Describe your approach to data proprocessing and information retrieval. Please choose at least 2 of any IR methods and compare their performance.

Method 1: Sentence Embedding-Based Evidence Extraction

Description:
這個方法使用了Sentence embedding 來從文本中提取與claim相關的evidences句子，並計算這些句子與claim之間的相似性。選擇這個方法的原因在於embedding model能夠有效地量化句子之間的語義相似性，能夠從大量的文本中迅速識別最相關的句子，從而提高準確性和處理效率。

Implementation Details:
此方法使用了 `SentenceTransformer` 庫中的「all-MiniLM-L6-v2」模型來生成句子嵌入。該模型是一個小型且高效的 Transformer 模型，能夠在 CPU 或 GPU 上快速運行，適合大規模文本處理。此外，為了加速我還使用了multi-thread和 NLTK 庫來進行文本的預處理和過濾，並用於加速處理過程中的分詞、停用詞移除、詞形還原等步驟。

```
from sentence_transformers import SentenceTransformer

model = SentenceTransformer('all-MiniLM-L6-v2', device='cpu')
embeddings = model.encode(sentences, batch_size=32, show_progress_bar=True)
```

Performance: Kaggle上預測分數為0.58245

Method 2: TF-IDF Based Evidence Extraction

Description:
此方法使用 TF-IDF（詞頻-逆文檔頻率） 計算claim與每個句子之間的相似度，並從文章中提取與claim最相關的句子。選擇此方法的原因是，TF-IDF 能夠有效量化詞彙在claim和文章句子中的相關性，提供快速的相似度計算。該方法特別適用於以較低計算成本來檢索具主題相關性的句子，從而輕鬆地識別出支持claim的evidences句子。

Implementation Details:
該方法使用 `TfidfVectorizer` 生成claim與文章句子的 TF-IDF 向量，並計算這些向量間的餘弦相似度，篩選出高於特定閾值（例如 0.38）的句子作為evidences。

```
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity

vectorizer = TfidfVectorizer()
tfidf_matrix = vectorizer.fit_transform([claim] + sentences)
similarities = cosine_similarity(tfidf_matrix[0:1], tfidf_matrix[1:]).flatten()
evidence = [sent for sent, score in zip(sentences, similarities) if score > 0.38]
```

Performance: Kaggle上預測分數為0.55839

Describe your approach to claim prediction. Details such as model selection, hyperparameters should be provided.

Approach to Claim Prediction

Model Selection

為了進行claim預測，我選擇了 **DeBERTa（Decoding-enhanced BERT with Disentangled Attention）** 模型，因為它在處理語意關係方面的表現優異，並且在需要理解細緻語言的 NLP 任務上表現出色。與 BERT 相比，DeBERTa 有兩個主要改進：Decoupled Attention和Relative Position Encoding。這讓它在處理claim和evidences時，能更靈活地理解語意和細微差異，對準確分類claim非常有幫助。

```
from transformers import DebertaV2Tokenizer, DebertaV2ForSequenceClassification

tokenizer = DebertaV2Tokenizer.from_pretrained('microsoft/deberta-v3-large')
model = DebertaV2ForSequenceClassification.
from_pretrained('microsoft/deberta-v3-large', num_labels=3).to(device)
```

Hyperparameters

為了達到最佳性能，我調整了以下超參數：
批次大小：設為 8，以平衡計算效率和穩定的梯度更新。
訓練回合數：設定為 3，允許模型收斂且避免過擬合。
學習率：1e-5，經過調整以確保收斂且最小化過擬合風險。
權重衰減：0.01，有助於透過懲罰大權重來防止過擬合。
預熱比率：0.1，允許學習率在訓練初期逐步提升以增強穩定性。
序列長度：最大 256 個 tokens，以平衡計算成本並保留claim和evidences文本的足夠上下文。

```
from transformers import AdamW, get_linear_schedule_with_warmup

optimizer = AdamW(model.parameters(),
lr=args.learning_rate, weight_decay=args.weight_decay)
scheduler = get_linear_schedule_with_warmup(
    optimizer,
    num_warmup_steps=int(args.warmup_ratio * total_steps),
    num_training_steps=total_steps
)
```

