



T BRAIN AI 實戰吧

AI CUP 2024 玉山人工智慧公開挑戰賽 -RAG與LLM在金融問答的應用

TEAM_6178

TABLE OF CONTENTS

01

Dataset & Baseline

02

Our Framework

03

Result & Evaluation

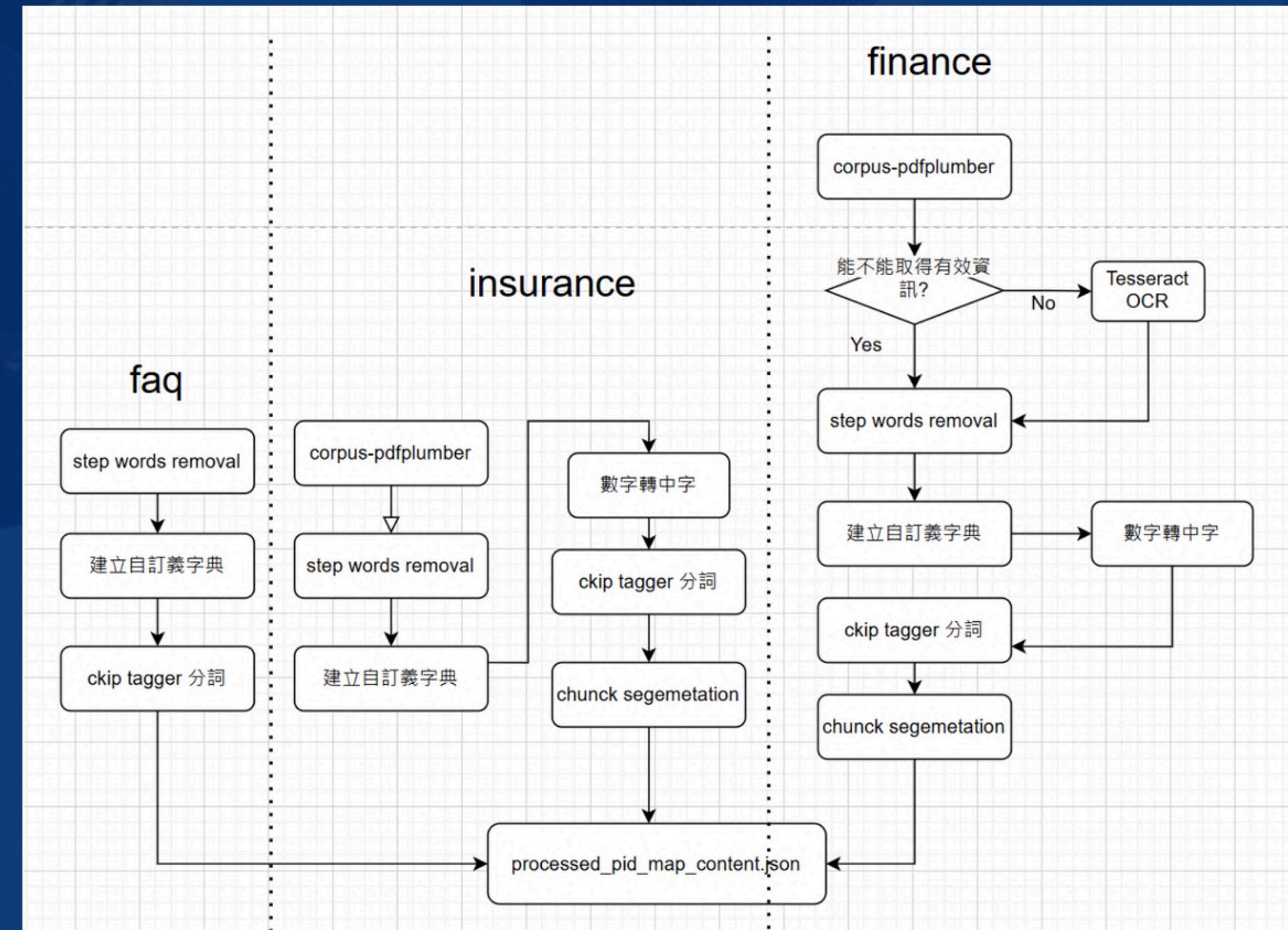


Our Framework

- 1. Preprocess (OCR, CKIP, Dictionary)**
- 2. Retrieval (Embedding + Reranker)**
- 3. Result**



