goal is to fine-tune existing retrieval models to improve performance on selected domains. In our experiments, we focus on the narrative books.

Our retrieval dataset consists of a corpus of documents $\mathcal{D} = \{d_1, d_2, \ldots, d_n\}$. Each document d_i is associated with a single summary chunk s_i. Each summary chunk s_i contains multiple claims, represented as $\mathcal{C}_i = \{c_{i1}, c_{i2}, \ldots, c_{im}\}$, where each claim c_{ij} is a sentence in the summary chunk s_i.

Given a claim c_{ij}, the goal of the retrieval task is to select the most relevant document chunk d_i such that d_i contains the information supporting or refuting c_{ij}.

Recall is calculated based on whether the document chunk selected by the retriever is actually the correct one, i.e., if it corresponds to the same document chunk that c_{ij} is mapped to.

3 Retrieval Module

In this section, we will describe our experiments with the retrieval module. First, we will discuss the dataset we use. It is constructed using hierarchical summaries from existing work, described at section 3.1. Then, in section 3.3, we will discuss results using both the existing retrieval models and improvements after fine-tuning.

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

3.1 Dataset Creation

The original dataset [1] breaks down a book into many text documents, where each document consists of fewer than 512 words. Each document is then passed into GPT-3 to generate summaries. These lower-level summaries are combined into batches and used to generate a smaller corpus of higher-level summaries, where there is a one-to-many relationship between higher-level and lower-level summaries.

To create the dataset for training and evaluating retriever models, we break down each lower-level summary into sentence claims and map them to their respective document text. We build another higher-level claim-to-document dataset by taking a single higher-level sentence claim and finding the lower-level summary chunk with the highest 2-gram overlap among all its corresponding lower-level summary chunks. We do this for each higher-level claim and map it to the document text that the found lower-level summary maps to.

For dataset statistics, see Section 1.

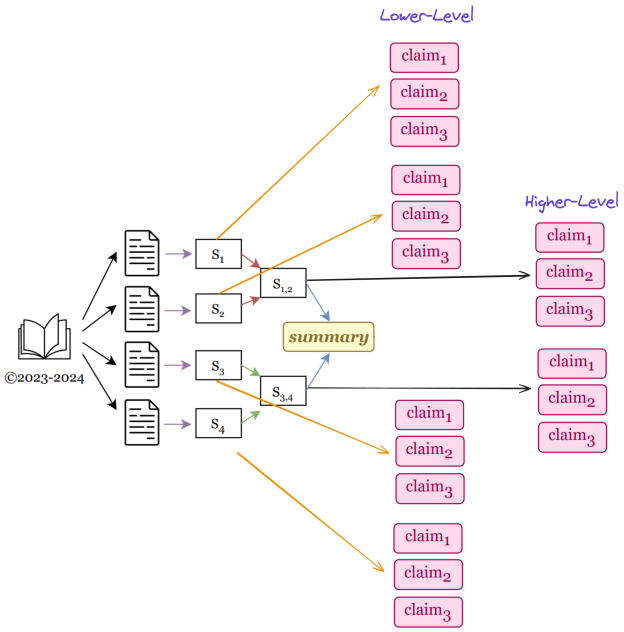

Figure 1: The above details the process of extracting higher-level and lower-level claims from book text. This image is modified from [3].

3.2 Fine-tuning

We use the contrastive loss objective to train our model (discussed next). We perform some hyperparameter tuning to choose the best hyperparameters to train our models

Loss Function In our learning setup, we use a contrastive objective with label smoothing, that pushes the context vector $\mathbf{c_i}$ close to its document vector $\mathbf{d_i}$, but away from all other document vectors $\mathbf{d_j}$ in the same minibatch ("in-batch negative sampling"). The loss is computed over a batch of claim and document embeddings.

