CS 728: Fact Extraction and VERification (FEVER)

Vaibhav Singh

April 2024

1 Retriever Architecture

Due to computation limitations on learning/indexing the corpus on sentence/line granularity, we explore the options for a two-stage retriever. The first retriever scores individual documents when given a claim, and the second retriever scores all sentences within the top retrieved documents.

1.1 Stage 1: Sparse Retrieval

For sparse retrieval, we use **Lucene-based indexer and searcher** through the *pyserini* library. The index is built at the document level using the *'text'* field of every document in the wiki-pages jsonl files.

Additionally, we incorporate **named-entity constraint** during index search. We add an additional field of NER tags to every document json as a preprocessing step before indexing. We use a `bert-base` token-classification model[1] to find named entities within the wiki document texts as well as the claims.

At inference time, given a claim and the named entities in the claim, we retrieve top-k documents by probing the sparse index with the named entity constraint. We set k=2 for our results.

1.2 Stage 2: Dense Retrieval

1.2.1 `DPRContextEncoder` and `DPRQuestionEncoder`

The sparse retriever scores documents from which we select the top k documents. We pass each sentence from these top k scoring documents to the second stage retrieval which uses Dense Passage Retrieval (DPR). We use the DPR model from [1] finetuned for retrieval on 4 different QA datasets, see paper for more details. The DPR model contains three BERT modules, namely, the context encoder, the question encoder, and the reader model. The context encoder encodes the passages, the question encoder encodes the query and the reader

[1] `https://huggingface.co/dslim/bert-base-NER`

model utilizes the retrieved passages to answer the question. We utilize the context encoder to encode evidence and the question encoder to encode the claim. We project the output hidden state of size 768 to 128 before computing the dot product between claim and evidence embeddings. We apply sigmoid on top of the dot product to bring the scores between 0 to 1.

We finetune the question and context encoder by sampling all sentences from relevant documents and we add all sentences from one document which is top-scoring but not relevant. We finetune these modules using binary cross-entropy loss.

1.2.2 `bert-base` Encoder

We also explore a case where both claims and evidences are encoded using the same encoder for dense retrieval. We use `bert-base-uncased` checkpoint and use the *pooler-output*, followed by a linear layer with sigmoid activation to train the model to score the extent to which an evidence entails a given claim. The inputs are fed as: [CLS] *evidence* [SEP] *claim.*

At inference time, we use the *pooler-output* of the BERT encoder layer to represent the claim and evidence in a (1, 768) dimension dense vector.

To train the model, we use all the gold evidence sentences that correspond to a claim as individual training instances. For every evidence, we randomly sample 5 claims as negative samples, where the evidence does not entail the claim. A total of 1,320,072 training instances are generated and used to train the BERT model. At inference time, all the sentences from the documents retrived from the sparse retrieval phase are ranked for relevance based on cosine similarity with the claim and top five evidences are selected for NLI.

2 NLI Architecture

We contrast between an Encoder-only model `BERT` and a Decoder-only model `GPT2`.

2.1 GPT2ForSequenceClassification

Given the nature of the fact verification task, we explore using a Decoder-only model, which is capable of processing longer contexts. We use GPT -2 for sequence classification, which is the GPT2 model with a classifier head that uses the hidden state of [CLS] token to predict the class label. We finetune the GPT2 model by using gold evidence for the ‘SUPPORTS‘ and ‘REFUTES‘ labels and we retrieve top-scoring sentences using our two-stage retriever for the ‘NEI‘ label. We concatenate the claim and evidence in the following manner,

$$\text{[CLS]}\langle\text{claim\_token}\rangle\text{claim}\langle\text{evidence\_token}\rangle\text{evidence1}\langle\text{evidence\_token}\rangle\text{evidence2[SEP]}$$

2.2 BertForSequenceClassification

Here, a `bert-base-uncased` sequence classification model is trained to output one of **SUPPORTS, REFUTES or NOT ENOUGH INFO** given an input sequence of the form: [CLS] *claim* [SEP] *evidence*. Here the top five evidence sentences ranked from the retriever are concatenated together to form a single evidence candidate spanning multiple documents.

During training, we use all train.jsonl evidence sentences corresponding to every claim as individual training instances. However, in the case of **NOT ENOUGH INFO**, we sample a random sentence from the wikipedia dump to simulate the case where the evidence does not have any information about the claim.

3 Results and Discussion

We evaluate the first stage retriever (BM25-based) on gold evidence from the validation set for the 'SUPPORTS' and 'REFUTES' labels.

Table 1: Improved MRR with the use of a Sparse Retriever indexed on named entities along with the document contents. The boolean query formulation for searching the index leverages named entities extracted from claims.

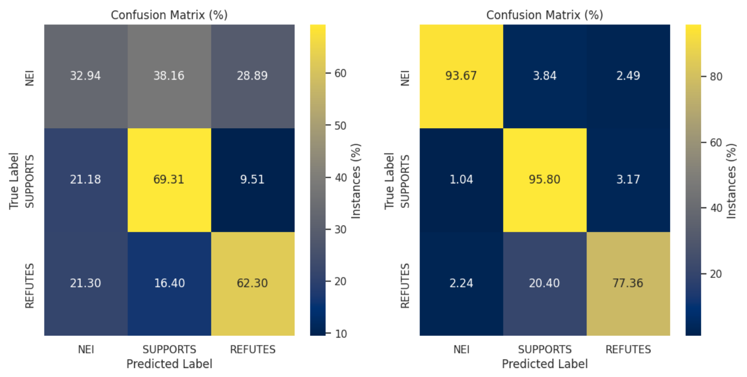
Metric	**MRR@1**	**MRR@10**	**MRR@100**
With NER	0.5342	0.6515	0.6557
Without NER	0.0009	0.0011	0.0011

Due to the improvement in retrieving relevant documents, the NLI model performs even better.

Table 2: We evaluate the validation set using the fever-scorer script. We observe an improvement in label accuracy by **10.29%** and evidence F1 by **15.29%** using NER indices.

Metric	**Without NER**	**With NER**
Strict Score	0.0811	0.3891
Label Accuracy	0.5024	0.6053
F1 Score	0.0572	0.2101

We finetune GPT2 with gold evidence for the 'SUPPORTS' and 'REFUTES' labels, and we retrieve evidence for the 'NEI' label. We suspect that the NLI model is learning some pattern among the retrieved evidence for the 'NEI' label, which results in maximum confusion, as seen in figure 1 when predicting a claim with the 'NEI' label. This pattern among retrieved evidence when observed for a claim with label 'SUPPORTS' or 'REFUTES', the model predicts 'NEI'.



(a) With retrieved evidence (b) With gold evidence

Figure 1: Confusion matrix on NLI model predictions on the validation set.

References

[1] Vladimir Karpukhin, Barlas Oğuz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen tau Yih. Dense passage retrieval for open-domain question answering, 2020.