Preprocess Workflow



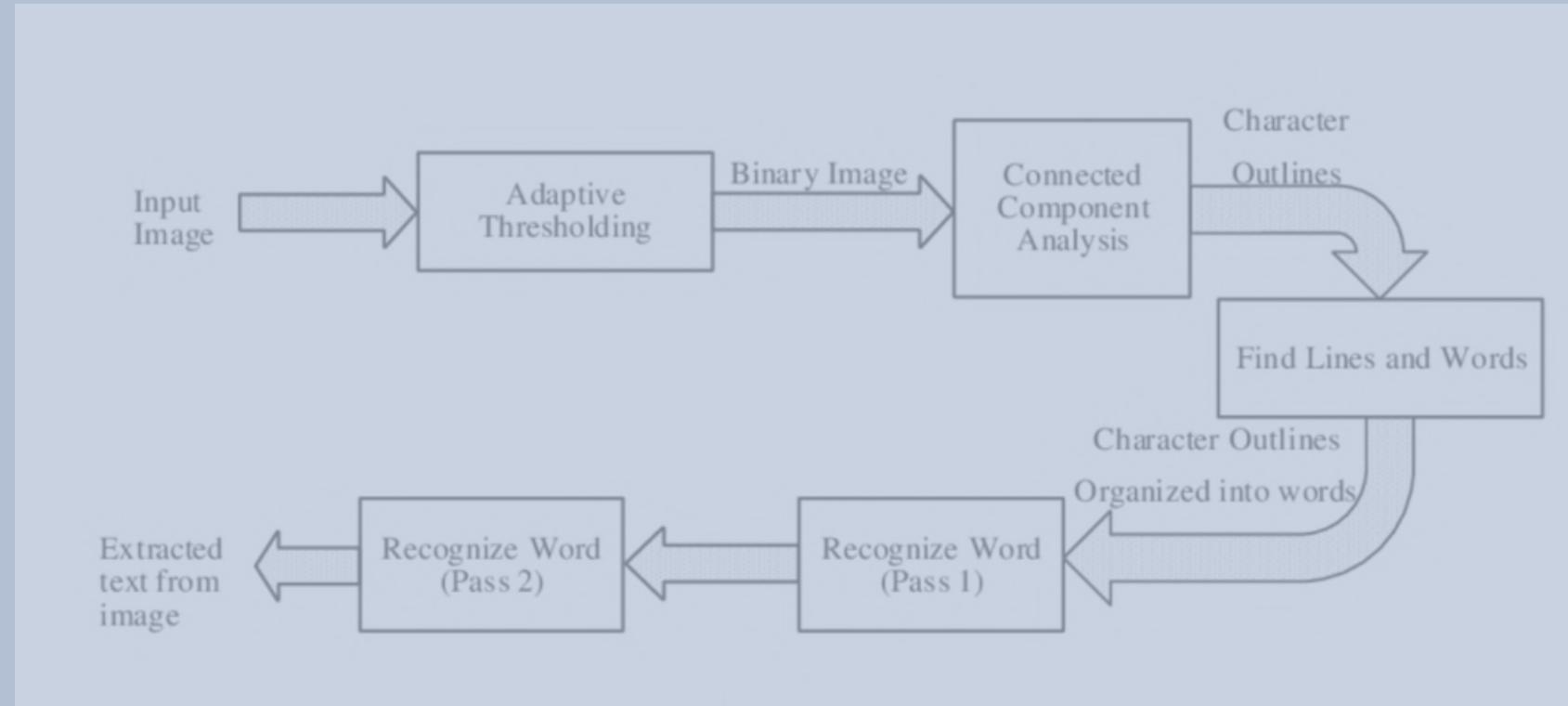
Preprocess-Extract Information

從 PDF 中提取文字：

將 PDF 文件中的文字內容提取出來，以便進行進一步的處理或分析

Tesseract OCR 技術支援：

遇到報表等無法直接提取時則使用tesseract OCR 輔助



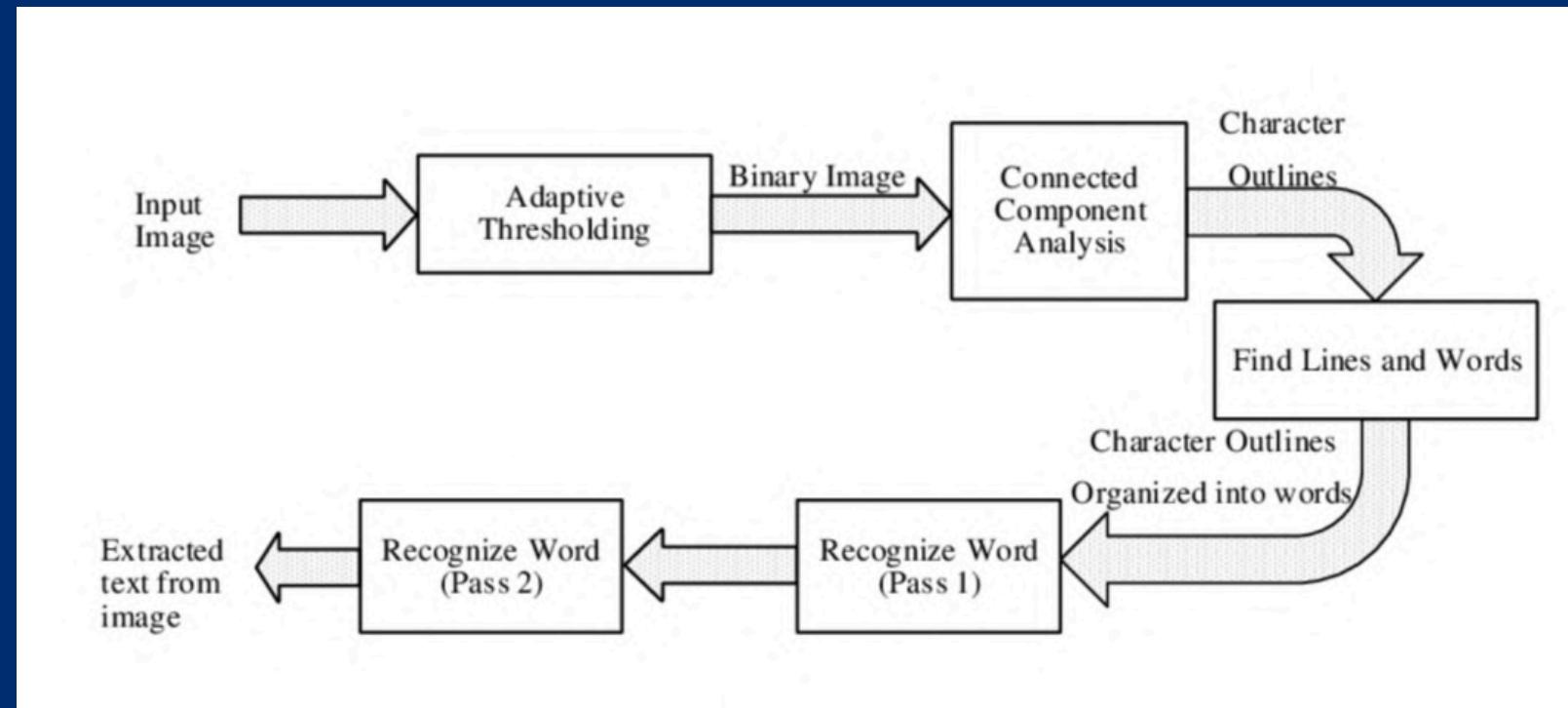
Preprocess-Extract Information

從 PDF 中提取文字：

將 PDF 文件中的文字內容提取出來，以便進行進一步的處理或分析

Tesseract OCR 技術支援：

遇到報表等無法直接提取時則使用tesseract OCR 輔助



Preprocess-Extract Information

Tesseract OCR 技術支援：

遇到報表等無法直接提取時則使用tesseract OCR 輔助

```
def read_pdf(pdf_loc, page_infos: list = None, max_tokens=512, overlap_tokens=50):
    try:
        pages = pdf.pages[page_infos[0]:page_infos[1]] if page_infos else pdf.pages
    except IndexError:
        logging.warning(f"Page range {page_infos} out of bounds for file {pdf_loc}. Extracting all pages.")
        pages = pdf.pages

    pdf_text = ''
    for page_number, page in enumerate(pages, start=1):
        try:
            text = page.extract_text()
            if text:
                logging.info(f"Extracted {len(text)} characters from page {page_number} of {pdf_loc} using pdfplumber.")
                pdf_text += text + "\n\n"
            else:
                logging.info(f"No text found on page {page_number} of {pdf_loc}. Attempting OCR.")
                image = page.to_image(resolution=300).original
                pil_image = image.convert("RGB")
                ocr_text = pytesseract.image_to_string(pil_image, lang='chi_tra', config='--psm 6') #Tesseract OCR
