COMP90042 Project: Retrieval and Classification

Student ID: 1001087 **Student ID: 1166507** **Student ID: 1254859**

Abstract

Based on the retrieval and classification tasks of the evidence corpus, this project explored the architectural design of the retrieval and classification models and their impact on the overall performance.Through systematic evaluation on the validation set, we compared and analyzed the performance of various methods in terms of accuracy, recall and F1-score indicators, providing a practical reference for model architecture design. Finally, the project achieved a classification accuracy of 0.47 on the test set and an F1-score of 0.14 in the retrieval task.

1 Introduction

With the rapid development of large language models, building a high-performance model structure has become an important research direction. This project is based on the retrieval and classification between Claim and Evidence Corpus to determine whether each claim is supported, refuted, insufficiently informed, or disputed by the evidence in the corpus. Therefore, we divide the task into two parts and conduct research using different model structures. The specific content of the project will be presented below.

2 Approach

2.1 Retrieval Task

For the first task, the first retrieval method we use is a classic and efficient information retrieval method TF-IDF (Term Frequency-Inverse Document Frequency). TF-IDF is a weighted technology commonly used in information retrieval. Its core method is to calculate the probability that a word appears in a document and its rarity in the entire corpus to determine whether the word can represent the main content of the document. This method does not consider the semantic meaning of the words. Instead, it focuses solely on their statistical frequency of occurrence within a document and across the corpus. The calculation formula of TF-IDF is shown below:

$$\text{TF-IDF}(t,d) = \text{tf}(t,d) \times \log\left(\frac{N}{\text{df}(t)}\right) \quad (1)$$

The report written by Mishra and Vishwakarmar(Mishra and Vishwakarma, 2015) explored and evaluated the performance of TF-IDF and its variants (like Lemur-TFIDF) in the FIRE 2011 news corpus. The results showed that TF-IDF performed well on multiple query tasks. Therefore, this project uses TF-IDF as the baseline method in our retrieval stage. In addition, we use cosine similarity to measure the similarity between the claim and each evidence, and select the top k (k is a constant representing the k most relevant evidence) evidence as candidate evidence for the claim.

The second retrieval method we explored leveraged the Word2Vec embedding model to capture the semantic similarity between claims and evidence. In contrast to frequency-based methods such as TF-IDF, Word2Vec provides continuous vector representations for words by training a shallow neural network that learns from local context windows. As introduced by Mikolov(Mikolov et al., 2013a), the core idea is to map words into a vector space where semantically similar words are close to each other. We experimented with two architectures: Continuous Bag of Words (CBOW) and Skip-gram, which differ in their prediction directions. CBOW predicts the center word based on its surrounding context, while Skip-gram predicts surrounding words given a center word.

However, softmax computations over large vocabularies are computationally expensive. To address this problem, we employ negative sampling (Mikolov et al., 2013b), a simplified objective that casts context prediction as a binary classification problem between true words and noise words:

$$\log \sigma(v_{w_O}^{\top} v_{w_I}) + \sum_{i=1}^{k} E_{w_i \sim P_n(w)} \left[\log \sigma(-v_{w_i}^{\top} v_{w_I}) \right] \tag{2}$$

where $\sigma(x) = \frac{1}{1+e^{-x}}$, and $P_n(w)$ is the noise distribution.

In our implementation, we used the Gensim library to train a Word2Vec model on the preprocessed text from the training claim and evidence corpus. After training, we calculated the average word embedding for each claim and each evidence paragraph. During retrieval, we calculated the cosine similarity between the claim vector and all evidence vectors, and selected the top 5 with the highest similarity as candidate results.