Let $\mathcal{B}$ represent a batch of size b. We strategically design the batches using a custom batch sampler, ensuring that each batch contains claims and documents from the same book. This approach increases the task's difficulty, preventing the model from relying solely on entity matching or word overlaps, thereby promoting a deeper learning process. Additionally, we ensure that each document d_i within a batch $\mathcal{B} = \{d_1, d_2, \ldots, d_b\}$ is unique and appears only once in the batch, ensuring that it maps

[1] For the specific data, please refer to `https://openaipublic.blob.core.windows.net/recursive-book-summ/website/index.html?data_id=175b%2F0&dataset=booksum#/booksum`

exclusively to a single claim c_i. For each claim c_i and its corresponding document d_i within the batch, we perform the following steps to compute the loss.

We compute the score matrix $\mathbf{S}$ where each element s_{ij} is the dot product between the embeddings of the i-th claim and j-th document:

$$s_{ij} = \mathbf{c}_i \cdot \mathbf{d}_j$$

We then apply the softmax function to the score matrix to obtain the predicted probabilities $\hat{y}_{ij}$:

$$\hat{y}_{ij} = \frac{e^{s_{ij}}}{\sum_{k=1}^{b} e^{s_{ik}}}$$

The true label distribution y_{ij} is adjusted using label smoothing. For a smoothing parameter α, the label distribution is defined as:

$$y_{ij} = \begin{cases} 1 - \alpha + \frac{\alpha}{b} & \text{if } j = i \\ \frac{\alpha}{b} & \text{if } j \neq i \end{cases}$$

The cross-entropy loss with label smoothing is then calculated as:

$$\mathcal{L} = -\frac{1}{b} \sum_{i=1}^{b} \sum_{j=1}^{b} y_{ij} \log(\hat{y}_{ij})$$

This loss function encourages the model to produce higher scores for matching claim-document pairs while penalizing incorrect pairings, with label smoothing helping to prevent overconfidence in predictions.

3.2.1 Hyperparameters

The fine-tuning process involves several hyperparameters that were carefully chosen to optimize model performance:

- **Learning Rate (lr):** The learning rate was set to 5×10^{-5}. This controls the step size in the gradient descent optimization.
- **Batch Size:** We used a batch size of 32. This defines the number of samples that are processed before the model's internal parameters are updated.
- **Epochs:** The model was trained for 10 epochs. An epoch is a full pass through the entire training dataset.
- **Temperature (T):** The temperature parameter was set to 1.0, which scales the logits before applying the softmax function.
- **Label Smoothing:** Label smoothing was applied with a smoothing factor of 0.1 to prevent the model from becoming overconfident in its predictions.
- **Warmup Steps:** The learning rate was linearly increased for the first 2000 steps to stabilize training.

3.2.2 Optimization and Scheduling

The optimizer used was AdamW, which is a variant of the Adam optimizer with weight decay regularization to prevent overfitting. The learning rate was adjusted using a linear scheduler with a warm-up phase. The total number of training steps was calculated as total_steps $=$ len(train_loader)$\times$ epochs.

3.2.3 Training and Evaluation

During training, the model's parameters were updated using the optimizer and the learning rate scheduler. The training loss was logged at regular intervals. After each epoch, the model was evaluated on the validation set, and the validation loss was computed to monitor the model's performance and prevent overfitting.

3.3 Results

3.3.1 Experiment Details

Type	Claim Level	Total Claims	Total Books	Average Number of Claims per Book
Train	Lower-Level	174316	235	742
Validation	Lower-Level	21789	17	1282
Test	Lower-Level	21791	17	1282
Test	Higher-Level	2160	16	135

Table 1: Claims and Books Statistics

In our experiments, we evaluate the effectiveness of three different retriever models: BM25[4], Contriever[1], and Dense Passage Retrieval (DPR)[2].

To evaluate the performance of these retriever models, we employ the following metrics:

- **Recall@K**: This metric measures the proportion of relevant documents retrieved within the top K results. It is a standard evaluation metric in information retrieval, providing insights into how effectively the model retrieves relevant information. Specifically, we will report Recall@1 to Recall@10.
- **Average Number of Top 10 Results in The Same Book**: In addition to Recall@K, we report the average number of retrieved text documents, out of the top 10, that come from the same book as the claim.