                if ocr_text.strip():
                    ocr_text = normalize_text(preprocess_text(ocr_text))
                    logging.info(f"Extracted {len(ocr_text)} characters from page {page_number} of {pdf_loc} using OCR.")
                    pdf_text += ocr_text + "\n\n"
                else:
                    logging.warning(f"OCR failed to extract text from page {page_number} of {pdf_loc}.")
        except Exception as e:
            logging.error(f"Error processing page {page_number} in {pdf_loc}: {e}")
    pdf.close()
```

Preprocess-Clean Data

使用方法：

1. 過濾資料：去除冗字
2. Number to Chinese
3. 切詞工具：CKIP Tagger (<https://github.com/ckiplab/ckiptagger>)
4. 自訂字典增加特定詞彙權重

1

過濾資料: 去除冗字

```
# 定義停用詞表
stopwords = {"的", "了", "之", "在", "和", "也", "有", "是", "於",
             "\n", ":", ".", "[", "]", "【", "】", "、", ";", "「", "」"}
def remove_stopwords(words, stopwords):
    # 過濾掉不需要的詞
    return [word for word in words if word not in stopwords]
```

程式碼示意圖

2

NUMBER TO CHINESE

```
def number_to_chinese(num):
    units = ["", "十", "百", "千", "萬", "億"]
    digits = "零一二三四五六七八九"
    result = ""
    str_num = str(num)
    length = len(str_num)

    for i, digit in enumerate(str_num):
        digit_value = int(digit)
        if digit_value != 0:
            result += digits[digit_value] + units[length - i - 1]
        elif not result.endswith("零"):
            result += "零"

    result = result.replace("-+", "+")
    result = result.rstrip("零")
    return result
```