The third method is to use sentence embedding model plus cosine similarity to find the top 5 evidences that are most similar to the claim. First, the choice of sentence embedding model. I imported the "all-MiniLM-L6-v2" model from huggingface. This model is very good at handling the similarity between sentences because it embeds the sentence based on the semantics of the sentence, so sentences with similar semantics are closer in the vector space. Experimental results from Chen Yin and his team show that the model performs well in processing sentence pairs with different structures but the same semantics. After fine-tuning, the accuracy increases from 0.82 to 0.94, and the precision increases from 0.74 to 0.91, which is significantly better than the performance without fine-tuning(Yin and Zhang, 2024).So I fine-tuned the model. I used each claim in the training set and its corresponding random evidence to form a similar sentence pair as a new training set and input it into the model. The training loss selected MultipleNegativesRankingLoss. This loss function does not require explicit negative labels. It will automatically treat other positive sentences in the same batch as negative examples of the target sentence. The following is the equation of the loss function.

$$\text{Loss} = -\sum_{i=1}^{N} \log \frac{e^{\text{sim}(a_i,b_i)}}{\sum_{j=1}^{N} e^{\text{sim}(a_i,b_j)}} \tag{3}$$

After model training, use the fine tuned model to embed the claims and evidences in train and dev into sentences and save them for later use. The next step is to select the most relevant evidences based on the similarity between the claims and evidences. I choose cosine similarity as the sentence similarity method because it can effectively measure whether the directions of two vectors in the semantic space are consistent and is not affected by the length of the vectors. It is very suitable for comparing the semantic similarity of sentences, especially when using sentence embedding to represent sentences. The formula is as follows:

$$\text{cosine\_similarity}(A, B) = \frac{A \cdot B}{\|A\| \cdot \|B\|} \tag{4}$$

Due to the RAM limitation of Colab, calculating the cosine similarity between each claim and 1.2 million pieces of evidence will cause memory overflow or slow operation. Therefore, I chose to use Facebook's open source vector similarity search library FAISS to speed up the retrieval process. According to the cosine similarity between claims and evidence, the most relevant evidences are selected for each claim.

2.2 Classification Task

In the second stage, we chose a text classification model based on the LSTM (Long Short-Term Memory). Compared with ordinary RNN, LSTM introduces a gating mechanism in each time step to control the retention and forgetting of information, to avoid the vanishing gradient problem in long-distance dependencies. The study (Dirash and Bargavi, 2021) proposed a text classification method using unidirectional LSTM in sentiment analysis tasks. The result shows that LSTM can not only effectively capture long-distance dependencies in text, but also perform classification tasks efficiently and accurately. In this task, we use the concatenation of each claim and the corresponding evidence as training samples and then use LSTM to predict the relation category of the claim. In the data pre-processing stage, we used the NLTK tool to perform text segmentation and convert all words into lowercase. Next, we built a vocabulary and introduced special symbols "<pad>" and "<unk>" to handle fixed-length input and unknown word problems. Finally, the processed text sequence is mapped to a token ID sequence and encoded as 0-3 for the four-category labels.

The second classification approach we explored leverages open-source lightweight large language models (LLMs) through context-based learning. In this setting, the model is not fine-tuned, but rather it is prompted with a small number of labeled examples and then asked to infer a label for a new claim-evidence pair. This design enables the model to apply general language and reasoning knowledge without requiring additional training while meeting the memory constraints of Colab.

We chose Flan-T5-base as our model because we consider it is suitable for this classification tasks. Flan-T5-base is a 250M-parameter encoder-decoder model that has been fine-tuned on a large number of instruction tracing tasks. It shows strong generalization capabilities on few-shot classification tasks.(Chung et al., 2022)

This approach is inspired by recent work on prompt-based learning, which shows that large language models can perform surprisingly well with only a few examples and without fine-tuning, especially when provided with label balancing and task-specific instructions.(Brown et al., 2020)

