23-1 Capstone Design

- 최종 보고서 -

과목명	캡스톤디자인(2) 05분반
교수명	김진성
제출일	2023.06.16.
팀명	마지막 여름방학
프로젝트명	PaperMate

학 번	20191660	20163997	20174921
학 과	소프트웨어학과	컴퓨터공학과	소프트웨어학과
이 름	김인서	서희재	양종원

1. 소개

1) 배경

학술 연구를 수행하는 과정에서 실험과 논문 작성만큼이나 중요한 것이 발표이다. 발표를 통해 자신의 연구 내용을 공유하고, 다른 연구자들의 의견을 들음으로써 더 의미있고 발전된 연구가 가능하다. 하지만 발표를 위한 발표 자료 작성은 굉장히 수고로운 일이다. 대개 파워포인트 프로그램을 이용하여 다음과 같은 과정을 거친다.

1) 작성된 논문을 각 구획별로 나누어 슬라이드로 만든다.
2) 각 구획의 내용을 2~3문장으로 요약한다.
3) 논문에 있는 그림, 표, 그래프 등을 적절한 위치에 삽입한다.

이 과정은 긴 연구 내용을 핵심적인 2~3문장으로 요약하는 작업이 어려울 뿐만 아니라, 단순 반복 작업이 많이 포함되어 있어 시간이 오래 걸려 연구에 집중할 수 있는 시간을 많이 뺏기게 된다. 우리는 이러한 문제를 해결하기 위해, 논문을 입력하면 자동으로 발표 자료의 초안을 작성해주는 기능을 제공하고자 한다. 이를 통해 연구자들은 발표 자료 작성에 소요되는 시간을 절약하고, 더욱 집중적으로 연구에 전념할 수 있다

2) 기능

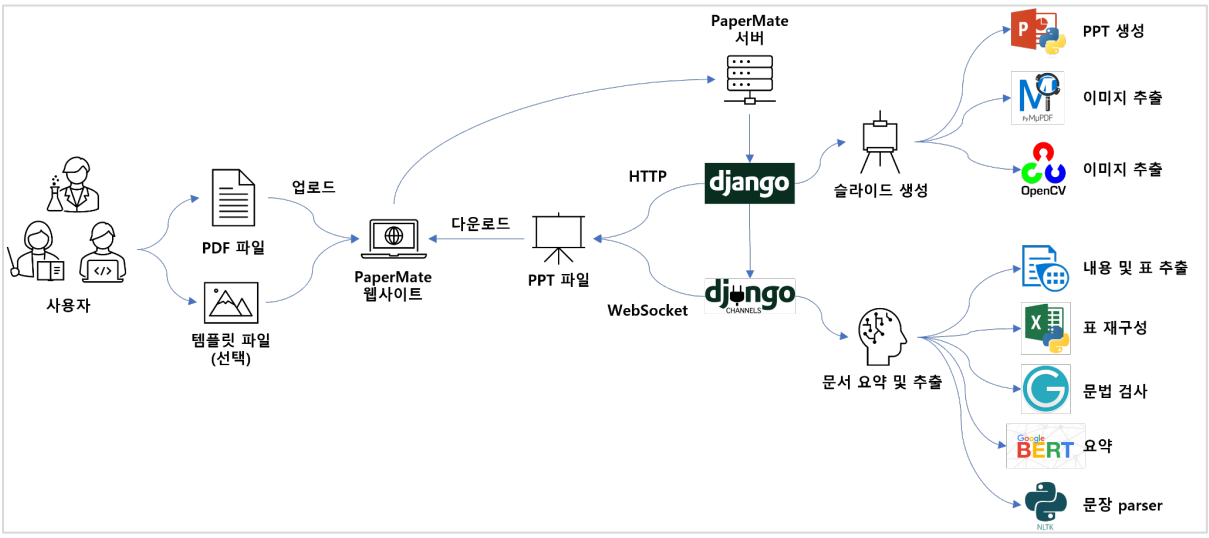
- 논문 요약
- ppt 초안 작성 자동화
    - 기본 템플릿 적용 기능
    - 사용자 템플릿 적용 기능
- ppt option 조정 기능

  조정할 수 있는 option의 종류는 아래와 같다.
    - 제목
    - 유저 이름
    - 제목 글씨체
    - 부제목 글씨체
    - 본문 글씨체

- ○ 자간
- ○ wide layout 여부
- ○ user template 사용 여부

● ppt 다운로드 기능

3) 서비스플로우



a) 사용자가 논문 파일을 papermate 웹사이트에 업로드 한다.

b) 사용자가 papermate 웹사이트에 템플릿을 선택하거나 업로드한다.

c) 해당 정보는 paperMate 서버로 가서 자연어처리를 이용하여 요약한다.

d) 요약된 정보를 바탕으로 슬라이드를 생성한다.

e) 슬라이드는 다시 paperMate 웹사이트를 통해 사용자에게 전달된다.

f) 사용자는 option을 변경해서 다시 슬라이드를 생성하거나 만들어진 슬라이드를 다운로드 받는다.

4) 유사서비스 비교

<table>
<tr><td></td><td>beautiful.ai</td><td>ChatBA</td><td>Gamma</td><td>auxi</td></tr>
<tr><td rowspan="2">해당 서비스</td><td colspan="2">AI에게 자연어로 요청을 하면, 알아서 관련된 슬라이드를 생성
+ 디자인 봇, 논문 특화 서비스 아님</td><td rowspan="2">Add-in 형태,
키보드
입력만으로 모든
기능을 사용,
AI Recommenda
tion</td></tr>
<tr><td colspan="2">새로운 지식을 바탕으로 슬라이드를
자동으로 생성하기는 어려움</td><td>text import 가능</td></tr>
<tr><td>PaperMate</td><td>새로운 지식으로
생성 가능</td><td>새로운 지식으로
생성 가능,
여러가지 수정
옵션 제공</td><td>table, figure
적절한 위치에
자동으로 구성</td><td>이용자가
슬라이드의
내용을 직접
입력해야 하는
번거로움 최소화</td></tr>
</table>

5) 기대 효과

- 단순 반복작업이 많은 발표자료 생성 과정을 효율화하여 연구인력의 생산성 증대를 꾀한다.