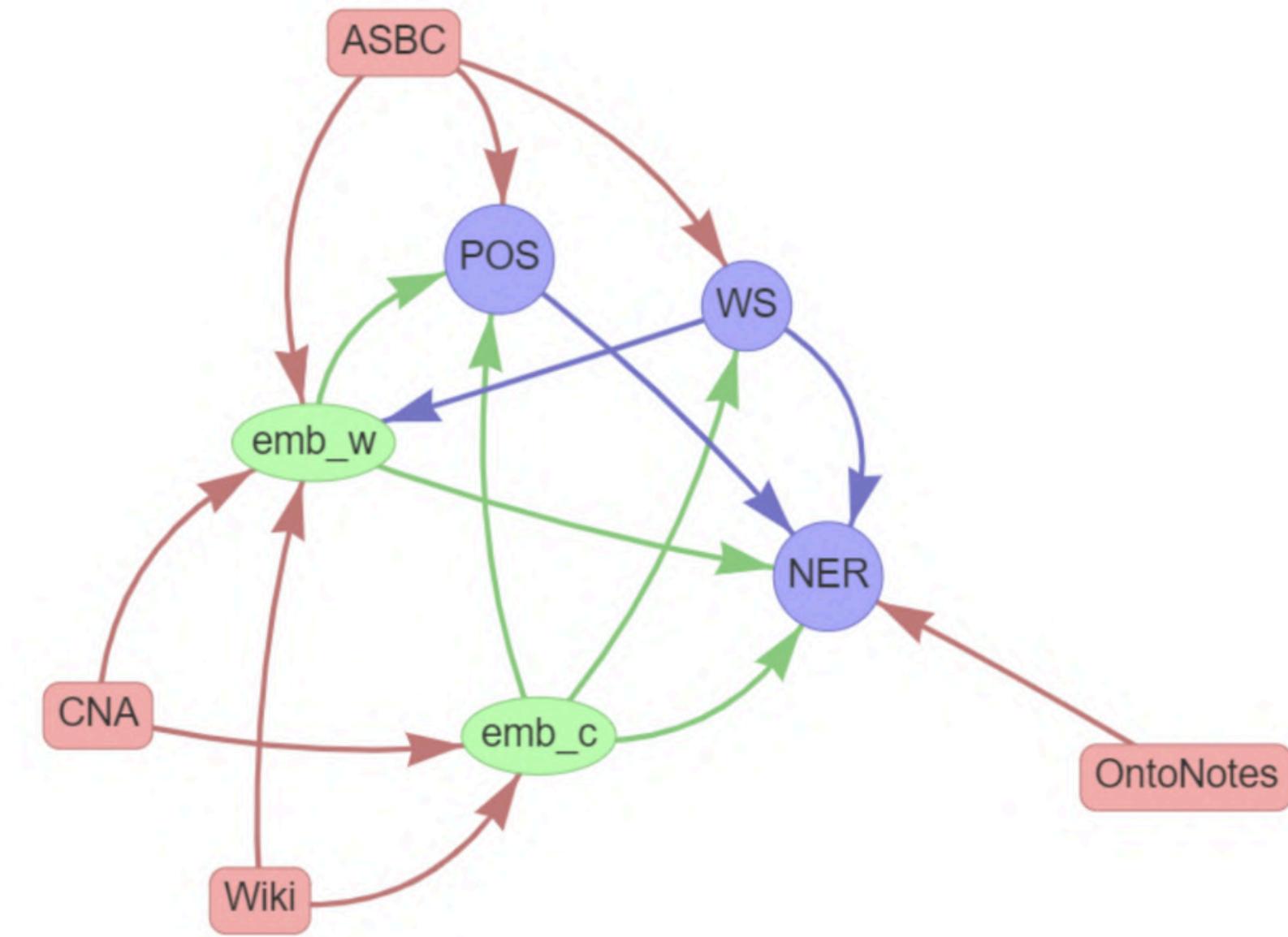
程式碼示意圖

3

切詞工具：CKIP TAGGER

CkipTagger Features

- Performance (compared with CKIPWS)
 - WS: +1.4% absolute F1 on ASBC 4.0 test split
 - POS: +4.0% absolute accuracy on ASBC 4.0 test split
- Ease-of-use
 - Do not auto delete/change/add characters
 - Support indefinitely long sentences
- Features
 - Do not rely on word list, word frequency statistics, POS frequency statistics
 - Support user-defined recommended-word list
 - Support user-defined must-word list



4

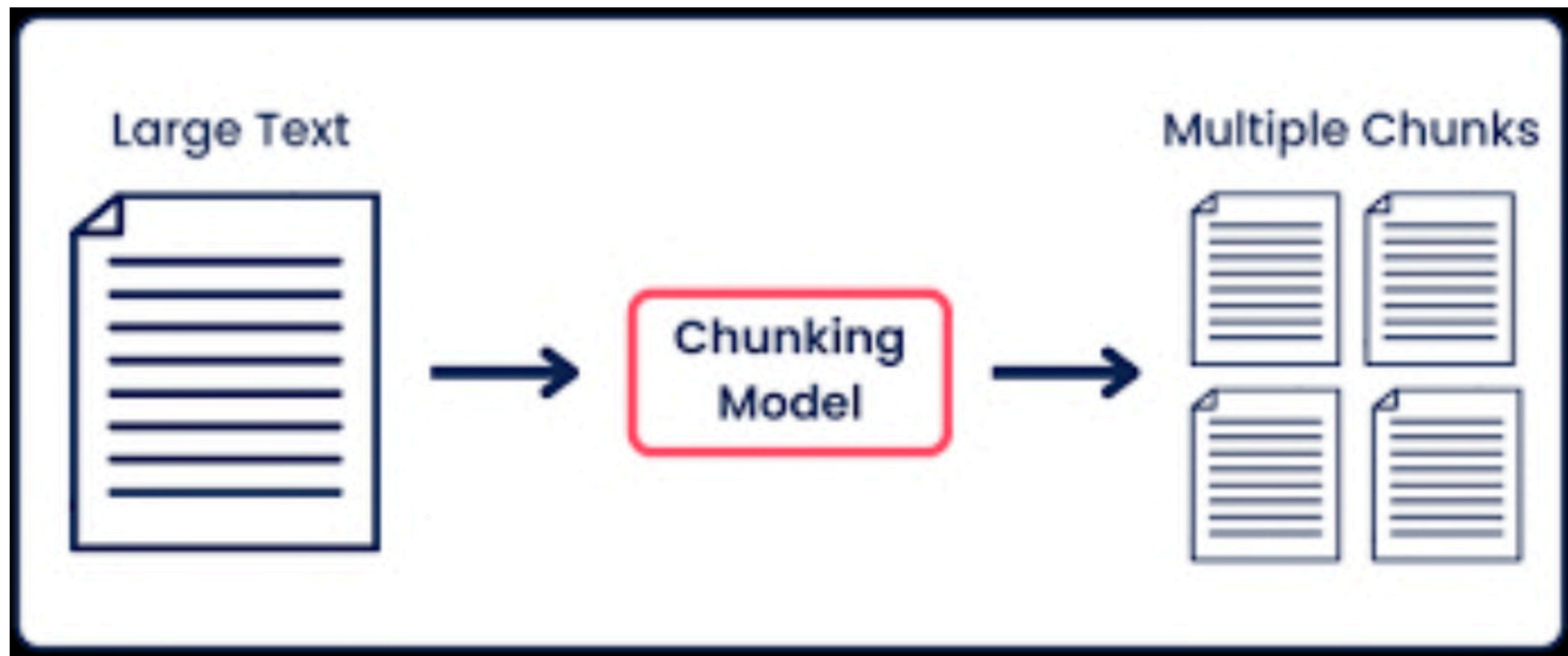
自訂字典

```
custom_dict = {  
    # 財務相關  
    "對帳單": 1, "綜合對帳單": 1, "支付寶": 1, "本期淨利": 1,  
    "約當現金": 1, "資產 負債 表": 1, "綜合 損益 表": 1,  
    "權益 變動 表": 1, "每股盈餘": 1,  
  
    # 保險相關  
    "要保人": 1,  
  
    # 科技公司相關  
    "瑞昱": 1, "華碩": 1, "智邦": 1,  
    "研華": 1, "台達電子": 1,  
  
    # 其他  
    "刷臉": 1  
}
```