The third approach is to add a customized classifier head based on the pre-trained language model BERT. We fine-tune the BERT model by adding a custom classifier layer, which has been shown to perform well in sentence-level classification tasks by Kumar and his team.(Kumar et al., 2024).First, I preprocessed the data, converted it into claim-evidence pairs, and added corresponding labels. For the pre-trained model, I chose the "bert-base-uncased" model, which is also a popular bert model from Google. I mainly customized the neural network classification layer to better adapt to the claim-evidence classification task. The main structure of FFNN is two layers of full connection + ReLU + Dropout.Figure 1 Shows the customized neural network structure diagram.Input is from BERT CLS embedding.The linear layer is used to reduce the dimension but retain the main features. ReLU activation enables the model to fit more complex relationships. The Dropout layer prevents overfitting. Two fully connected layers are used to further aggregate semantic features. In the classification model, we combine each claim with its corresponding multiple evidences to form a claim-evidence pair and predict them separately. For each pair, the model will output a predicted label. In order to determine the final classification result of the claim, we use majority voting, that is, counting the prediction results of all pairs and selecting the label with the highest frequency as the final predicted label.

3 Experiments

3.1 Evaluation method

In the retrieval task, we mainly use f1-score as our evaluation method, because in the retrieval task,

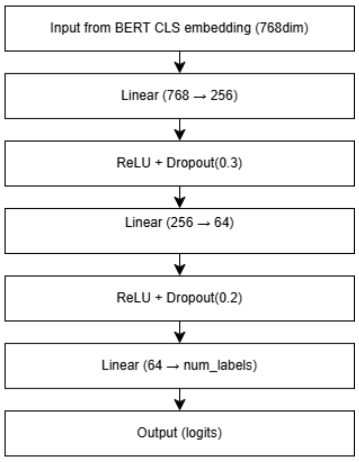


Figure 1: Architecture of the classifier built on top of BERT [CLS] embedding.

we not only care about whether the evidence found by the model is accurate (precision), but also care about whether the model has missed the correct evidence (recall), so it is more reasonable to choose F1-score that takes both into account. In the classification task, we choose accuracy as the criterion for judgment, because the classification task is a single-label prediction problem, and each sample has only one correct answer. We care about whether the model predicts correctly, so accuracy is the most insightful and appropriate evaluation indicator.

3.2 Experimental details and results

3.2.1 approach 1

Average Precision	0.0714
Average Recall	0.1434
Average F1 Score	0.08888

Table 1: TF-IDF Retrieval Result

Our retrieval baseline model uses TF-IDF retrieval. As shown in the figure 1, the results of TF-IDF as a baseline are not ideal. Therefore, we introduce other retrieval methods in the following. Next, the classification task uses a unidirectional LSTM model. We use a unidirectional LSTM structure, including an Embedding layer, an LSTM layer, a Dropout layer and a fully connected layer. During the training process, CrossEntropyLoss is used as the loss function to optimize the target. The optimizer is set to Adam and the learning rate is 1×10^{-3}. The model prediction output uses softmax as the activation function and using its maximam value as the classification result. In order to improve the final performance of the model, we designed three model configurations: model 1:

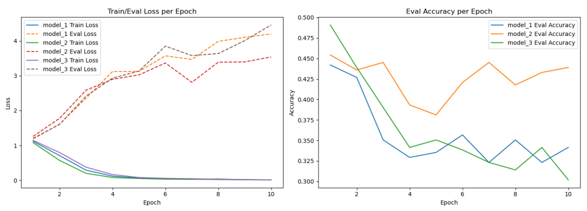


Figure 2: Enter Caption

Hidden Dim: 128, Dropout: 0.3; model 2: Hidden Dim: 256, Dropout: 0.3; "model 3": Hidden Dim: 128, Dropout: 0.5. The number of training rounds for each model is 10. Figure 2 shows the loss values of the three models on the training set and validation set and the accuracy of the validation set. The training loss of the three models dropped rapidly, proving that the model's fitting ability was sufficient. However, from the fourth round, the loss on the validation set continued to rise, indicating that the model was overfitting. Although model 3 used a higher dropout, the validation loss still showed a gradual upward trend, indicating that the strength of regularization did not improve the model's overfitting ability, which may be due to the noise of the data itself or the model is not the best case. Compared with the accuracy curves of the three models, model 2 has the most stable validation accuracy, the highest accuracy is about 0.45. The accuracy of model 1 was not good. Although model 3 performed well in the early stage, it subsequently dropped to 0.3, indicating that the generalization of the model is unstable.

