YAI 15th toy project

Fake News Detection with Golden Evidence

기초심화 NLP 팀

이현서   양민석   조윤진   박성용
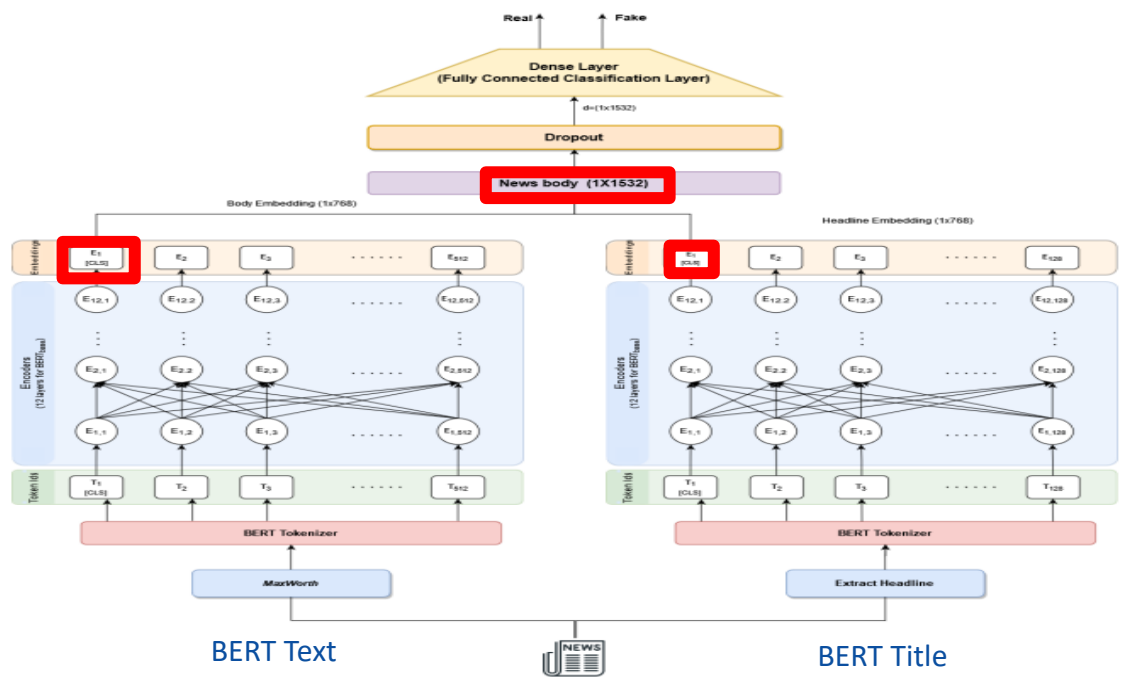
Index

Introduction

가짜 뉴스(Fake news)

"뉴스로 제시된 허위 또는 오해의 소지가 있는 정보" – Wikipedia

가짜 뉴스 판별 전략

- 뉴스 자체의 어투/패턴 : 다양한 데이터셋을 통해 학습

- 사실과의 일치여부 : 시의성 있는 사실 자료 없이는 판단이 어려움
  ⇒ Evidence based 접근

Background
Dual BERT

Kaggle Dataset

Other dataset

Background
LLM with RAG

```
You are responsible for evaluating the credibility of a news article.
Your task is to critically assess whether the given news aligns with facts, logi
c, and established patterns of misinformation.

### **Evaluation Criteria**:
1. **Lack of Factual Basis**: Does the article make claims without clear sources
or evidence?
2. **Logical Fallacies**: Are there exaggerated or unnatural cause-effect relatio
nships?
3. **Bias and Framing**: Does the article present a one-sided narrative while omi
tting counterarguments?
4. **Pattern Recognition**: Does the article resemble past misinformation or hoax
es?

### **Input**:
#### News Article:
"{news_text}"

#### Retrieved News:
{retrieved_news}

### **Assessment**:
- **1** = Completely False (No factual basis, likely misinformation)
- **5** = Uncertain (Some facts may be true, but unverified claims exist)
- **9** = Highly Plausible (Strong evidence and logical consistency)

Be as **conservative** as possible in your rating.
```
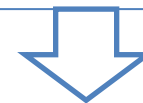
> 프롬프트

- Google에 title 기반 search
- 논문(Chen et.al)의 fake news 판단 기준 반영

=> 1-9점 척도로 출력, 가장 정확도 높은 역치 설정

1. 긴 title의 경우 search 결과가 나오지 않음
2. 긴 추론시간 : News 1개 당15-20초
3. API 비용 부과
4. 정해진 양식 따르지 않는 hallucination
5. 그럼에도 높은 성능 => 이유 설명 불가

```
Best Threshold: 7
   threshold  accuracy  precision   recall   f1_score  roc_auc
6          7  0.949896  0.946281  0.954167  0.950207  0.949887
```

> 모델 평가 결과

Chen, S., Huang, Y., & Dhingra, B. (2024). Real-time Fake News from Adversarial Feedback. *arXiv preprint arXiv:2410.14651*. https://arxiv.org/pdf/2410.14651v2

Objective

"신뢰성 있는 추론"
→ RAG(검색-증강 생성): 근거 문장 추출
→ BERT 기반 모델: LLM의 할루시네이션 문제를 회피
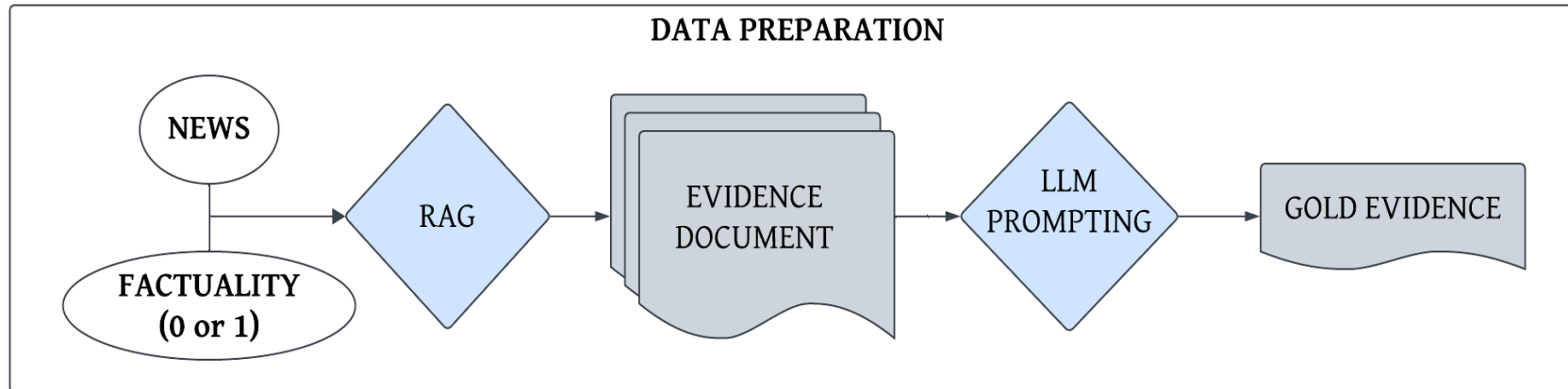
"가볍고 (계산 비용)효율적인 모델"
→ BERT 기반 모델