程式碼示意圖

Text Chunking

進行文本切分 (text chunking) 時，可以根據設定的最大字數 (max chunk size) 來進行切分，並利用標點符號來幫助確定每個區塊的邊界。這樣可以保證每個切分出的區塊都具有自然語言的邏輯結構，不會在半句話或不完整的語意上進行切分。

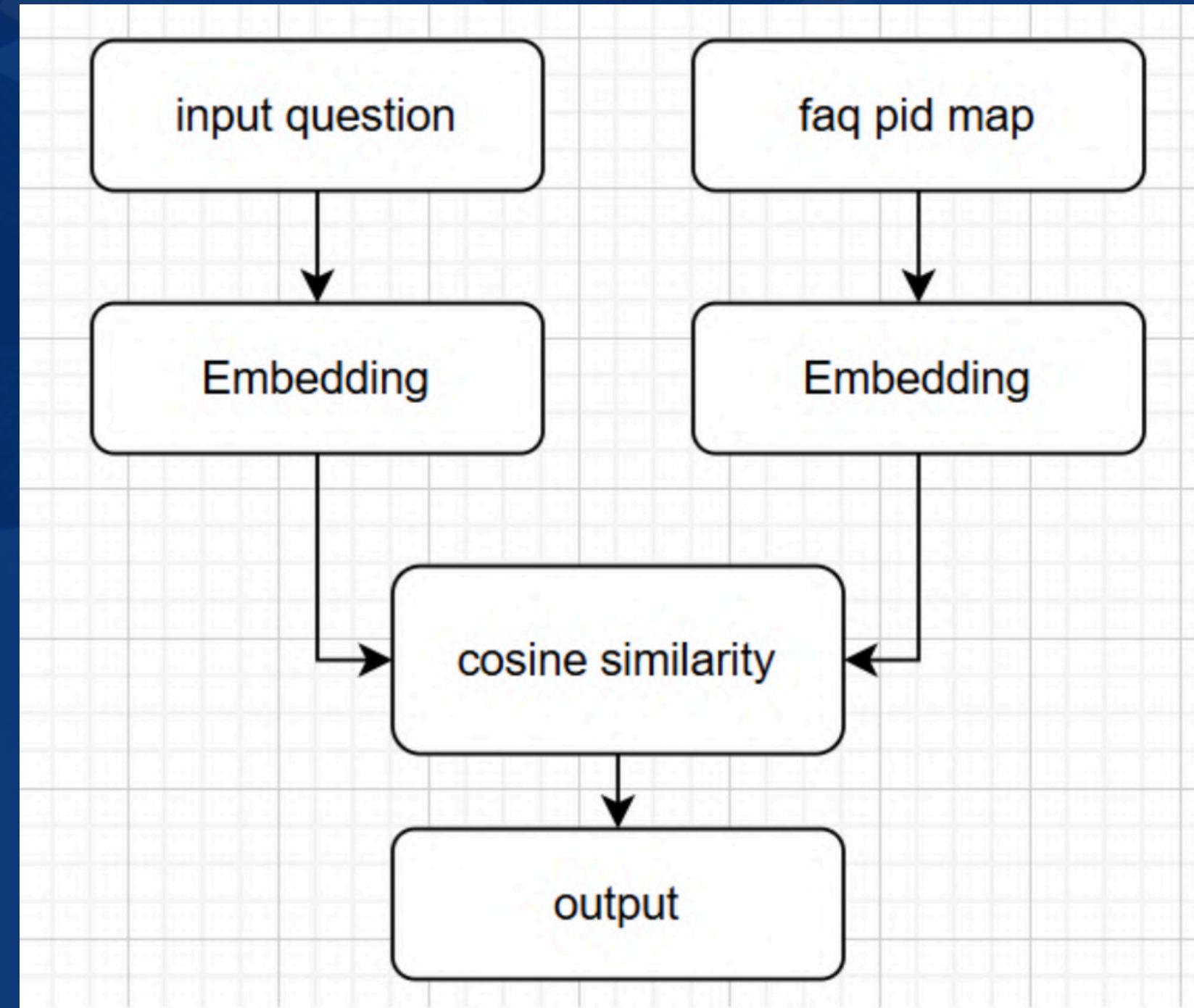


Text Chunking 流程示意圖

Preprocess-Clean Data

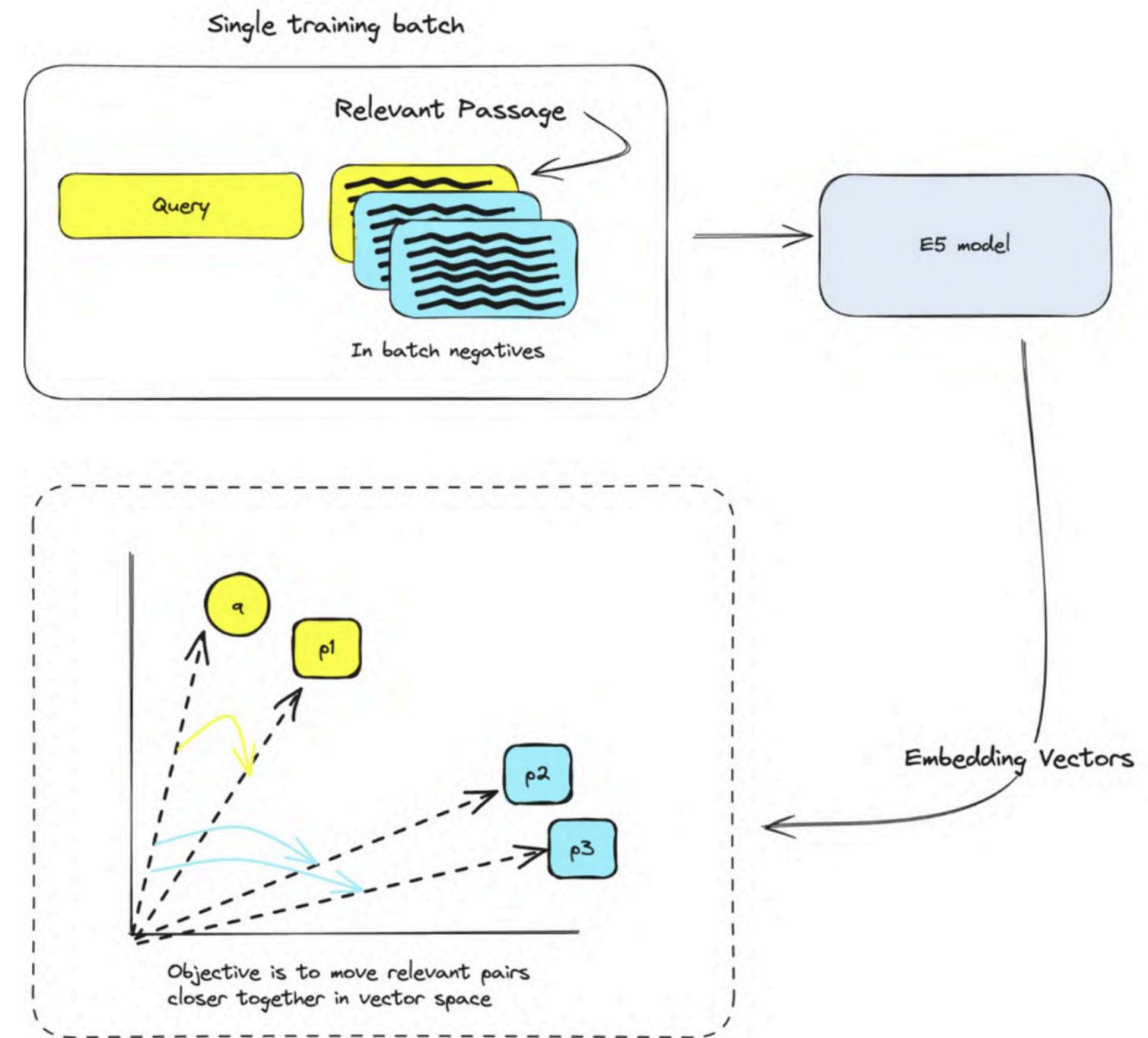
```
"0": [
  {
    "問題": "什麼 跨境 手機 掃碼 支付 ?",
    "答案": [
      "允許 大陸 消費者 可以 用 手機 支付寶 App 台灣 實體 商店 購買 商品 或 服務"
    ]
  }
],
"1": [
  {
    "問題": "什麼 網路 金融卡 收款 ?",
    "答案": [
      "網路 金融卡 收款 消費者 透過 讀卡機 與 晶片 金融卡 立即 轉帳 支付 購買 商品 或 服務 款項 即時 入帳 付款 輕鬆 無 斷點"
    ]
  }
],
{
  "corpus_dict": {
    "0": [
      {
        "問題": "網路 金融卡 收款 特點 什麼 ?",
        "答案": [
          "款項 即時 入帳",
          "付款 輕鬆 無 斷點"
        ]
      }
    ],
    "1": [
      {
        "corpus_dict": {
          "1": [
            "註 十 本 集團 民國 一百一十一 年 第一季 投資 成立 寶元 智造 公司 由 本 集團 持 100% 股權 另