3.2.2 approach 2

For the evidence retrieval task, we trained the Word2Vec model for six different configurations to study the impact of algorithm design and objectives on the results. Specifically, we conducted the following experiments:

- Architecture: CBOW vs. Skip-gram
- Objective Function: Hierarchical Softmax vs. Negative Sampling ($k = 5$ and $k = 10$)

All models are trained on tokenized claim and evidence texts using the Gensim library. Each text is represented by the average of its word vectors, and cosine similarity is used to compute the top 5 most relevant evidence paragraphs for each claim in the development set.

The retrieval performance is evaluated using the average F1 score in the development set, as shown in Table 2.

Model Configuration	**F1 Score**
CBOW + Hierarchical Softmax	0.0442
Skip-gram + Hierarchical Softmax	0.0635
CBOW + Negative Sampling (k=5)	0.0414
Skip-gram + Negative Sampling (k=5)	0.0724
CBOW + Negative Sampling (k=10)	0.0433
Skip-gram + Negative Sampling (k=10)	**0.0724**

Table 2: Retrieval performance of Word2Vec models

Overall, Skip-gram significantly outperforms CBOW on both evaluation metrics, which is consistent with previous research findings that Skip-gram is better at capturing the semantics of rare words. However, the best configuration (Skip-gram + negative samples $k = 10$) only achieves an F1 score of 0.0724, which is relatively low. We speculate that this may be due to (1) the large evidence corpus weakening the vector similarity and (2) the averaging of word vectors ignores word order and grammar, which may limit the accuracy of semantics.

For the classification stage, we used Flan-T5-base in a contextual learning setting. The prompts consisted of providing an example for each label, followed by the new argument and the top 5 retrieved evidences from the evidence retrieval step, with an explicit requirement that the output format be one of the four labels. On the development set, the model achieved an accuracy of 0.4285. However, observing the labels revealed that the model almost exclusively predicted the label "support", indicating a significant prediction bias.

We attribute this result to two main factors: (1) the quality of the retrieved evidence is relatively low, which limits the classifier's ability to make informed decisions; and (2) the length of the prompt may exceed the optimal input capacity of the model, thereby weakening the signal of the minority class examples and reducing the classification accuracy.

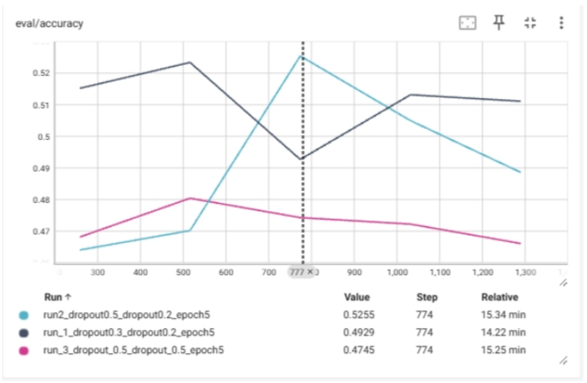
3.2.3 approach 3

In the retrieval task, we use all-MiniLM-L6-v2 model as the base encoder, and applied the MultipleNegativesRankingLoss loss function to fine tune it on the training set of claim-evidence pair. for training configuration, we set the batch size to 32, trained for 3 epochs, and used the default AdamW optimizer.To evaluate the effect of training epoches on retrieval performance, we fine-tuned the all-MiniLM-L6-v2 model with different numbers of epoche: 3,5,7. The Table 3 shows the retrieval task performace on developement set under different training epochs. We found that the model reached the best performance in 5 epochs, with an f1 score of 0.18.