2. 설계

1) 웹서비스 설계

a) 논문 업로드

i) 논문 업로드는 HTTP POST 요청을 통해 multipart/form-data 방식으로 PDF 파일을 업로드 한다.

ii) 업로드된 파일은 Django의 FileSystemStorage를 이용하여 서버에 파일 형태로 저장된다. 이 때, 중복된 파일명이 있을 시 자동으로 uniquifier를 append한다.

b) 논문 요약

i) 논문 요약은 pdf_to_text 함수를 통해 이루어진다. 이 함수에 대한 설계는 2) PDF to Text 에서 다룬다.

ii) 요약 과정은 시간이 오래 걸리기 때문에 HTTP 프로토콜은 부적합하다고 판단되어 WebSocket을 이용하였다.

iii) WebSocket 구현을 위해 Django Channels 라이브러리를 사용하였다.

iv) 해당 부분을 구현한 Consumer 클래스는 아래의 코드와 같다.

v) 또한 클라이언트 측에서 WebSocket 요청을 보내고 기다리는 JavaScript 코드는 아래와 같다.

```python
class PDFConsumer(WebsocketConsumer):
    def connect(self):
        self.accept()

    def disconnect(self, code):
        pass

    def receive(self, text_data):
        text_data_json = json.loads(text_data)
        filename = text_data_json['filename']
        name, _ = os.path.splitext(filename)
        print("start converting...")
        start = time.time()
        result, _ = pdf_to_text(settings.MEDIA_ROOT / filename, settings.MEDIA_ROOT / f'{name}.json')
        end = time.time()
        print(f"time elapsed: {end - start} sec")
        self.send(text_data=f'{name}.json')
```

```javascript
<script>
const filename = JSON.parse(document.getElementById('filename').textContent);

const ws = new WebSocket(
    'ws://'
    + window.location.host
    + '/ws/process-pdf/'
    + filename
    + '/'
);

ws.onmessage = function(e) {
    alert('완료되었습니다!');
    window.location.pathname = '/p2s/step-2/' + e.data;
};

ws.onopen = function() {
    console.log("ws sending...")
    ws.send(JSON.stringify({
        'filename': filename
    }));
    console.log("ws sent")
}
```

c) 옵션 조정

i) 옵션 조정 단계 직전 및 옵션 조정 단계에서는 generate_slide 함수를 통해 PPT 파일을 생성한다. genrate_slide 함수에 대한 설계는 6) 에서 다룬다.

ii) 생성된 PPT 파일을 웹상에서 미리 보여주기 위해 임시로 PDF 파일로 변환을 거친 뒤, 클라이언트에서 iframe을 이용하여 보여준다.

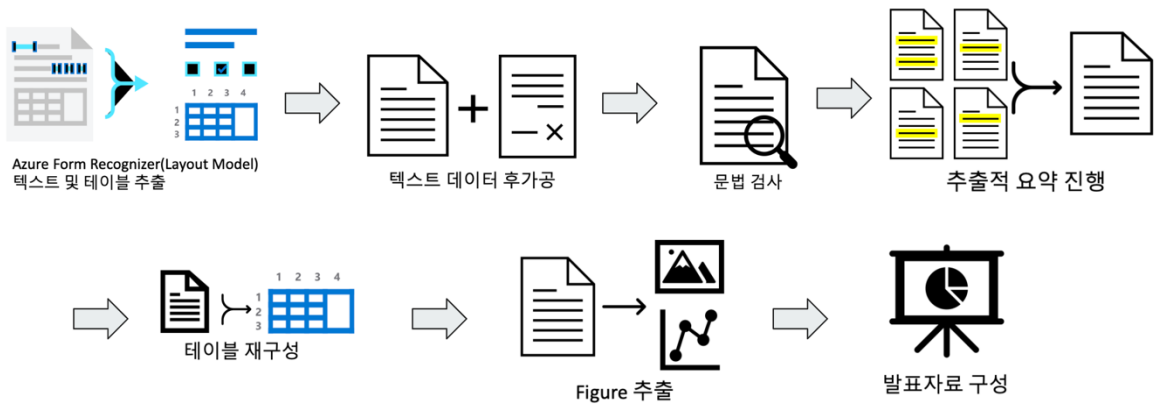
iii) 이 때, Windows의 COM object 접근을 위해 win32com 라이브러리를 사용하였다.

d) 다운로드

i) 웹 상에서 사용자가 생성된 PPT 파일을 다운로드 할 수 있도록 하기 위해, Response의 Content-Disposition을 attachment로 설정하여 반환하도록 하였다.

```
mime_type, _ = mimetypes.guess_type(fl_path)
response = HttpResponse(fl, content_type=mime_type)
response['Content-Disposition'] = "attachment; filename=%s" % filename
return response
```

Input PDF를 받은 이후부터의 **PDF to PPT workflow



2) PDF to Text

a) PDF 텍스트, 테이블 추출

- Microsoft의 Azure Form Recognizer 활용

```
def extract_data(path):
    source = path
    form_recognizer_client = DocumentAnalysisClient(ENDPOINT, AzureKeyCredential(API_KEY))
    with open(source, "rb") as f:
        data_bytes = f.read()
    poller = form_recognizer_client.begin_analyze_document("prebuilt-layout",document=data_bytes)#,content_type='application/pdf')

    form_result = poller.result().to_dict()
    return form_result
```

- output 예시:


```
 {'role': 'title',
  'content': 'Have my arguments been replied to? Argument Pair Extraction as Machine Reading
Comprehension',
  'bounding_regions': [{'page_number': 1,
    'polygon': [{'x': 1.315, 'y': 1.008},
     {'x': 6.9514, 'y': 1.008},
     {'x': 6.9514, 'y': 1.408},
     {'x': 1.315, 'y': 1.408}]}],
  'spans': [{'offset': 2, 'length': 92}]},
```

```
 {'role': None,
  'content': 'Argument pair extraction (APE) aims to au- tomatically mine argument pairs
from two in- terrelated argumentative documents. Existing studies typically identify
argument pairs indi- rectly by predicting sentence-level relations be- tween two documents,
neglecting the modeling of the holistic argument-level interactions. To- wards this issue,
we propose to address APE via a machine reading comprehension (MRC) framework with two
phases. The first phase employs an argument mining (AM) query to identify all arguments in
two documents. The second phase considers each identified argu- ment as an APE query to
extract its paired arguments from another document, allowing to better capture the argument-
level interac- tions. Also, this framework enables these two phases to be jointly trained in
a single MRC model, thereby maximizing the mutual benefits of them. Experimental results
demonstrate that our approach achieves the best performance, outperforming the state-of-the-
art method by 7.11% in F1 score.',
  'bounding_regions': [{'page_number': 1,
    'polygon': [{'x': 1.2183, 'y': 3.3538},
     {'x': 3.7957, 'y': 3.3538},
     {'x': 3.7957, 'y': 7.1222},
     {'x': 1.2183, 'y': 7.1222}]}],
  'spans': [{'offset': 432, 'length': 1007}]},
```

```
{'role': 'sectionHeading',
 'content': '2 Related Work',
 'bounding_regions': [{'page_number': 2,
   'polygon': [{'x': 0.9869, 'y': 4.1809},
    {'x': 2.2196, 'y': 4.183},
    {'x': 2.2194, 'y': 4.2992},
    {'x': 0.9867, 'y': 4.2971}]}],
 'spans': [{'offset': 4815, 'length': 14}]},
```