            "100% 股權 註 十七 本 集團 民國 一百一十一 年 第四季 投資 設立 BEST AUTOMATION 本 集團 持有
          ],
          "10": [
            "延期 間 內 發生 第十六條 或 第十七條 本 公司 應 負 保險 責任 事故 時 其 約定 保險金 計算 方式
            "金額 上限 為 借款 當日 保單 價值 準備金 百分之八十 未 償還 借款 本息 超過 其 保單 價值 準備金
            "三 、 因 投保 年齡 錯誤 而 致 短 繳 保險費 者 要保人 得 補繳 短繳 保險費 或 按照 所 付 保險費
            "發給 批註書 身故 受益人 同時 或 先於 被保險人 本人 身故 除 要保人 已 另行 指定 受益人 外 以 被
          ],
          "100": [
            "南山 人壽 新福 愛 小額 終身 壽險_MPL3\n 金 或 喪葬 費用 保險金 歸還 本 公司 其間 若 應 索償
            "其餘 死亡 索償 約定 被保險人 滿 十五 足歲 日 起 發生 效力 ； 被保險人 滿 十五 足\n 歲 前 死亡
            "身故 保險金 變更為 喪葬 費用 保險金 第四 項 未滿 十五 足歲 被保險人 民國 一百零九年 六月 十二日
            "為 基礎 並 溯自 本 契約 生效日 起 計算 二 、 若 依 本 契約 第二十三條 辦理 減額 繳清 保險者
            "完全 失能 等級 時 本 公司 僅 索償 一 款 完\n 全 失能 保險金 本 公司 依 本 條 約定 索償 完全
          ]
        }
      }
    ]
  }
]
```