"해석 가능한 모델"
→ 핵심 근거 문장 추출 (golden evidence)
→ BERT 기반 모델 (LLM은 blackbox)

Data Preparation

Fake News Dataset

- 87175개 public NEWS dataset with Factuality label(0 or 1)

- Preprocessing : 소문자화, 구두점 제외 다른 문장부호/링크 제거, title이 있을 시 포함

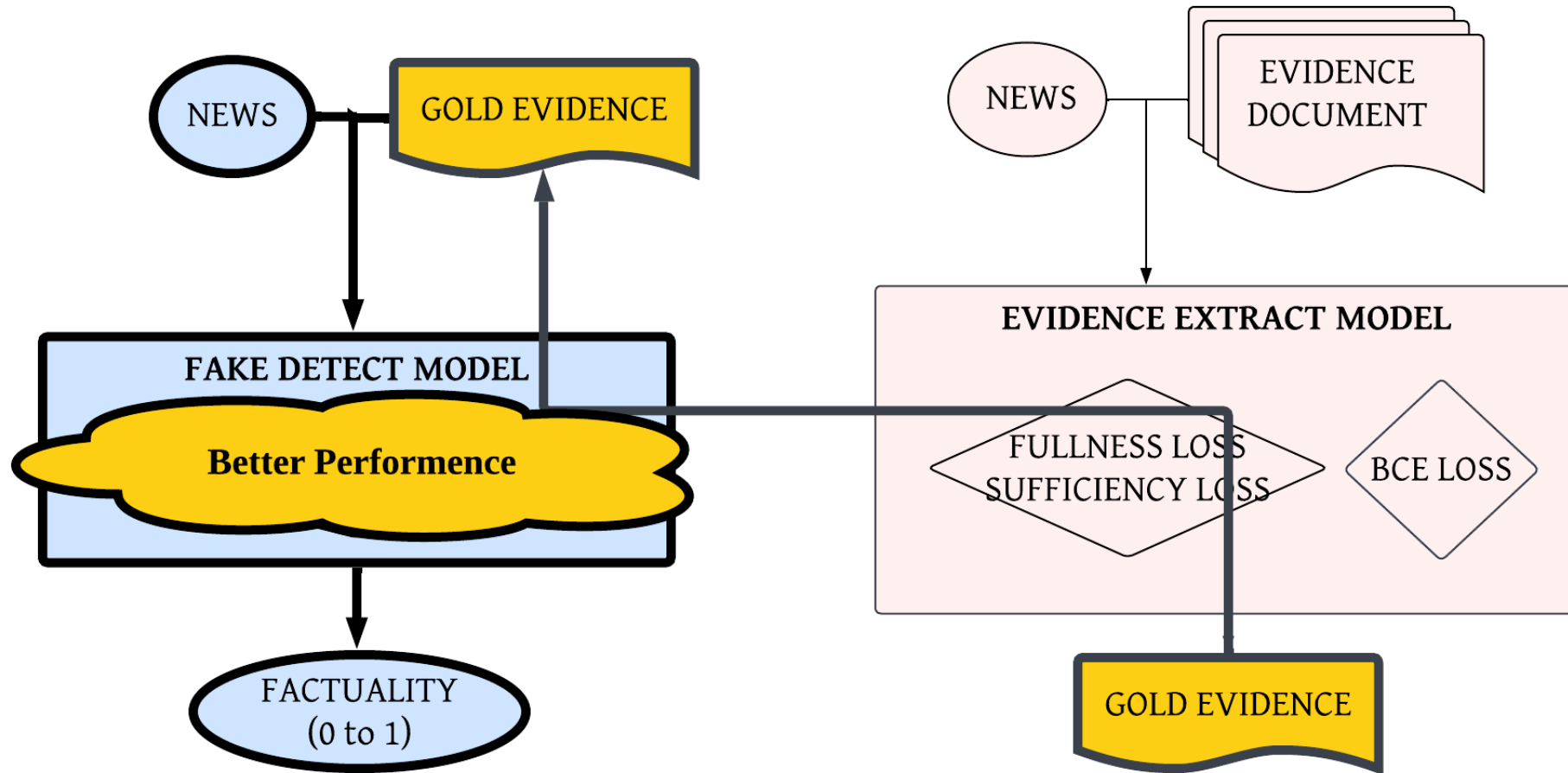- 87175개의 데이터 중 1336개의 데이터를 무작위로 추출하여 학습을 위한 evidence document와
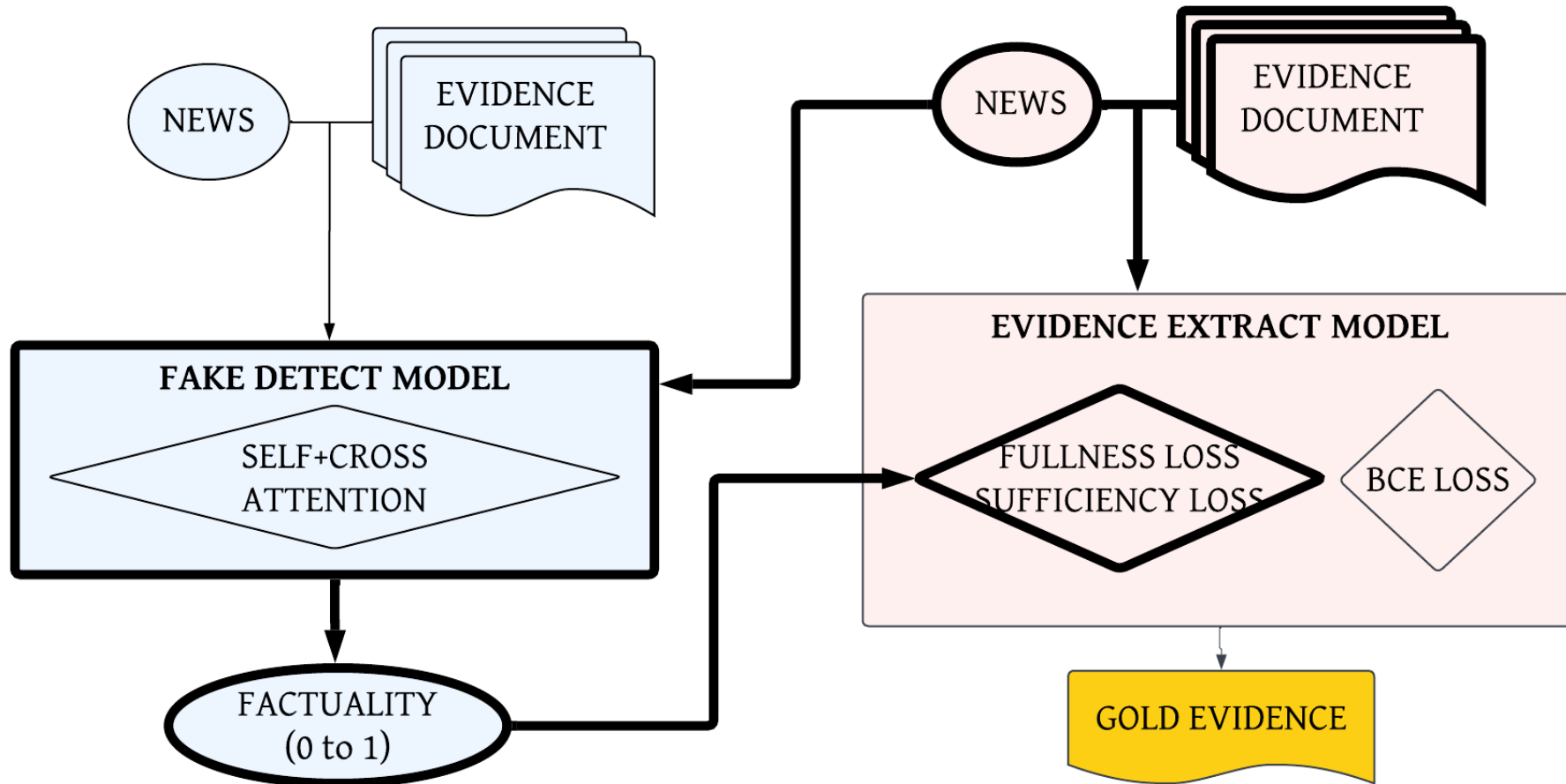 gold evidence를 준비