Model Setting	precision	recall	F1-score
Baseline (no fine-tuning)	0.123	0.222	0.147
Fine-tuned (3 epoch)	0.147	0.263	0.176
Fine-tuned (5 epochs)	**0.151**	**0.266**	**0.180**
Fine-tuned (7 epochs)	0.133	0.234	0.152

Table 3: Retrieval task performance under different training epochs.

In the classification task, we fine-tune the BERT model by adding a custom classification layer. For training configuration, we set the training batch size to 16, evaluation batch size to 32, learning rate to $3e^{-5}$, trained for 5 epochs, and used the default AdamW optimizer. After each epoch, we evaluate the model on the validation set and save the model based on the epoch with the highest validation accuracy. Finally, we select the model with the highest validation accuracy as the final model for testing and analysis. To explore whether the dropout value in the classification layer affects the classification performance, we trained the model using the same claim dataset with different dropout values and the same epochs. The accuracy on the validation set is shown in the figure 3. The model with dropout=0.5/0.2 achieved the best results, with the highest accuracy of 0.525.



	Run ↑	Value	Step	Relative
●	run2_dropout0.5_dropout0.2_epoch5	0.5255	774	15.34 min
●	run_1_dropout0.3_dropout0.2_epoch5	0.4929	774	14.22 min
●	run_3_dropout_0.5_dropout_0.5_epoch5	0.4745	774	15.25 min

Figure 3: Development set accuracy across different dropout configurations. The notation `dropout0.5_dropout0.2` indicates that the dropout rate is 0.5 after the first linear layer and 0.2 after the second linear layer.

In the retrieval task, Approach 3, which is based on a fine-tuned sentence embedding model and combined with cosine similarity for evidence retrieval, achieved the best results. Therefore, in the classification task, all our methods use the evidence information retrieved by Approach 3 as input.

On the final test set, Approach 3 performed as follows: the harmonic mean of F value and accuracy (H(F, A)) was 0.21, the evidence retrieval F-score was 0.14, and the claim classification accuracy was 0.47.

4 Result Comparison

The table 4 shows the fine-tuned sentence embedding model with cosine similarity achieves the highest evidence retrieval F-score of 0.18, significantly outperforming traditional baselines such as TF-IDF (0.0888). The main reason is that the traditional word vector can only represent a single word, while the sentence embedding model can capture the contextual semantics and grammatical structure of the entire sentence, and is fine-tuned according to the train-evidence pair to make it more suitable for specific contexts or goals.

Despite semantic competence, our Word2Vec-based search underperformed TF-IDF. We ascribe this to averaging embeddings, which masks-out syntactic as well as directional information that's essential to differentiate factual relationships. TF-IDF, in contrast, overtly leverages surface-level lexical overlap, which worked better in our dataset to identify pertinent pieces of evidence. So in the classification task, we consistently use the results of the fine-tuned sentence embedding model with the best performance in the retrieval task as the test input.

The table 5 shows the classification task performance comparison of different models on the dev set.Fine-tuned BERT model with custumized classifier head achieves the highest Claim Classification accuracy of 0.525, which is higher than LSTM (baseline) and LLM (flan-t5-base), because BERT's embedding representation is very suitable for classification, and the custom classification head further improves the model's expressiveness. In contrast, LSTM has difficulty generalizing complex semantics and cannot capture long-distance dependencies well.

While Flan-T5-base is a more powerful pre-trained language model, its contextual performance is limited by prompt length and lack of task-specific fine-tuning in our setting. In contrast, the LSTM classifier is optimized for the classification objective directly using labeled data and is better able to exploit task-specific signals, leading to higher development accuracy.

Table 6 shows task performance comparison of different models on the test set. On the test set, the Fine-tuned BERT model with customized classifier head still achieves the best or tied for the best in all three models.