Techniques to Enhance Prediction Accuracy

為進一步提升預測準確性，我嘗試了常用於大型語言模型（LLM）的先進技術，如「In-Context Learning」和「Chain-of-Thought Prompting」，讓 DeBERTa 模型在進行claim預測時表現更佳。

1. **In-Context Learning (ICL)**：
   在訓練數據中提供了幾個已標註的claim範例，包含不同的真實性等級（如：真、部分真、假），以幫助模型學習不同claim的分類。
   這種方法有助於 DeBERTa 在微調過程中更好地進行概括，提高模型在處理不同細微差異的claim時的準確性。
2. **Chain-of-Thought Prompting (CoT)**：
   我使用了一種逐步思考的方式，讓模型在得出最終分類前先考慮中間推理步驟。例如，模型首先分析evidences上下文，檢查是否有矛盾或支持，然後再進行最終預測。
   對於需要多重evidences推理的複雜claim，這種技術顯著提升了模型的表現，使模型能夠更有邏輯地分解和評估每個claim。

3. **Self-Consistency Decoding**:
   為了增強一致性，我透過對同一claim生成多次輸出並選擇最頻繁的分類結果，減少了預測中的雜訊和變異性，尤其在處理模糊的claim時表現較佳。

```
from torch.cuda.amp import GradScaler, autocast

scaler = GradScaler()

with autocast():
    outputs = model(**inputs)
    loss = torch.nn.CrossEntropyLoss()(outputs.logits, labels)

scaler.scale(loss).backward()
scaler.step(optimizer)
scaler.update()
```

Do error analysis or case study. Is there anything worth mentioning while checking the mispredicted data? Share with us. Anytime you try to make a conclusion about the data or model, you should provide concrete data example.

Error Analysis

在模型訓練和驗證過程中，我觀察到模型在處理某些claim的時候會比較弱一點。以下是兩個具體的錯誤案例，展示了當模型在處理複雜或特定情境的claim時，因evidences不足而導致的誤判問題。

Case 1

Claim: "OpIndia claimed Greta Thunberg's real name is Ghazala Bhat."
True Rating: 0 (False)
Predicted Rating: 2 (True)

Analysis

這個Claim其實是假的，因為 Greta Thunberg 的真實名字並不是 Ghazala Bhat，但模型卻將其預測為真。出現這樣的錯誤，可能主要原因在於模型在處理此claim時，無法獲得充足且具有反駁性的evidences來支持“假”的判斷。

evidences處理能力不足（Insufficient Evidence Handling）：模型未能找到明確指向Thunberg真實姓名的反駁evidences，可能是因為模型在訓練中缺少足夠的、具體的假claim數據來學習如何應對這類虛假的說法。當evidences不足以支持反駁時，模型傾向於將該claim判斷為真。此外，模型對“OpIndia”這樣的來源缺乏足夠的上下文理解，無法識別來源的可信度偏差，使得模型更容易在缺少反面evidences的情況下誤判。

Case 2

Claim: "38,000 prisoners were released from federal prison during the Obama administration."
True Rating: 1 (Partially True)
Predicted Rating: 2 (True)

Analysis

這個Claim實際上是部分真實的，因為在奧巴馬政府期間確實有囚犯被釋放，但數字不完全準確。然而，模型將其預測為完全真。此誤判的可能原因如下：

evidences範圍不足（Insufficient Range of Evidence Support）：模型在處理這類涉及具體數字的claim時，難以找到針對“38,000”這個具體數字的evidences來驗證其準確性。由於無法找到足夠廣泛的背景信息來支持部分真實的標記，模型只能根據現有有限的evidences將其判斷為完全真。

以上兩個case是我在訓練並觀察錯誤預測時所觀察到的case