Retrieval workflow - FAQ



Retrieval - FAQ embedding model

multilingual-e5-large



Retrieval - FAQ embedding model

multilingual-e5-large

```
def embedding_e5_l(input_texts, relevant_doc_ids):
    # 分詞
    batch_dict = tokenizer_l(input_texts, max_length=512, padding=True, truncation=True, return_tensors='pt')
    # 獲取模型輸出
    outputs = model_l(**batch_dict)
    embeddings = average_pool(outputs.last_hidden_state, batch_dict['attention_mask'])
    # 正規化嵌入向量
    embeddings = F.normalize(embeddings, p=2, dim=1)
    # 計算查詢與每個文檔的相似度分數
    query_embedding = embeddings[0] # 第一個是查詢的嵌入
    doc_embeddings = embeddings[1:] # 其餘的是相關文檔的嵌入
    # 計算相似度分數
    scores = (query_embedding @ doc_embeddings.T) * 100

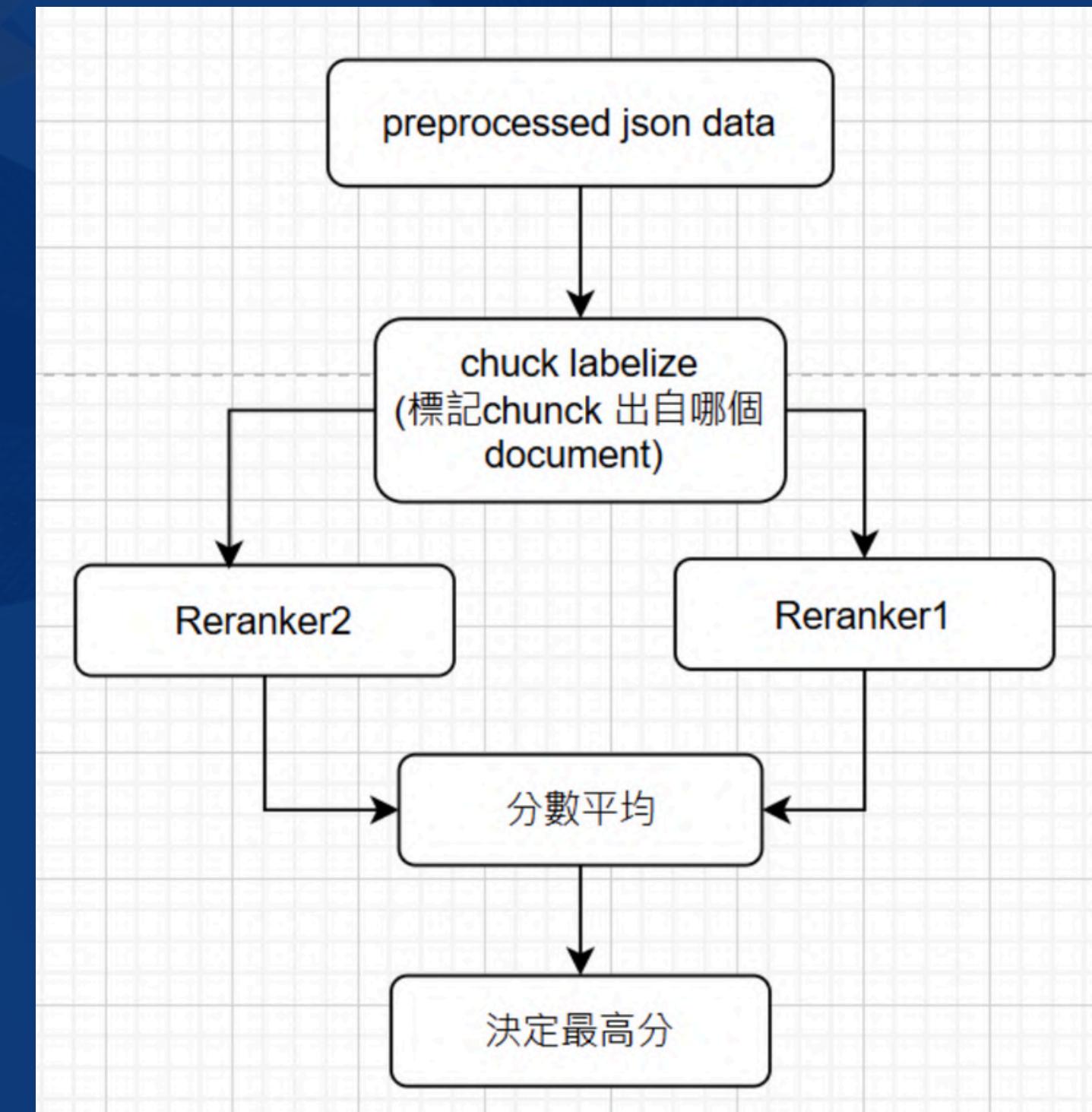
    scores = scores.tolist()
    # 建立字典來儲存每個 doc_id 的最大分數
    doc_score_dict = {}

    for score, doc_id in zip(scores, relevant_doc_ids):
        # 如果 doc_id 已存在於字典中，取較大分數
        if doc_id in doc_score_dict:
            doc_score_dict[doc_id] = max(doc_score_dict[doc_id], score)
        else:
            doc_score_dict[doc_id] = score

    return doc_score_dict
```