> Overall Flowchart

https://huggingface.co/datasets/BeardedJohn/FakeNews
https://www.kaggle.com/datasets/clmentbisaillon/fake-and-real-news-dataset
https://web.eecs.umich.edu/~mihalcea/downloads/fakeNewsDatasets.zip

> Data source

Model Overview

Training Architecture

Fake Detect Model

Model Architecture
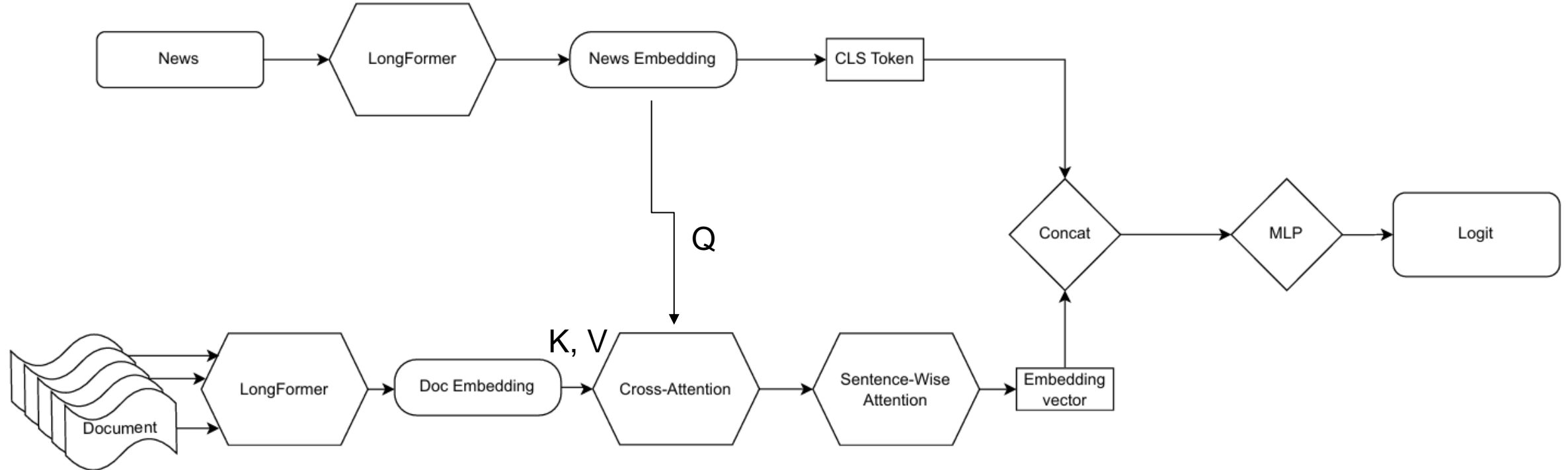
Evidence Extractor Model

Model Architecture

" [CLS] 'news' [SEP] 'evidence document' [SEP]   input

Bigbird-RoBERTa-base

[CLS] (size = 768)

MLP
768 -> 128 -> 40

[0, 1, 0, 0, 0, ... , 0, 0, 1]