Model / Method	Evidence Retrieval F-score
TF-IDF (baseline)	0.0888
Word2Vec	0.0724
Fine-tuned sentence embedding model with cosine similarity for retrieving evidence	**0.18**

Table 4: Retrieval task performance comparison of different models on the dev set.

Model / Method	accuracy
LSTM (baseline)	0.45
in-context learning LLMs(flan-t5-base)	0.428
fine-tuned BERT model with custumized classifier head	**0.525**

Table 5: Classification task performance comparison of different models on the dev set.

Model / Method	Harmonic Mean of F and A	Evidence Retrieval F-score	Claim Classification Accuracy
LSTM(baseline)	0.0948	0.0564	0.2987
in-context learning LLMs(flan-t5-base)	0.21	0.14	0.44
Fine-tuned BERT model with customized classifier head	**0.21**	**0.14**	**0.47**

Table 6: Task performance comparison of different models on the test set.

5 Conclusion

This project focuses on developing and evaluating various retrieval and classification models for the claim verification task. In the retrieval stage, we tried three different methods, analyzed and compared their performance. In the classification task, we also compared three different classification models. Through some fine-tuning and hyper-parameter changes, the performance of the model was improved and the best model was obtained by comparison. The experimental results show that Sentence Embedding has the best performance in the retrieval task. Combined with Sentence embedding, the accuracy of classification prediction using Fine-tuned BERT is the highest. In general, this project not only completed the system implementation and evaluation of the two subtasks, but also compared multiple models and provided assistance for the design of novel model architectures. In the future, we can consider using multi-round reasoning, self-attention, and more advanced large language models.

6 Team contributions

Every member of our team participated in all parts of the project, including the design and implementation of the model and the writing of the experiments and conclusions.The specific contributions are as follows:

1254859 was responsible for the design and application of approach 1, using TF-IDF retrieval method and combining unidirectional LSTM with different hyper-parameters to explore the design of model mechanism.

1166507 was responsible for the design and implement of approach 2, using word2vec + average pooling + cosine similarity for evidence retrieval and try to implement solutions of in-context learning LLMs and design prompts

1001087 was mainly responsible for the design and application of approach 3, using fine-tuned sentence embedding model + cosine similarity for retrieving evidence and fine-tuned bert with customized Classifier Head for classification task.

References

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, and 12 others. 2020. Language models are few-shot learners. *Preprint*, arXiv:2005.14165.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, Albert Webson, Shixiang Shane Gu, Zhuyun Dai, Mirac Suzgun, Xinyun Chen, Aakanksha Chowdhery, Alex Castro-Ros, Marie Pellat, Kevin Robinson, and 16 others. 2022. Scaling instruction-finetuned language models. *Preprint*, arXiv:2210.11416.

AR Dirash and SK Bargavi. 2021. Lstm based text classification. *IITM Journal of Management and IT*, 12(1):62–65.

Ajay Kumar, Nilesh Ware, and Shaurya Gupta. 2024. Leveraging transfer learning: Fine-tuning methodology for enhanced text classification using bert. In *2024 IEEE Pune Section International Conference (PuneCon)*, pages 1–5.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *Preprint*, arXiv:1301.3781.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013b. Distributed representations of words and phrases and their compositionality. *Preprint*, arXiv:1310.4546.

Apra Mishra and Santosh Vishwakarma. 2015. Analysis of tf-idf model and its variant for document retrieval. In *2015 International Conference on Computational Intelligence and Communication Networks (CICN)*, pages 772–776.

Chen Yin and Zixuan Zhang. 2024. A study of sentence similarity based on the all-minilm-l6-v2 model with "same semantics, different structure" after fine tuning. In *Proceedings of the 2024 2nd International Conference on Image, Algorithms and Artificial Intelligence (ICIAAI 2024)*, pages 677–684. Atlantis Press.