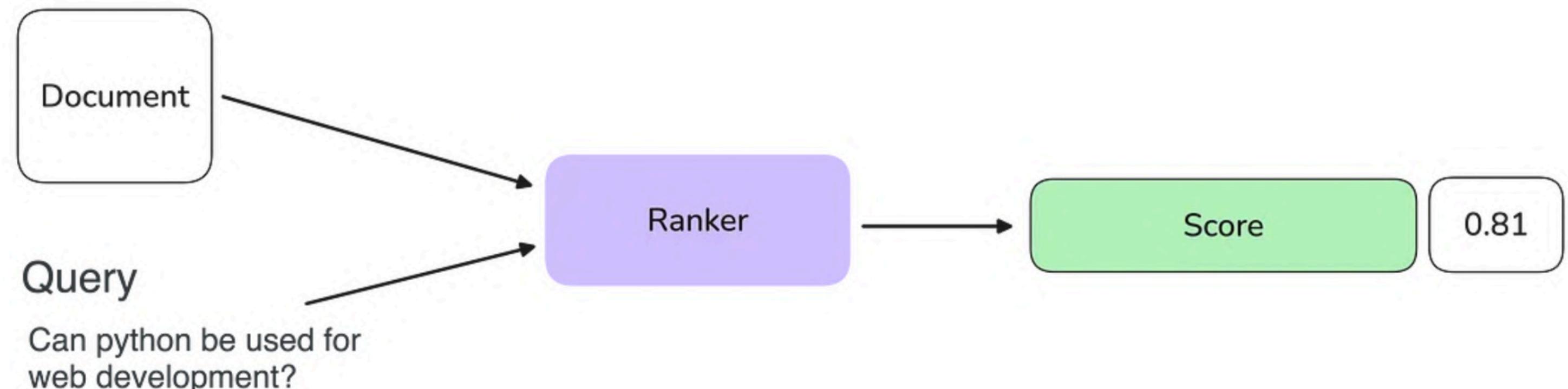
程式碼示意圖

Retrieval workflow - Finance & Insurance



Reranker

使用繁體中文評測各家 Reranker 模型的重排能力
<https://ihower.tw/blog/archives/12227>



Reranker 流程示意圖

Retrieval-Finance Insurance Reranker

BAAI/bge-reranker-large

```
def rerank_with_bge(valid_relevant_docs):
    # Reranker1: compute scores using FlagReranker
    try:

        score_inputs = [[query, doc] for doc in valid_relevant_docs]
        reranker_bge_scores = reranker_bge.compute_score(score_inputs)
        logging.info(f"QID: {qid}, Reranker1 scores: {reranker_bge_scores[:5]}")

        # 標準化 Reranker1 分數到 -1 ~ 1
        normalized_reranker_bge_scores = [score / 10 for score in reranker_bge_scores]
        logging.info(f"QID: {qid}, Normalized Reranker1 scores: {normalized_reranker_bge_scores[:5]}")
    except Exception as e:
        logging.error(f"Error during reranking with Reranker1 for QID: {qid}: {e}")
        answer_dict['answers'].append({"qid": qid, "retrieve": None})

    return normalized_reranker_bge_scores
```

程式碼示意圖

Retrieval-Finance Insurance Reranker

maidalun1020/bce-reranker-base_v1

```
# Function to compute reranker2 scores using AutoModelForSequenceClassification
def rerank_with_bce(query, documents, reranker_model, reranker_tokenizer, device, batch_size=32):
    try:
        reranker_model.eval()
        sentence_pairs = [[query, doc] for doc in documents]
        scores = []
        with torch.no_grad():
            for i in tqdm(range(0, len(sentence_pairs), batch_size), desc="Computing Reranker2 Scores"):
                batch = sentence_pairs[i:i+batch_size]
                inputs = reranker_tokenizer(batch, padding=True, truncation=True, max_length=512, return_tensors="pt")
                inputs = {k: v.to(device) for k, v in inputs.items()}
                outputs = reranker_model(**inputs)
                logits = outputs.logits.view(-1).float()
                # 應用 sigmoid 轉換為概率分數
                probabilities = torch.sigmoid(logits)
                batch_scores = probabilities.cpu().numpy()
                scores.extend(batch_scores)
            logging.info(f"QID: {qid}, Reranker2 scores: {scores[:5]}")
    except Exception as e:
        logging.error(f"Error during reranking with Reranker2 for QID: {qid}: {e}")
        answer_dict['answers'].append({"qid": qid, "retrieve": None})
    return scores
```

程式碼示意圖

Retrieval- Average Scores

```
# Calculate final scores based on model choice
if args.model_type == 'bge':
    final_scores = rerank_bge_score
elif args.model_type == 'bce':
    final_scores = rerank_bce_score
else: # both
    final_scores = [(bge + bce) / 2 for bge, bce in zip(rerank_bge_score, rerank_bce_score)]

# 將文檔與平均分數及 doc_id 組合
doc_scores = list(zip(valid_relevant_docs, final_scores, valid_relevant_doc_ids))
# 按平均分數降序排序
top_n = 1
top_n_docs = sorted(doc_scores, key=lambda x: x[1], reverse=True)[:top_n]
if not top_n_docs:
    logging.warning(f"No documents after reranking for QID: {qid}")
    answer_dict['answers'].append({"qid": qid, "retrieve": None})
    continue
# 選擇最佳文檔
best_doc, best_score, best_doc_id = top_n_docs[0]
logging.info(f"QID: {qid}, Selected Document ID: {best_doc_id}, Average Score: {best_score:.4f}")
#print(best_doc_id)
# 將結果添加到 answer_dict
answer_dict['answers'].append({"qid": qid, "retrieve": best_doc_id, "category": category})
```