1 = gold evidence sentence
0 = not a gold evidence sentence
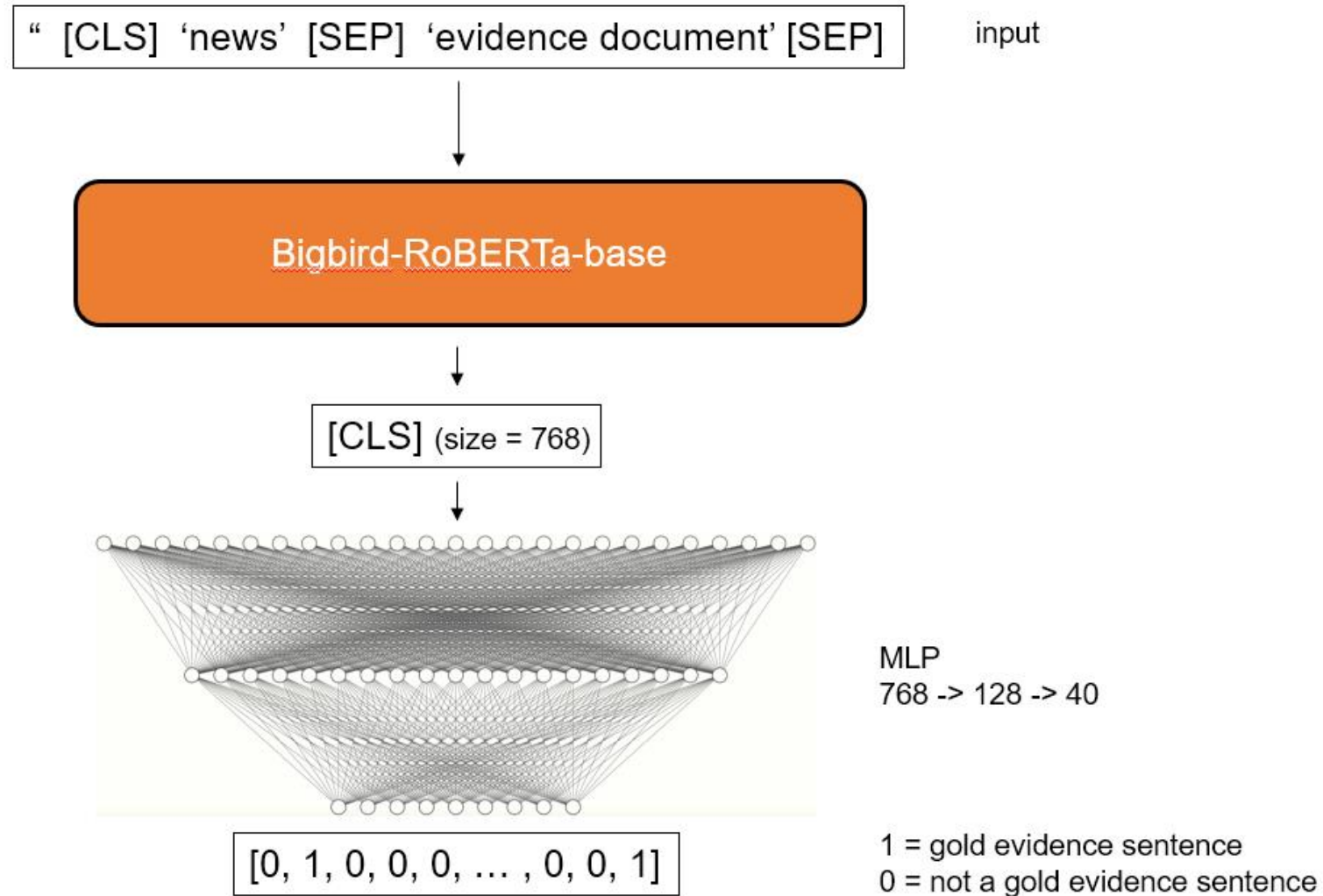
Evidence Extractor Model

Loss Function
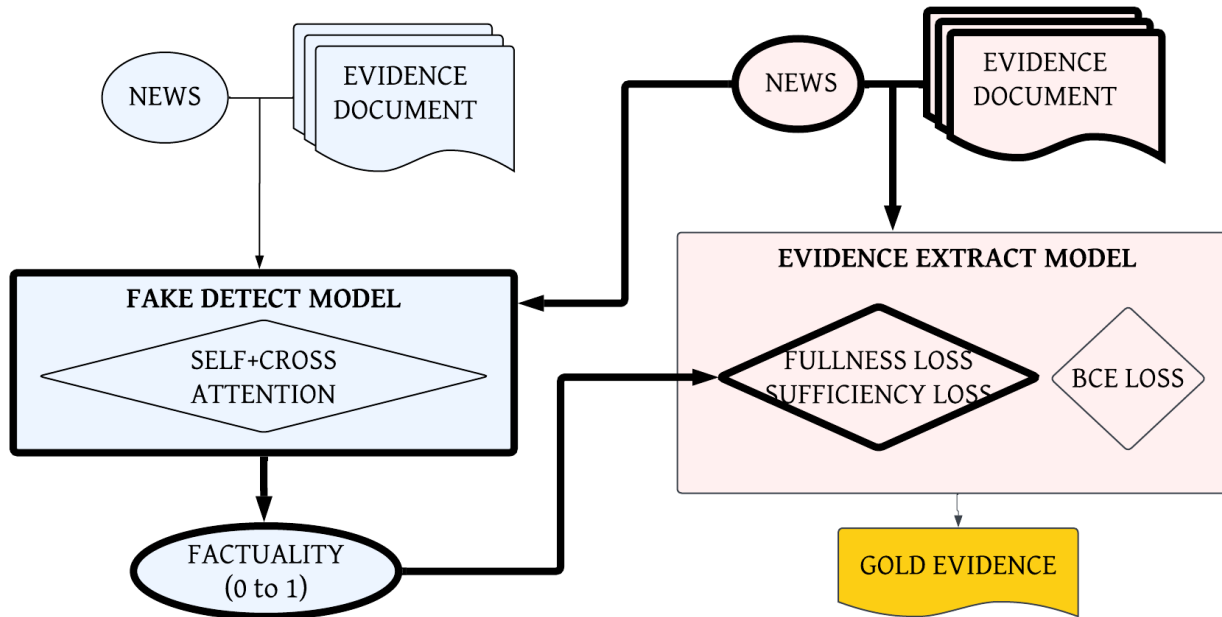
1. **Loss_fullness**

   - 전체 evidence doc 사용
     vs. golden evidence 제거 evidence doc 사용
   - $L_{\text{fullness}} = P_{\text{original\_doc}} - P_{\text{mani\_doc}}$