- role은 크게 title, sectionHeading, None 세가지로 나뉨

b) 후가공

- OCR 기반의 데이터 추출으로, 부정확하게 추출된 텍스트가 많이 존재했고, 논문의 특성에 맞게 문제를 해결

문제점	해결 방안
문단 및 단락 끊김 발생	content로 인식된 부분을 끊김 없이 이어붙이기

불필요한 정보(figure, table의 text까지 추출) 포함	figure와 table에서 추출되는 text의 조건 파악(비교적 짧은 content) 및 내용 구성에서 배제
줄바꿈을 고려하지 못하기 때문에, 단어 끊김 발생(ex. oriented → ori- / ented)	문장에서 "- "을 제거
et al., i.e., e.g., cf., …, 소수점 등 온점이 문장에 포함되면 추후 임베딩 구성에 해를 끼침	줄임말은 다시 풀어서 쓰고, 소수점은 쉼표로 대체. 요약과 문장 분리 후 대체한 방법 그대로 원상 복구

c) 문법 검사

- Gingerit api
- 단어 끊김 문제를 해소하는 과정에서 발생하는 문제(원래부터 -가 포함된 단어의 경우에도 -가 제거)를 맞춤법 검사로 복구함

non-task-oriented → nontaskoriented(X) → (Gingerit) → non-task-oriented(O)

d) 테이블 재구성

- data의 형태로 전달되는 table 데이터를 통해 xlsx 테이블로 재구성

3) 요약

a) 요약 방식 선정

(1) 추출적 요약: 원본 텍스트에서 중요한 문장이나 구를 추출하여 요약하는 방식

(2) 생성적 요약: 기계가 원본 텍스트를 이해하고 새로운 문장을 생성하여 요약하는 방식

● 요약 방식 선정 이유(1)

	추출적 요약	**생성적 요약**
장점	**높은 정보의 정확성과 신뢰성** 문법적인 오류, 불일치 최소화 원본의 스타일과 톤 유지	다양한 스타일과 톤으로 요약 가능 상대적으로 더 긴 텍스트도 요약 가능 좀 더 사람과 같은 요약 스타일

	일관성과 상응성 유지	

→ 신뢰성 있는, 정확한 정보 전달이 핵심인 논문 발표에 가장 중요한 덕목

- 요약 방식 선정 이유(2)
  - facebook/bart-large-cnn → summarization SOTA급 모델
    - 문장이 끊기는 문제 발생
    - 논문과 맞지 않은 요약문 생성
  - SmartPy/bart-large-cnn-finetuned-scientific_summarize → 논문 전용으로 fine tuning(전문 - 초록)
    - 초록의 구조를 지나치게 따라감(this article ~, this paper~ 형식)
  - togethercomputer/GPT-JT-6B-v1
    - 정상적인 요약이 되지 않고, 들쭉날쭉한 성능을 보임

- 요약 방식 선정 - 추출적 요약

선정 모델

- Bert-extractive-summarizer(https://github.com/dmmiller612/bert-extractive-summarizer)

  > 특징

- 추출적 요약 모델
- Custom model 및 custom tokenizer를 담아서 사용 가능
  → Scientific Paper 특화로 학습된 'allenai/scibert_scivocab_cased'
  → 논문에서 사용되는 전문 용어 및 표현을 잘 인식(논문 특화 토큰화 + 요약)
- 적절한 요약 문장 수(elbow) 찾아서 요약 수행

4) Table 재구성

- output 데이터 구조 파악
- openpyxl을 활용해 테이블 재구성

5) Figure, Table 위치 찾기

- Figure와 Table을 적절한 위치에 삽입하기 위해 언급된 위치를 파악

6) Image 추출하기

- PyMuPDF의 fitz모듈을 이용하여 PDF문서의 배경을 없애고 각 페이지를 하나의 png 파일로 변환함

```
for i, page in enumerate(doc):
    pix = page.get_pixmap(matrix=fitz.Matrix(300/72, 300/72), dpi=None,
                          colorspace=fitz.csRGB, clip=True, alpha=True, annots=True)
    pix.save(wholepage_dir + f"\\samplepdfimage-%03d.png" %
             page.number)  # save file
```

- openCV를 이용하여 이미지 crop
    - cvtColor를 이용하여 각 페이지 이미지를 gray scale이미지로 변환
    - threshold를 이용하여 이미지 이진화
    - findContours로 외곽선 검출

```
gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

_, binary_image = cv2.threshold(
    gray_image, 100, 255, cv2.THRESH_BINARY)

# 외곽선 검출
contours, _ = cv2.findContours(
    binary_image, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
```

    - 외곽선의 너비와 높이가 일정 크기 이상인 경우, 해당 영역을 cropped 이미지로 저장함

```
# 흰색 사각형 영역 검출 및 자르기
for contour in contours:
    x, y, w, h = cv2.boundingRect(contour)
    if w > width//9 and h > 10:  # 흰색 사각형으로 인정할 최소 너비와 높이 설정
        cropped_image = image[y-5:y+h+5, x-5:x+w+5]
        cv2.imwrite(cropped_dir + "\\" + "figure_" +
                    str(n+1) + ".png", cropped_image)
        n += 1
```

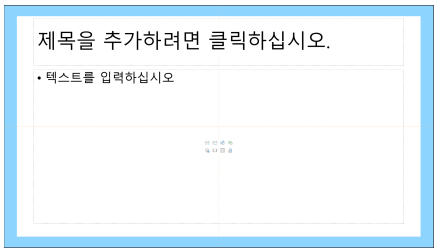
7) 슬라이드 구성하기

- 기본적인 구성은 [표지-목차-서브섹션 페이지-내용-QnA페이지] 구조
- 각 서브섹션은 text, text with figure, text with table, text with both로 나뉜다 .

- def just_insert_text(presentation, title, summary_seq, option):
  일단 모든 내용을 담은 하나의 페이지를 만들어 본 후 폰트 크기를 측정.

측정한 폰트 크기가 24보다 작으면 -> n개로 나눠서 페이지 구성

이를 통해 페이지 내 글씨가 너무 작아지는 현상을 방지

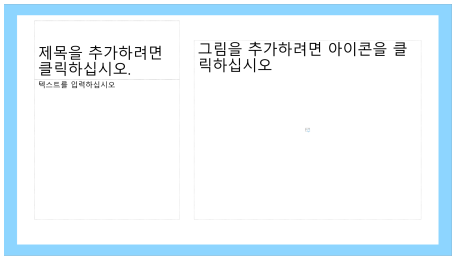


- def insert_text_with_picture(presentation, title, summary_seq, picture_seq, option):

  문장 리스트를 순서대로 일반 문장, figure가 있는 문장으로 구분한다.

  일반 문장은 just_insert_text 함수를 호출한다.

  figure가 있는 문장은 "picture with caption" layout을 사용하여 layout내 shape이라는 객체를 채워가며 슬라이드를 구성한다.



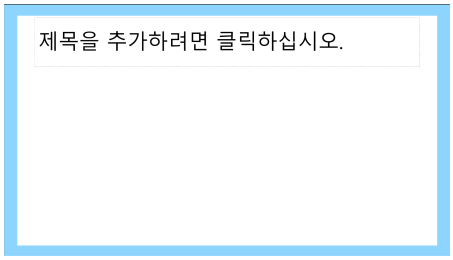
- def insert_text_with_table(presentation, title, summary_seq, table_seq, option):

  문장 리스트를 순서대로 일반 문장, table이 있는 문장으로 구분한다.