By comparing the performance of BM25, Contriever, and DPR using these metrics, we aim to identify the most effective retriever model for our fact-checking pipeline. This evaluation will guide us in selecting the optimal retriever to enhance the overall performance of our fact-checking approach.

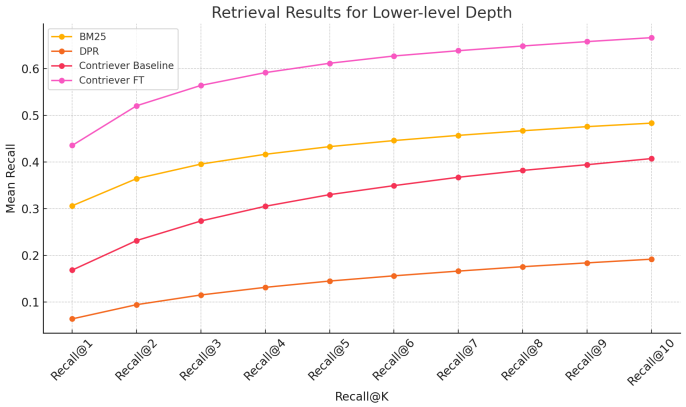

Figure 2: Recall@K results for granular, lower-level claims.

3.3.2 Performance of base models

First, we study the performance of base retrieval models without fine-tuning on our narrative dataset. Our results are shown in Fig 2 and Fig 3. At the lower-level depth, BM25 consistently outperforms

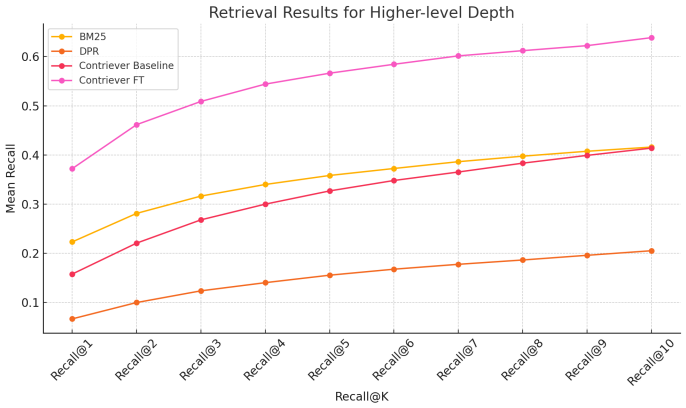

Figure 3: Recall@K results for higher-level, broader claims.

both Contriever and DPR across all recall levels. Specifically, BM25 achieves a Recall@10 of 48.30%, indicating that it retrieves relevant chunks more effectively than the other models. Contriever follows, with a Recall@10 of 41.60%, while DPR trails significantly with a Recall@10 of 20.53%. This pattern suggests that the Contriever Baseline and BM25 are better suited for retrieving relevant information in this setting, with BM25 having a slight edge. The results at the higher-level depth show a similar trend, where Contriever and BM25 perform similarly, and DPR lags significantly behind. This highlights the challenge DPR faces in retrieving relevant documents in our context.

- **BM25**: As a traditional probabilistic retrieval model, BM25 shows relatively robust performance in both higher-level and lower-level depths. This is expected, however, as its reliance on term frequency and inverse document frequency allows it to correctly map claims containing book-specific and document-specific terms to documents.
- **DPR**: Although DPR utilizes an advanced neural architecture with dense vector representations, its recall rates are significantly lower in our retrieval setting compared to BM25 and the Contriever Baseline. This disparity can be attributed to the nature of the training dataset used for DPR, specifically the Natural Questions (NQ) dataset[2]. The NQ dataset consists predominantly of short, factual queries mapped to concise, specific passages. In contrast, our task involves retrieving relevant book text given summary sentences, which are often abstract and require understanding of broader contextual relationships. DPR, being trained on factual and straightforward question-answer pairs, may struggle to capture the subtleties and semantic richness needed for effective retrieval in the context of literary summaries and book passages, leading to its lower performance in our experiments.
- **Contriever Baseline**: Contriever Baseline consistently outperforms DPR and is comparable to BM25, indicating its ability to capture deeper semantic relationships between queries and documents. Its neural network architecture seems to provide a better understanding of the context, leading to higher recall rates.