2. **Loss_sufficiency**

   전체 evidence doc 사용
     vs. golden evidence만 사용
   - $L_{\text{suff}} = P_{\text{original\_doc}} - P_{\text{mani\_doc}}$

3. **Final Loss (& BCELoss)**

   - $L = 0.7 * L_{BCE} + 0.3 * (L_{\text{suff}} - L_{\text{fullness}})$

Experiment & Result

Fake Detect Model

뉴스-증거 문장 관련성 가중치

학습 결과

Experiment & Result

Evidence Extractor Model

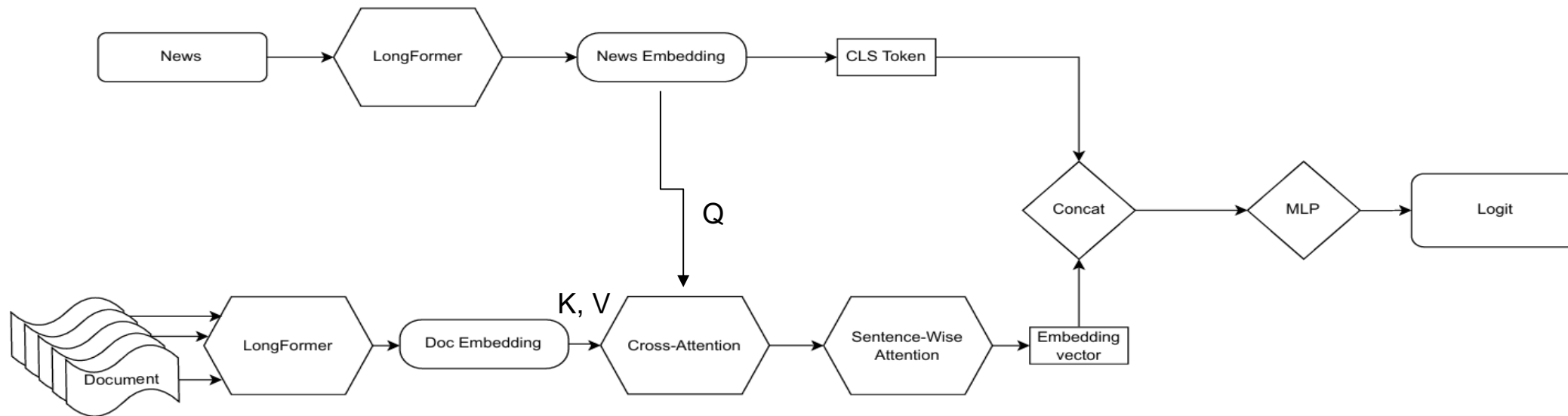주요 증거 문장(golden evidence) 추론 결과
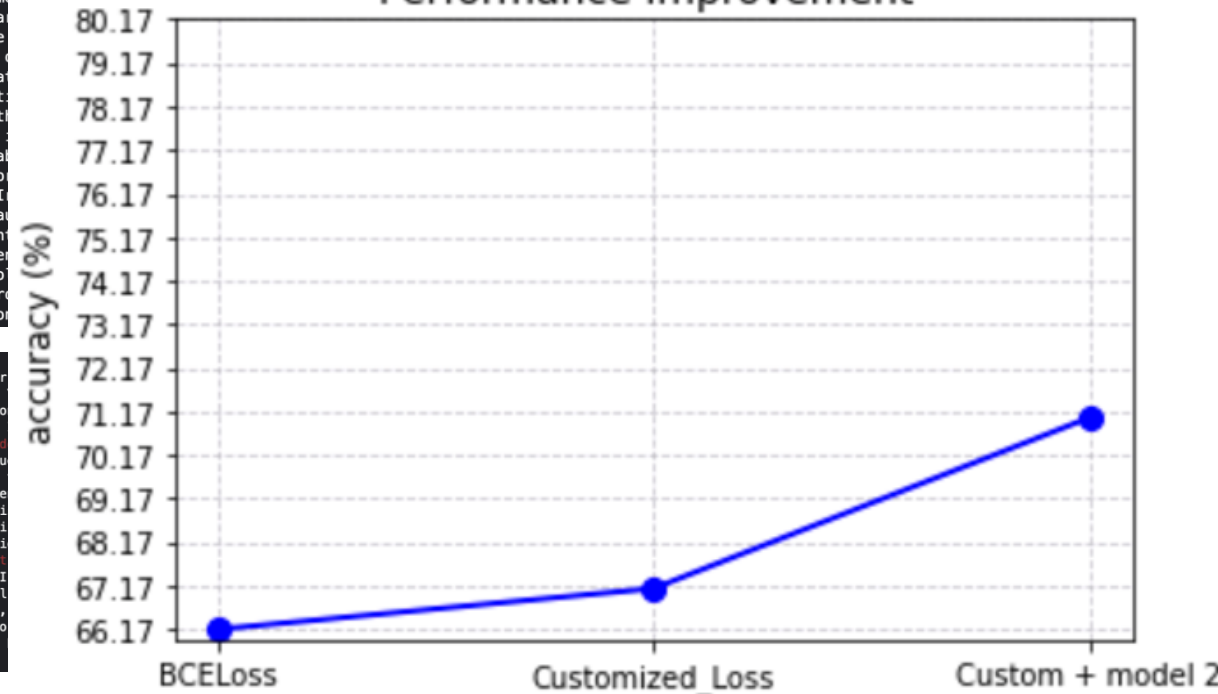
학습 결과

Summary

- 가짜 뉴스를 분류와 그 근거를 보여주는 모델 구조 구상하고 학습

- 신뢰성 있는 추론, 가볍고 효율적인 모델, 해석 가능한 모델 목표

- 뉴스 내용의 사실 확인을 위해 RAG를 통한 시의성 있는 사실 자료 이용

- Evidence Extractor Model과 Fake detect model, 두 모델을 이용하여 자료에서 중요한 문장들 (golden evidence)를 얻고 해당 문장들을 근거로 가짜 뉴스를 판별할 수 있도록 함.

- Evidence Extractor Model 학습 시에 Fake detect model를 이용하면 성능 향상이 가능함을 보임

Disscusion

Model2가 학습되지 않는 이유가 무엇인가?