  일반 문장은 just_insert_text 함수를 호출한다.

  table이 있는 문장은 " title and content" layout을 사용하여 layout을 구성한다.

  이때, table의 cell을 순회하면서 cell의 height의 변화를 감지 -> 변화가 있다면, 줄바꿈을 하고 있다는 뜻이므로, table의 width를 1.15배해도 layout 밖으로 넘어가지 않는다면, table의 width를 1.15배 한다.

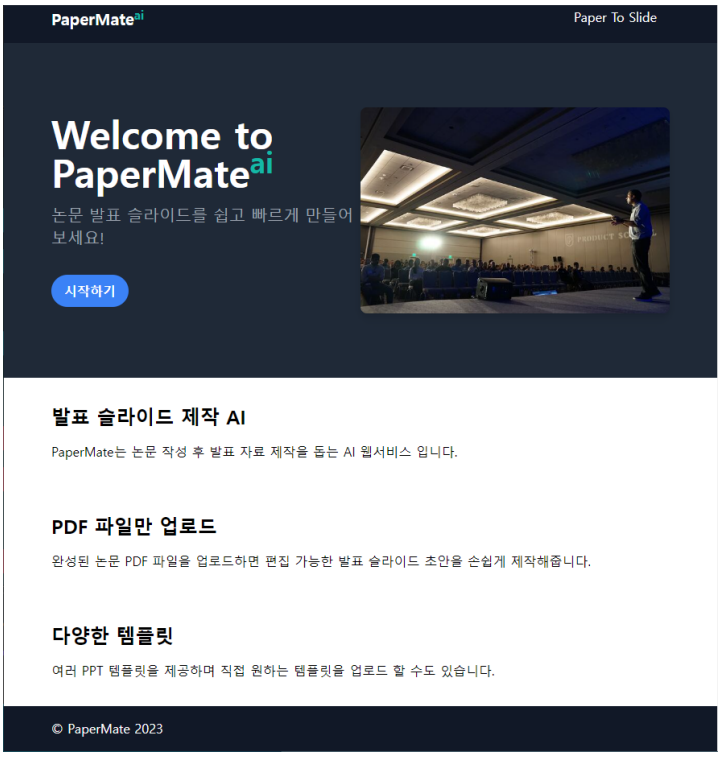


- def insert_text_with_both(presentation, title, summary_seq, figure_seq, option):
  문장 리스트를 순서대로 일반문장, figure가 있는 문장, table이 있는 문장으로 구분해간다.
  일반 문장을 just_insert_text함수를 호출한다.
  figure가 있는 문장은 insert_text_with_picture 함수를 호출한다.
  table이 있는 문장은 insert_text_with_table 함수를 호출한다.

- 각 단계별로 페이지를 구성할 때, option에 해당하는 값들을 적용해가면서 구성한다.
- 맨 마지막에는 template을 적용하여 마무리한다.

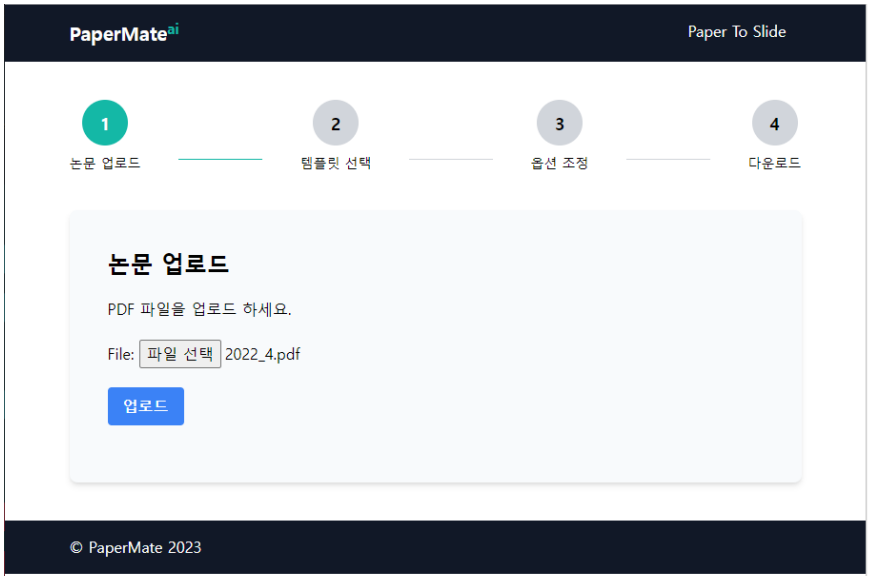
3. 구현 및 시연

1) 메인 화면



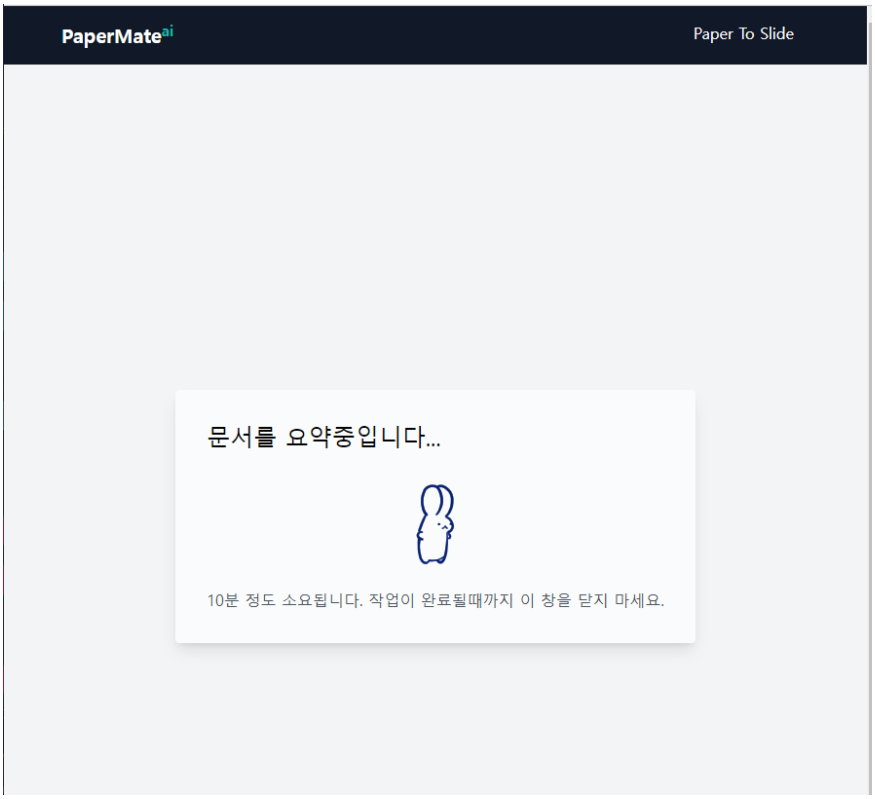
메인 화면에서 시작하기를 누르면 서비스를 시작할 수 있습니다.

2) PDF 업로드



파일 선택 버튼을 누르고 원하는 논문 pdf 파일을 선택한 후, 업로드를 누르면 요약을 시작합니다.