3.3.3 Performance of Fine-Tuned Models

After fine-tuning the Contriever model using our lower-level claim dataset, we observe notable improvements in the model's performance across both lower-level and higher-level claims as shown in Fig 2 and Fig 3.

For lower-level depth, where the fine-tuning was directly applied, the improvements are substantial. The fine-tuned Contriever model achieves a Recall@10 of 66.61%, a significant increase from the 40.72% recall rate of the Contriever Baseline. The fine-tuned model better understands the specific nuances and context of the claims, leading to more accurate retrievals.

Interestingly, the fine-tuning also positively impacts the performance at the higher-level depth, even though the fine-tuning was not directly targeted at this depth. The fine-tuned Contriever achieves a

[2] `https://huggingface.co/datasets/google-research-datasets/nq_open`

Recall@10 of 63.84% at the higher-level depth, compared to the 41.39% of the Contriever Baseline. This improvement suggests that the benefits of fine-tuning extend beyond the specific dataset used for training. The fine-tuned model seems to have developed a more generalized understanding that enhances its performance across different depths.

Overall, we have shown that automatic fine-tuning of the Contriever model via the use of summarization models can greatly improve retrieval performance in the narrative-book domain; thus, it holds the potential to become the de facto method for training other retrieval models. Coupled with entailment models, which we will explore in the next section, this standardization would greatly enhance the entailment model's ability to accurately determine whether a claim is supported or refuted. This methodological advancement would not only improve retrieval performance but also streamline the development of more sophisticated and reliable fact-checking systems.

4 Retriever-Entailment Pipeline

Now that we have improved our retrieval models, we next wanted to test whether these models can help improve the performance of fact-checking pipelines.

We developed a retrieval-then-entailment pipeline to assess the effectiveness of various retriever models. For the entailment component, we leveraged BART [3], a transformer-based sequence-to-sequence model with 407 million parameters. BART is pre-trained as a denoising autoencoder, enabling it to excel in a diverse array of natural language processing tasks, such as text generation and classification. Specifically, we employed the version of BART that has been fine-tuned on the Multi-Genre Natural Language Inference (MNLI) dataset, which has demonstrated robust performance in discerning relationships between sentences, making it highly suitable for entailment tasks in our pipeline.

The pipeline was executed using the following process:

```
result = self.classifier(f"{truncated_premise}{self.tokenizer.sep_token}{claim}")
```

where *truncated_premise* is the retrieved text document from the retriever.

4.1 Dataset

Our datasest in Section 3.1 only gives us access to presumably correct claims. However , we need both correct and incorrect claims to test whether the pipeleine can distinguish between the two.

The testing dataset for retriever-entailment pipelines consists of:

- **Positive Data**: Taken from depth-3 retrieval data, where each summary sentence is taken as the claim, and the entailment result is *Entailment*.
- **Negative Data**: Generated by taking a claim and document pair, and passing the document into another BART model [4], that generates the negative version of the document. The entailment result is either *Contradiction* or *Neutral*.

4.2 Results and Discussion

In this section, we analyze the performance of different retrieval models within our retrieval-then-entailment pipeline. The models evaluated include DPR, Contriever, BM25, and a fine-tuned version of Contriever. The evaluation metrics include overall accuracy and true accuracy (correct predictions when the gold passage is retrieved and used to get the correct entailment).

The results clearly demonstrate that a stronger retriever model directly leads to a much more effective entailment pipeline. The fine-tuned Contriever model, in particular, not only excels in retrieving the most relevant passages but also significantly enhances the accuracy of the entailment process. This improvement in retrieval quality allows the entailment model to work with more precise and contextually appropriate information, thereby boosting its ability to make correct predictions.