- Document의 각 문장과 news 본문 전체가 모두 동일한 모델(LongFormer)를 이용함.

- cross-attention layer 하나의 cross attention만으로 구성됨

- news와 document간의 복잡한 관계: BERT의 표현 한계

16

LLM에 대한 BERT의 차별점

LLM을 대신 BERT 모델을 사용하는 것의 장점이 있는가?

- 신뢰도 있는 이해와 해석: LLM의 hallucination 문제

- 분류 태스크에서의 강점: LLM 한계 - 생성 모델 특성상, 세밀한 텍스트 패턴을 구별 하고 응답하는 능력이 fine-tuning된 RoBERTa보다 떨어질 가능성이 큼

- 적은 연산 자원에서 동작: LLM은 많은 메모리와 연산 비용이 필요

- mechanistic interpretability: LLM은 blackbox

하지만,

- LLM의 경량화: Distillation, Quantization, Efficient Architecture Design(sparse attention, LoRA)
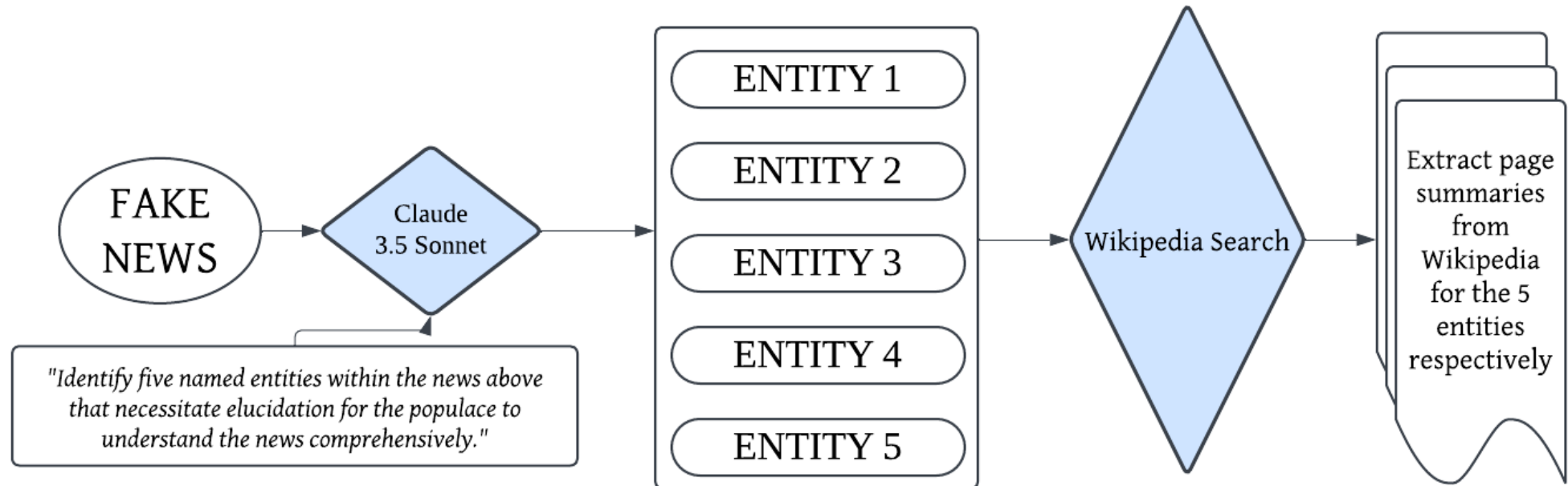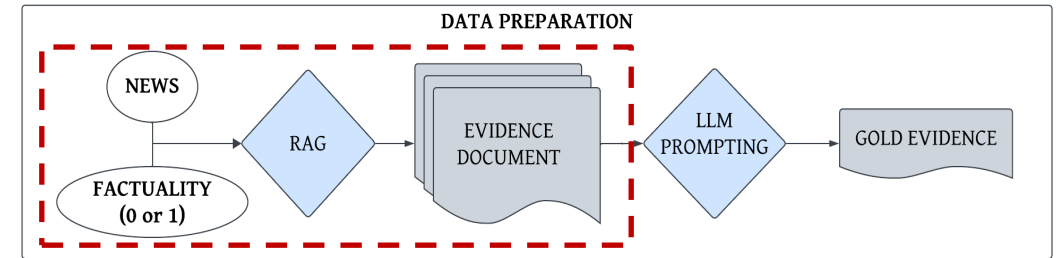
- 하드웨어 발전: 시간이 지날수록 GPU의 메모리, 연산 속도 증가

Thank You.

1) 위키피디아 검색을 활용한 뉴스 관련 Document 추출

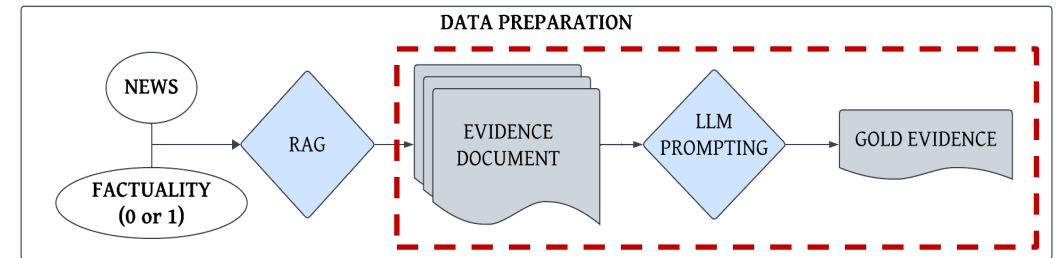a) Fake news 각각에서 중요한 키워드 5개 선정
b) 각 키워드(entity)를 위키피디아에 검색하여 상위 문서 요약

2) Noisy한 Evidence Document 중, 뉴스의 진위 판별에 중요한 'Gold Evidence' 추출

a)   Wiki summary 중 진위판별에 중요한 문장(gold evidence) 추출
b)   a)의 gold evidence를 포함한, wiki_summary의 30문장 선택
c)   Gold evidence의 <summarized evidence sentences>에서의 index 추출

- **Role-based Prompting**: "학생이 evidence sentence로 진위 판별을 할 수 있게 돕는 선생님"로 역할을 부여.
- **Few shot Prompting**: 모범적인 답변 예시를 제시함으로써 답의 일관성/성능을 높임.
- **Chain of Thought : 판단의 이유를 출력**하게 하여 모델이 논리적으로 사고할 수 있게 함.