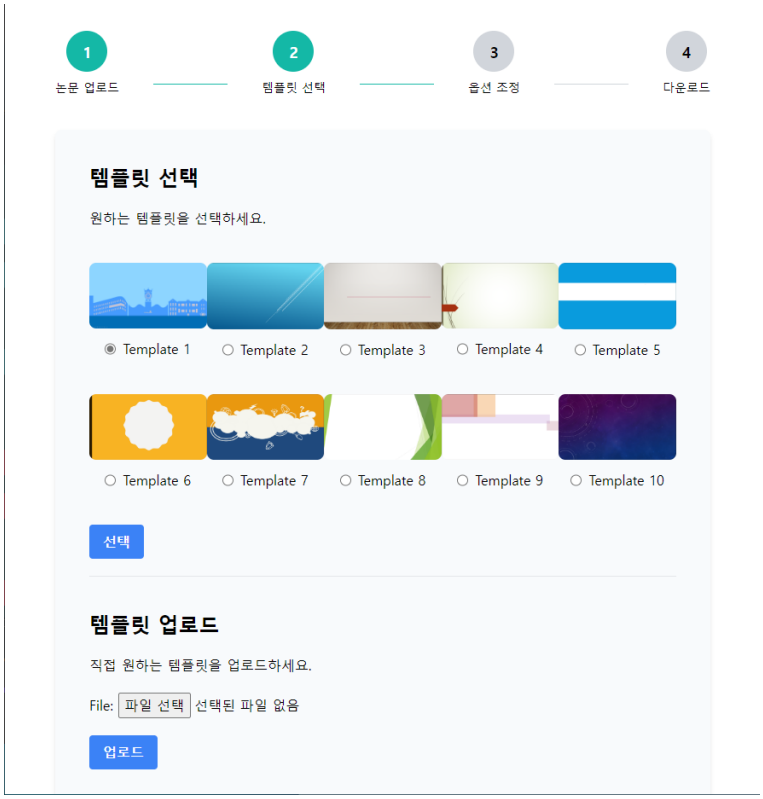
3) 요약 진행



논문의 길이에 따라 5-10분 가량 소요됩니다.

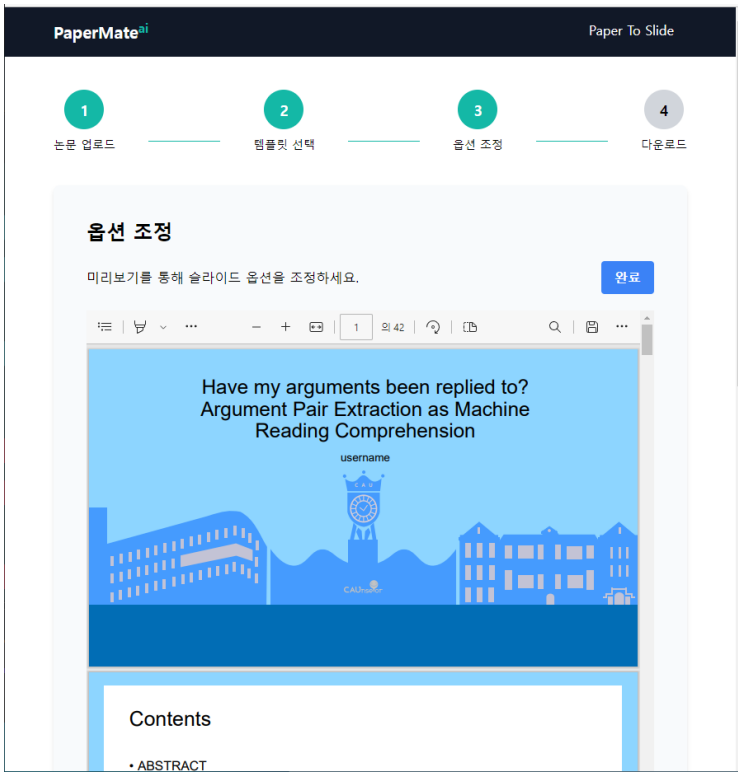
완료 시 팝업으로 완료를 알립니다.

4) 템플릿 선택 / 업로드



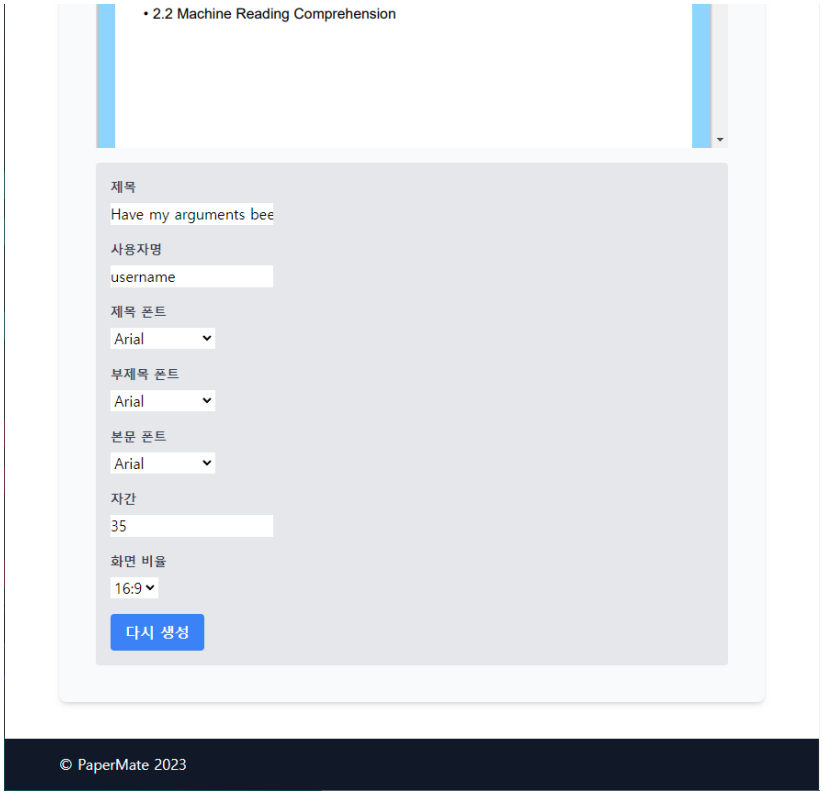
기본 제공된 템플릿을 선택하거나, 아래의 템플릿 업로드 기능을 통해 .potx 형식의 템플릿을 업로드 하여 ppt를 생성할 수 있습니다.