[3] `https://huggingface.co/facebook/bart-large-mnli`

[4] `https://huggingface.co/minwhoo/bart-base-negative-claim-generation`

Model	Overall Accuracy (%)	True Accuracy (%)
DPR	43.32	13.24
Contriever	62.68	30.87
BM25	66.25	34.09
Contriever Fine-Tuned	73.94	64.15

Table 2: Overall and True Accuracy for Different Models

Our experiments reveal a strong correlation between the performance of the retriever and the overall success of the entailment model. While pre-trained neural retrievers like DPR show potential, they often falter in complex tasks that require understanding abstract summary sentences and matching them with relevant book text. On the other hand, traditional methods like BM25 and advanced models like Contriever, especially when fine-tuned, offer superior performance. The fine-tuned Contriever's consistent outperformance across all metrics underscores the importance of adapting the retriever model specifically to the task at hand.

Moreover, with the improved retrieval performance, there is considerable potential to introduce more sophisticated entailment models, such as GPT-4. Due to resource constraints, our current work employs BART, which, even when provided with the correct document, achieves an accuracy of only 72.78

5 Conclusion

This work demonstrates that training retriever models using generated summaries from summarization models can significantly enhance retrieval performance, surpassing baseline methods. The improvements observed in both lower-level and higher-level retrieval tasks underscore the potential of fine-tuning retrieval models, particularly in complex domains like narrative books. By refining the retriever's ability to accurately match claims with relevant text, the overall accuracy and effectiveness of fact-checking pipelines are markedly improved. These findings suggest that our approach can be extended as a standard practice for training retrieval models, ultimately bolstering the performance of entailment models in distinguishing whether claims are supported or refuted. This advancement is a critical step toward the development of more reliable and sophisticated fact-checking systems.

References

[1] Gautier Izacard, Mathilde Caron, Lucas Hosseini, Sebastian Riedel, Piotr Bojanowski, Armand Joulin, and Edouard Grave. Unsupervised dense information retrieval with contrastive learning. *Transactions on Machine Learning Research*, 2022. Reviewed on OpenReview: `https://openreview.net/forum?id=jKN1pXi7b0`.

[2] Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6769–6781, Online, November 2020. Association for Computational Linguistics.

[3] Yekyung Kim, Yapei Chang, Marzena Karpinska, Aparna Garimella, Varun Manjunatha, Kyle Lo, Tanya Goyal, and Mohit Iyyer. Fables: Evaluating faithfulness and content selection in book-length summarization, 2024. UMass Amherst, Adobe, Allen Institute for AI, Princeton.

[4] Stephen E. Robertson and Hugo Zaragoza. The probabilistic relevance framework: Bm25 and beyond. `https://www.researchgate.net/publication/220613776_The_Probabilistic_Relevance_Framework_BM25_and_Beyond`, 2009. Accessed: 2024-08-08.

A Appendix / supplemental material

Table 3: Retrieval Results

Retriever	Depth	Avg Top 10 in Same Book	Recall@1	Recall@2	Recall@3	Recall@4	Recall@5	Recall@6	Recall@7	Recall@8	Recall@9	Recall@10
BM25	2	4.3542	0.2233	0.2812	0.3164	0.3400	0.3583	0.3725	0.3863	0.3975	0.4075	0.4160
DPR	2	2.7517	0.0668	0.1000	0.1238	0.1404	0.1556	0.1678	0.1777	0.1865	0.1960	0.2053
Contriever Baseline	2	5.9019	0.1579	0.2208	0.2681	0.3000	0.3269	0.3481	0.3653	0.3833	0.3991	0.4139
Contriever FT	2	4.9583	0.3722	0.4616	0.5088	0.5440	0.5662	0.5843	0.6014	0.6120	0.6222	0.6384
BM25	3	3.5716	0.3062	0.3643	0.3954	0.4164	0.4328	0.4458	0.4569	0.4668	0.4756	0.4831
DPR	3	2.3765	0.0640	0.0944	0.1152	0.1315	0.1450	0.1562	0.1663	0.1757	0.1839	0.1918
Contriever Baseline	3	4.9333	0.1685	0.2317	0.2736	0.3051	0.3300	0.3493	0.3672	0.3819	0.3941	0.4072
Contriever FT	3	4.2865	0.4354	0.5203	0.5639	0.5913	0.6113	0.6269	0.6382	0.6483	0.6577	0.6661