1. News
2. Factuality Label
3. Evidence by wiki : [sen1,sen2, …, sen30]
4. Gold evidence Label : [0,0,1,0,…,1]

Appendix

➔ Bert 모델이 가지는 최대 입력 토큰 길이의 한계 - news를 전부 입력 받지 못함.이를 위해 최대 입력 토큰 길이가 큰 모델에 대해서 학습함 (Longformer, BigBird)

- Longformer, BigBird 모델은 입력 토큰 길이 제한 문제를 해결하기 위해 가까운 토큰끼리만 정보를 공유하도록 하는 Sliding window 기법을 채택 (CNN의 filter_size와 stride 개념과 유사)

- 하지만 Sliding window는 주변 토큰간의 관계만 포함하여 가장자리 Context의 정보를 반영하지 못함 -> 뉴스가 두괄식이기 때문에 더 critical한 영향을 주게 됨.

Appendix

Fake Detect Model Architecture
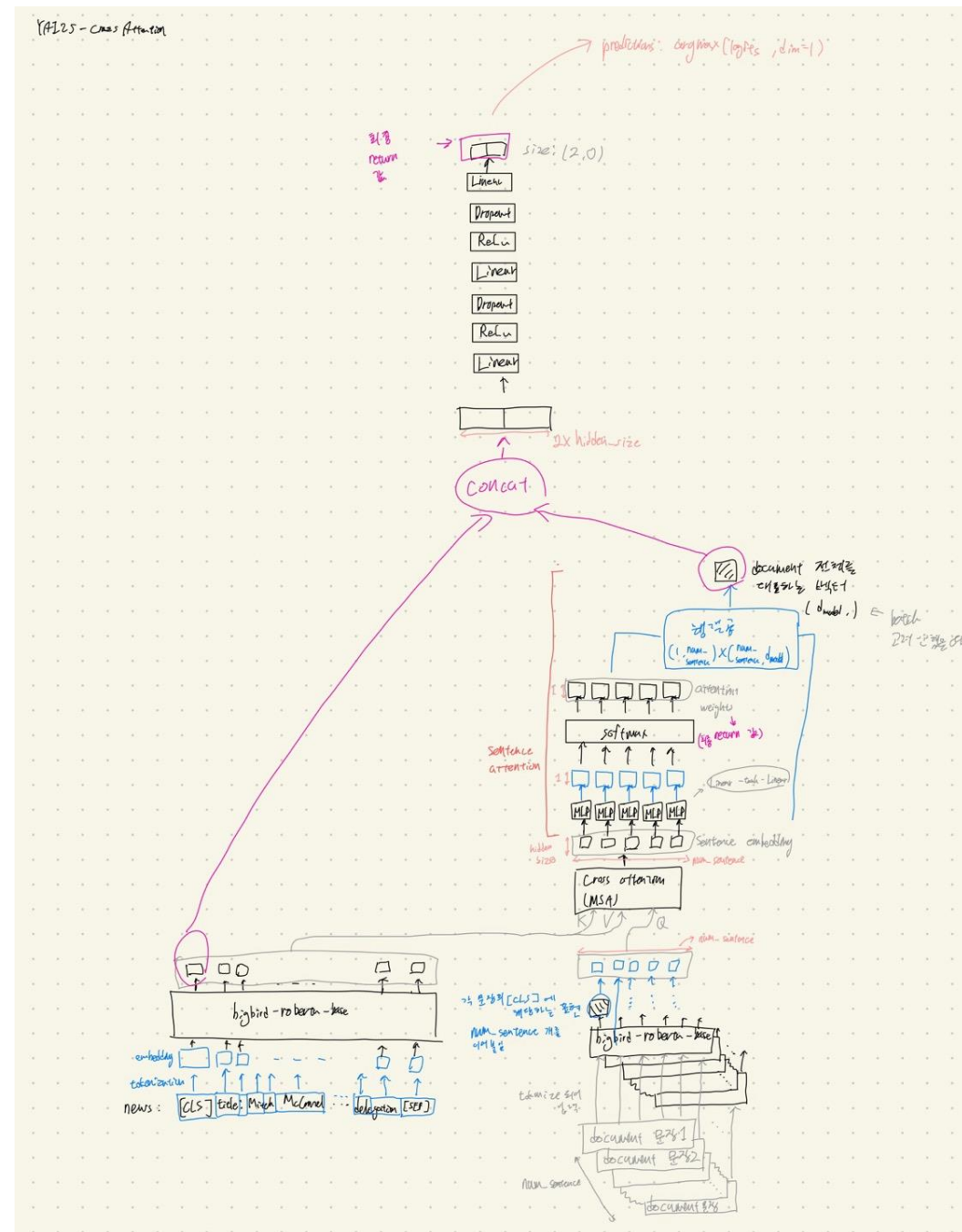
Appendix
Fake Detect Model Cross-attention

```
Cross Attention Output: tensor([[[ 0.0844, -0.3101,  0.0440,  ..., -0.3433, -0.1407, -0.1447],
        [ 0.0896, -0.3066,  0.0452,  ..., -0.3472, -0.1426, -0.1489],
        [ 0.0856, -0.3137,  0.0411,  ..., -0.3450, -0.1451, -0.1469],
        ...,
        [ 0.0839, -0.3093,  0.0439,  ..., -0.3430, -0.1400, -0.1449],
        [ 0.0834, -0.3088,  0.0442,  ..., -0.3457, -0.1387, -0.1470],
        [ 0.0829, -0.3091,  0.0441,  ..., -0.3428, -0.1450, -0.1490]]],
       device='cuda:0', dtype=torch.float16, grad_fn=<SqueezeBackward1>)
```

```
Raw Attention Scores: tensor([[[118.2500, 121.6250, 189.1250,  ..., 100.0625, 117.6875, 308.0000],
        [118.0000, 121.3125, 191.0000,  ..., 100.4375, 118.0000, 307.5000],
        [114.1875, 117.1250, 182.6250,  ...,  96.5000, 113.5625, 297.2500],
        ...,
        [134.3750, 139.1250, 218.6250,  ..., 114.8125, 133.5000, 349.0000],
        [114.0625, 118.1250, 183.3750,  ...,  97.1875, 114.1875, 297.0000],
        [113.0625, 116.9375, 183.3750,  ...,  97.1875, 114.1250, 294.7500]]],
       device='cuda:0', dtype=torch.float16, grad_fn=<UnsafeViewBackward0>)
Attention Weights: tensor([[[0.0020, 0.0020, 0.0029,  ..., 0.0018, 0.0020, 0.0052],
        [0.0020, 0.0020, 0.0029,  ..., 0.0018, 0.0020, 0.0052],
        [0.0020, 0.0020, 0.0028,  ..., 0.0018, 0.0020, 0.0051],
        ...,
        [0.0020, 0.0020, 0.0030,  ..., 0.0018, 0.0020, 0.0059],
        [0.0020, 0.0020, 0.0028,  ..., 0.0018, 0.0020, 0.0050],
        [0.0020, 0.0020, 0.0028,  ..., 0.0018, 0.0020, 0.0050]]],
       device='cuda:0', grad_fn=<SoftmaxBackward0>)
Final Attention Output: tensor([[[-0.0271, -0.0010,  0.1580,  ..., -0.3037, -0.0191,  0.0058],
        [-0.0270, -0.0010,  0.1581,  ..., -0.3037, -0.0190,  0.0056],
        [-0.0270, -0.0009,  0.1577,  ..., -0.3035, -0.0193,  0.0060],
        ...,
        [-0.0273, -0.0018,  0.1595,  ..., -0.3037, -0.0183,  0.0047],
        [-0.0271, -0.0009,  0.1577,  ..., -0.3032, -0.0191,  0.0059],
        [-0.0271, -0.0010,  0.1576,  ..., -0.3035, -0.0190,  0.0058]]],
```