5) PPT 생성



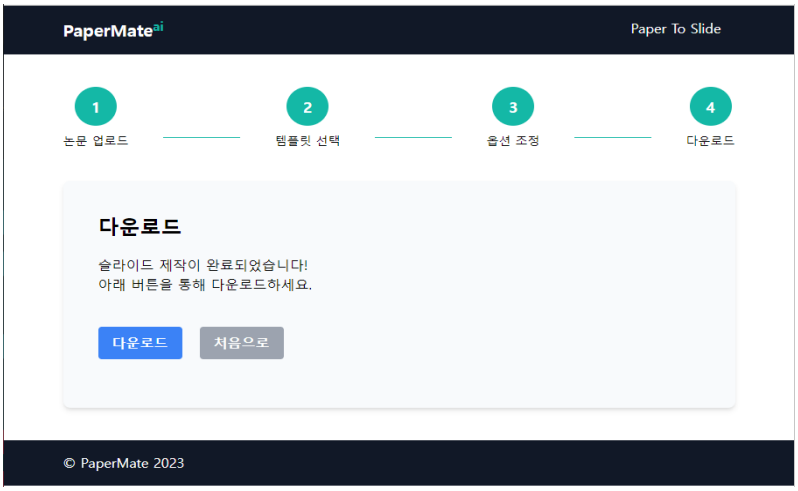
선택한 템플릿에 맞게 요약 및 추출된 내용과 데이터에 따라서 ppt를 생성하고, 미리보기가 가능합니다.

6) 옵션 조정



제목, 사용자명, 각각의 폰트들, 자간, 화면 비율을 조정할 수 있습니다. 다시 생성 버튼을 클릭시 수정한 내용을 반영해 ppt가 재생성 됩니다.

7) 다운로드



완성된 발표 자료를 다운로드 받을 수 있습니다. 수정 가능한 pptx 형태로 제공됩니다. 처음으로를 선택해 최초 화면으로 돌아갈 수 있습니다.

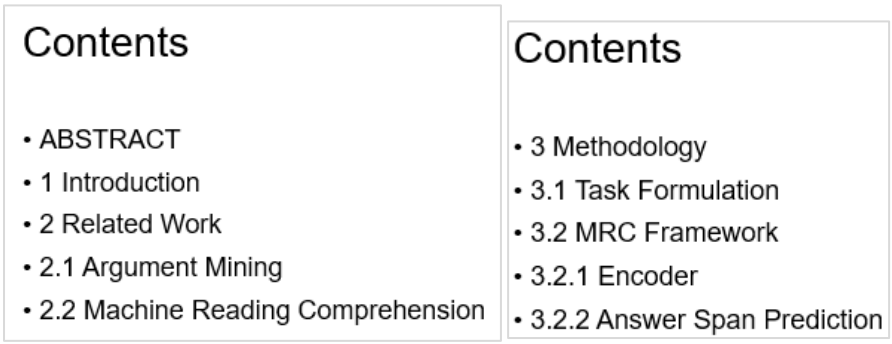
4. 성능 검증

*설명을 위해 일부분을 발췌해서 비교하였음.

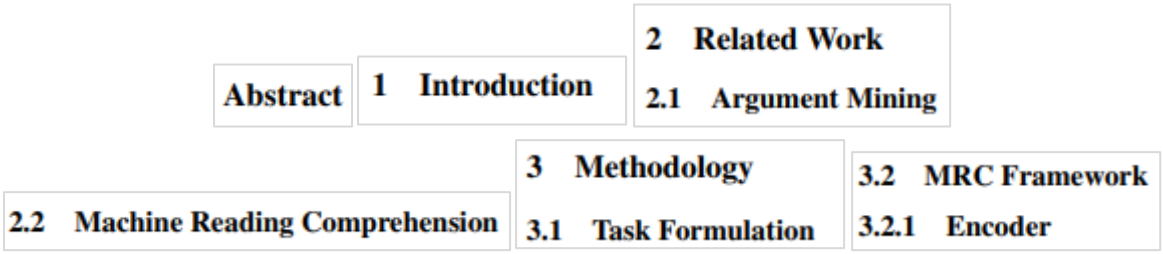
1) 제목 추출

Main Title, ABSTRACT부터 시작해, 1. Introduction, 2. Related Work 부터 Acknowledgments까지, 그리고 2.1 Argument Mining 등의 소제목들까지 잘 추출해오는 것을 확인할 수 있다.

(위) 추출 (아래) 논문 캡쳐



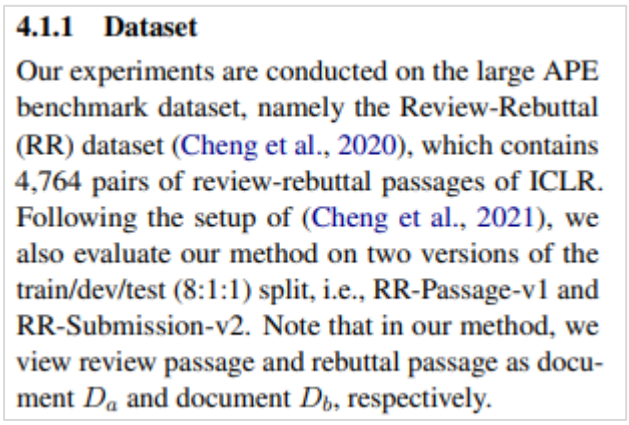
(위) 추출 (아래) 논문 캡쳐



2) 내용 검증

4.1.1 Dataset 단락의 핵심 내용인 어떤 데이터셋을 사용했는지가 담긴 문장을 올바르게 추출했다.

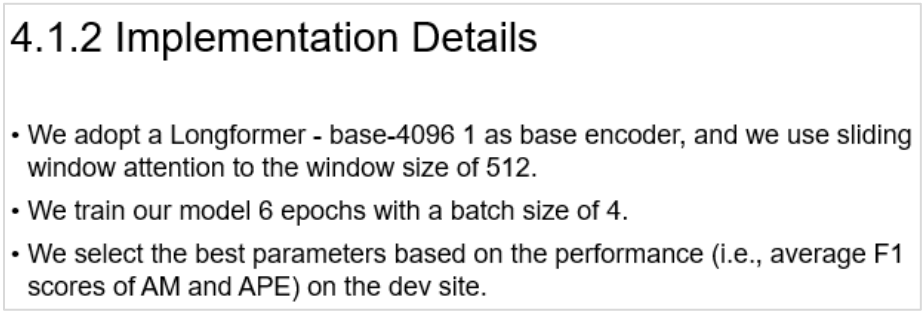
(위) 추출 (아래) 논문 캡쳐



4.1.2 Implementation Details에서 사용한 base encoder 모델, window size, batch size, learning rate 등 비교적 짧은 단락이지만 핵심적인 세 개의 문장을 추출해 온 것을 확인할 수 있다.

논문 특화로 학습된 모델을 사용하고, 적절한 요약 문장 수를 찾아서 요약하는 기술이 가장 큰 효과를 발휘한 파트이다.

또한, 사이에 끼어있는 불필요한 주석은 추출 시에는 단락에 포함되는 것처럼 취급되었는데, 적절한 후가공으로 단락에서 제외시켰다.



(위) 추출 (아래) 논문 캡쳐

4.1.2 Implementation Details

We adopt Longformer-base-4096 [1] as base encoder, and we use sliding window attention with the window size of 512. We train our model 6 epochs with a batch size of 4. AdamW (Kingma and Ba, 2015) is used as the optimizer, and the learning rates for Longformer and other layers are 1e-5 and 1e-3.[2]

The evaluation metrics contain two aspects, namely AM and APE. Different from (Cheng et al., 2021, 2020), sentence pairing is not included as a metric because we extract argument pairs directly.

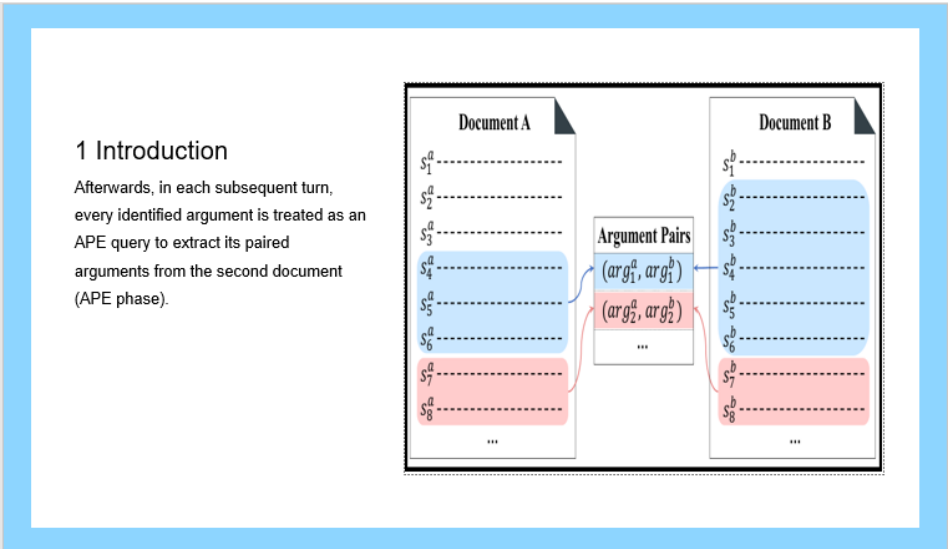
[1] https://huggingface.co/allenai/longformer-base-4096

[2] Our source code is available at https://github.com/HLT-HITSZ/MRC_APE

We select the best parameters based on the performance (i.e., average F_1 scores of AM and APE) on the dev set. All scores are averaged across 5 distinct trials using different random seeds.

3) Figure, Table 삽입 위치

1 Introduction 파트에서 언급된 Figure 1이 정상적으로 Introduction 파트에 삽입된 것을 확인할 수 있다.



(위) 추출 (아래) 논문 캡쳐

Abstract

Argument pair extraction (APE) aims to automatically mine argument pairs from two interrelated argumentative documents. Existing studies typically identify argument pairs indirectly by predicting sentence-level relations between two documents, neglecting the modeling of the holistic argument-level interactions. Towards this issue, we propose to address APE via a machine reading comprehension (MRC) framework with two phases. The first phase employs an argument mining (AM) query to identify all arguments in two documents. The second phase considers each identified argument as an APE query to extract its paired arguments from another document, allowing to better capture the argument-level interactions. Also, this framework enables these two phases to be jointly trained in a single MRC model, thereby maximizing the mutual benefits of them. Experimental results demonstrate that our approach achieves the best performance, outperforming the state-of-the-art method by 7.11% in F_1 score.

1 Introduction

As a salient part of argument mining (AM), the analysis of dialogical argumentation has received increasing research attention (Morio and Fujita, 2018; Chakrabarty et al., 2019; Ji et al., 2021; Cheng et al., 2021; Yuan et al., 2021). Argument pair extraction (APE), proposed by Cheng et al. (2020), is a new task within this field that focuses on extracting interactive argument pairs from two interrelated documents (e.g., peer reviewer and rebuttal). Figure 1 presents an example of APE where two interrelated documents are segmented into arguments and non-arguments at sentence level. Two arguments from different documents that discuss the same issues are regarded as an argument pair.

*Equal Contribution
†Corresponding Author

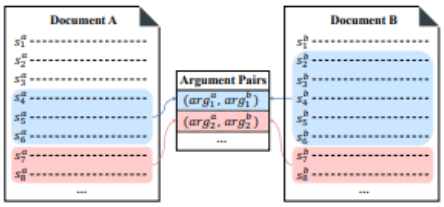


Figure 1: A simplified example of APE task, where each dashed line in the two documents denotes a sentence. s_j^i is the j-th sentence in document i, and arg_j^i is an argument in the j-th argument pair from document i. Sentences without colors indicate non-arguments, while sentences covered by colors can form arguments. Two arguments with the same color are regarded as an argument pair.

Previous works (Cheng et al., 2020, 2021) commonly address APE by decomposing it into two sentence-level subtasks, i.e., a sequence labeling task and a sentence relation classification task. These methods identify arguments by sentence-level sequence labeling and determine whether two sentences belong to the same argument pair by sentence relation classification. Afterwards, the argument pairs are inferred indirectly by certain rules combining the results of the two subtasks. However, such a paradigm only considers sentence-level relations, while the holistic argument-level relations can not be well modeled.

In this paper, we argue that APE can be considered as a multi-turn machine reading comprehension (MRC) task with two phases, i.e., an AM phase and an APE phase. Specifically, in the first turn, a special AM query is employed to identify all the arguments in the first document (AM phase). Afterwards, in each subsequent turn, every identified argument is treated as an APE query to extract its paired arguments from the second document (APE phase). Similarly, this process can also be performed in another direction, that is, using the

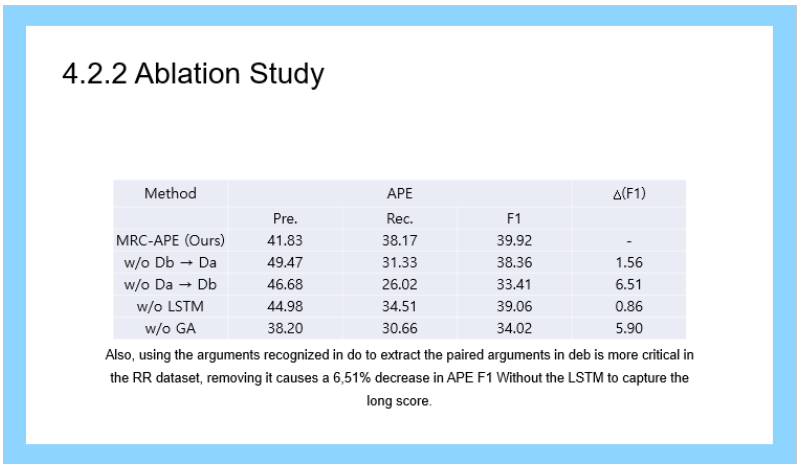
4.2.1 Main Results

Data	Methods	Argument Mining			Argument Pair Extraction		
		Pre.	Rec.	F1	Pre.	Rec.	F1
RR-Submission-v2	PL-H-LSTM-CRF	67.02	68.49	67.75	19.74	19.13	19.43
	MT-H-LSTM-CRF	70.74	69.46	70.09	27.24	26.00	26.61
	MLMC	69.53	73.27	71.35	37.15	29.38	32.81
	MRC-APE-Bert	73.36	68.35	70.77	42.26	34.06	37.72
	MRC-APE-Sep.	72.45	71.58	72.01	41.09	36.99	38.93
	MRC-APE (Ours)	71.83	73.05	72.43	41.83	38.17	39.92
RR-Passage-v1	PL-H-LSTM-CRF	73.10	67.65	70.27	21.24	19.80	20.22
	MT-H-LSTM-CRF	71.85	71.01	71.43	30.08	29.55	29.81
	MLMC	66.79	72.17	69.38	40.27	29.53	34.07
	MRC-APE-Bert	66.81	69.84	68.29	34.70	35.53	35.11
	MRC-APE-Sep.	75.27	67.90	71.39	36.63	40.05	38.26
	MRC-APE (Ours)	76.39	70.62	73.39	37.70	44.00	40.61

Besides, our MRC-APE achieves better results than MRC-APE-Sep. on both AM and APE tasks, indicating that joint training two phases in a single MRC model could maximize the mutual benefits of the two phases.

4.1.2 Main Results 파트에서 언급된 Table1도 적절한 위치에 삽입되었다. 언급된 문장 이후의 위치에 삽입한 이유는, 언급했을 때는 개괄적인 설명을 하고, 테이블 내용에 대한 구체적인 언급들은 대부분 이후

파트에서 이루어지기 때문이다.



4.2.2 Ablation Study

Method	APE			Δ(F1)
	Pre.	Rec.	F1	
MRC-APE (Ours)	41.83	38.17	39.92	-
w/o Db → Da	49.47	31.33	38.36	1.56
w/o Da → Db	46.68	26.02	33.41	6.51
w/o LSTM	44.98	34.51	39.06	0.86
w/o GA	38.20	30.66	34.02	5.90

Also, using the arguments recognized in do to extract the paired arguments in deb is more critical in the RR dataset, removing it causes a 6,51% decrease in APE F1 Without the LSTM to capture the long score.

4.2.2 Ablation Study에도 마찬가지로 적절하게 Table 2가 삽입되었다.

Data	Methods	Argument Mining			Argument Pair Extraction		
		Pre.	Rec.	F_1	Pre.	Rec.	F_1
RR-Submission-v2	PL-H-LSTM-CRF	67.02	68.49	67.75	19.74	19.13	19.43
	MT-H-LSTM-CRF	70.74	69.46	70.09	27.24	26.00	26.61
	MLMC	69.53	**73.27**	71.35	37.15	29.38	32.81
	MRC-APE-Bert	**73.36**	68.35	70.77	42.26	34.06	37.72
	MRC-APE-Sep.	72.45	71.58	72.01	41.09	36.99	38.93
	MRC-APE (Ours)	71.83	73.05	**72.43**	**41.83**	**38.17**	**39.92**
RR-Passage-v1	PL-H-LSTM-CRF	73.10	67.65	70.27	21.24	19.30	20.22
	MT-H-LSTM-CRF	71.85	71.01	71.43	30.08	29.55	29.81
	MLMC	66.79	**72.17**	69.38	**40.27**	29.53	34.07
	MRC-APE-Bert	66.81	69.84	68.29	34.70	35.53	35.11
	MRC-APE-Sep.	75.27	67.90	71.39	36.63	40.05	38.26
	MRC-APE (Ours)	**76.39**	70.62	**73.39**	37.70	**44.00**	**40.61**

Table 1: Main results on RR-Submission-v2 and RR-Passage-v1 (%). The best scores are in bold.

We select the best parameters based on the performance (i.e., average F_1 scores of AM and APE) on the dev set. All scores are averaged across 5 distinct trials using different random seeds.

4.1.3 Baselines

We compare our model with several baselines. **PL-H-LSTM-CRF** (Cheng et al., 2020) independently trains an argument mining task and a sentence pairing task, while **MT-H-LSTM-CRF** (Cheng et al., 2020) trains two subtasks in a multi-task framework. **MLMC** (Cheng et al., 2021) is an attention-guided model based on a table-filling approach, which is the current state-of-the-art method.

Furthermore, we implement two additional baselines. For a fair comparison with MLMC, **MRC-APE-Bert** replaces Longformer with Bert, where documents with excessively long length are splited into several segments. Instead of jointly training AM and APE phases, **MRC-APE-Sep.** trains the two phases separately.

4.2 Results and Analysis

4.2.1 Main Results

As shown in Table 1, our model achieves the best performance on both versions of the RR dataset. Concretely, on RR-Submission-v2, our model significantly outperforms the current state-of-the-art model MLMC by at least 7.11% in APE F_1 score. On RR-Passage-v1, our model obtains at least a 6.54% higher APE F_1 score than the MLMC. Also, our model achieves the best performance on AM. Furthermore, without applying Longformer as the base encoder, MRC-APE-Bert still outperforms MLMC in APE F_1 score, demonstrating that our improvement is not only brought by Longformer. However, for the AM task, MAC-APE-Bert achieves slightly lower F_1 score than MLMC. The reason may be that, in MLMC, the predictions of the AM task are influenced by the APE task through a complex attention interaction mechanism. However, our model does not require such a complex design and can achieve much better results on the APE task. Besides, our MRC-APE achieves better results than MRC-APE-Sep. on both AM and APE tasks, indicating that jointly training two phases in a single MRC model could maximize the mutual benefits of the two phases.

In addition, to analyze the error propagation from the first phase to the second phase, we use the true label of AM task to predict APE task. Under this setting, our model can achieve around 59.44% F_1 score for APE task, showing effectiveness in identifying argument pairs.

Method	APE			$\Delta(F_1)$
	Pre.	Rec.	F_1	
MRC-APE (Ours)	41.83	**38.17**	**39.92**	-
w/o $D_b \rightarrow D_a$	**49.47**	31.33	38.36	1.56
w/o $D_a \rightarrow D_b$	46.68	26.02	33.41	6.51
w/o LSTM	44.98	34.51	39.06	0.86
w/o GA	38.20	30.66	34.02	5.90

Table 2: The results of ablation experiments on RR-Submission-v2 (%). The best scores are in bold. *w/o* GA indicates that the global attention is not included in Longformer.

4.2.2 Ablation Study

The ablation study results are shown in Table 2. It can be observed that using two directions contributes greatly to our method. Also, using the arguments recognized in D_a to extract the paired arguments in D_b is more critical in the RR dataset, removing it causes a 6.51% decrease in APE F_1 score. Without the LSTM to capture the long-

6. 역할

김인서	이미지 추출, ppt슬라이드 구성, option format 구성, 요약 파일 format 구성
서희재	PDF 텍스트 및 테이블 추출, 텍스트 데이터 후가공, 논문 요약
양종원	웹서비스 구현, 서버 구현, 클라이언트 구현, 프로젝트 통합

7. Github

https://github.com/jongwonyang/paper-mate.git

8. 참고문헌

Jianzhu Bao, Jingyi Sun, Qinglin Zhu, and Ruifeng Xu. 2022. Have my arguments been replied to? Argument Pair Extraction as Machine Reading Comprehension. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 29–35, Dublin, Ireland. Association for Computational Linguistics.