- Cross Attention output이 매우 비슷한 벡터들이 나옴.
- Evidence로 나온 문장들이 유사하여 cross-attention의 값이 비슷하게 나오는 것으로 추정됨

- Query, Key의 분포가 비슷해서 내적값도 유사한 분포를 갖게 되는 것으로 추정.
- 따라서 attention_weights가 매우 작은 숫자로 나옴 + 비슷한 분포로 나옴

Appendix
Fake Detect Model Gradient
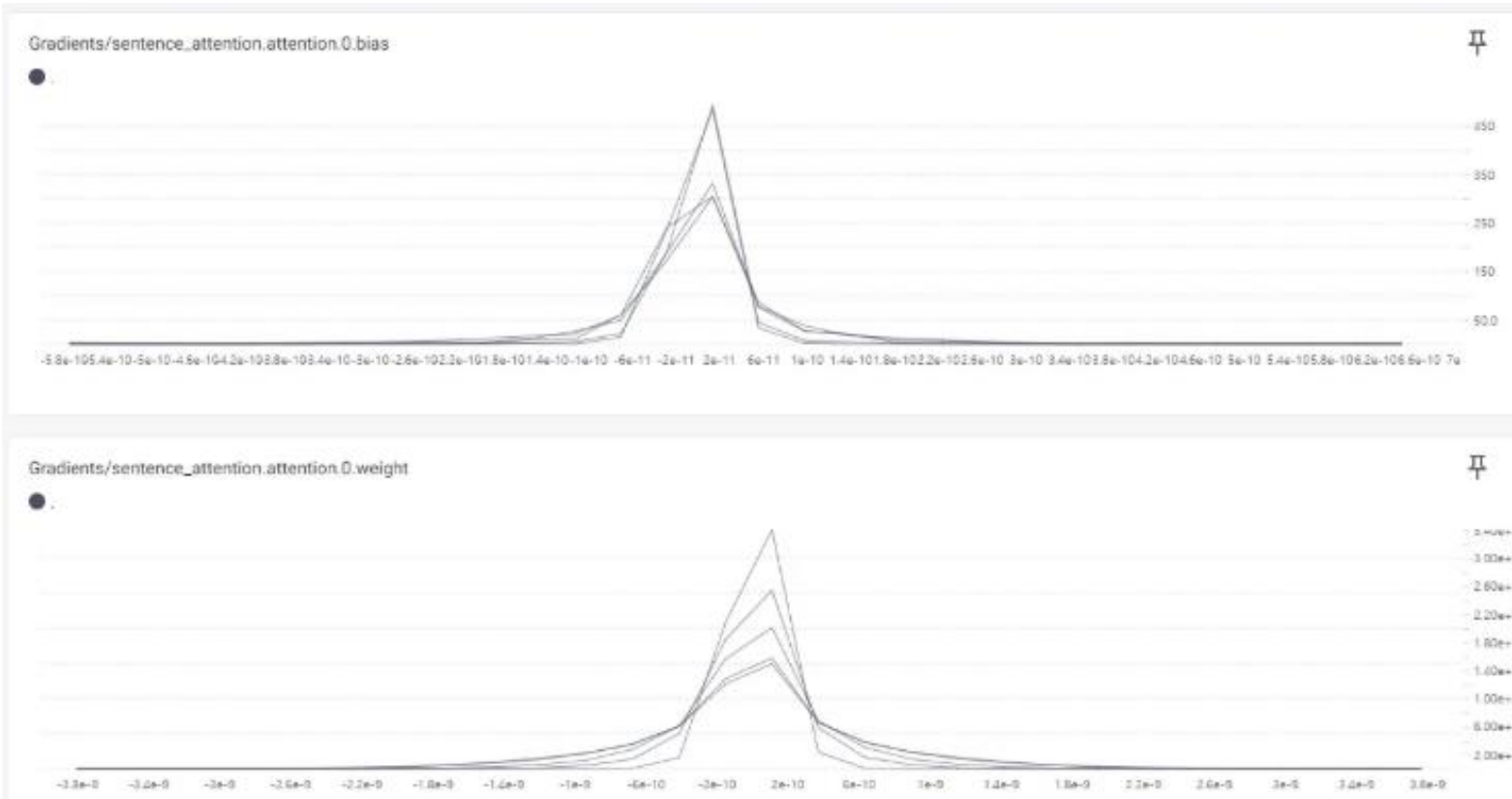
• Cross attention의 출력이 유사하여 Sentence attention 레이어가 학습이 되지 않는 것을 관찰 가능.

Appendix

Fake Detect Model Execution Time

• 1 batch(batch size = 1) 당 평균 7초정도의 실행 시간이 걸림

```
Execution time: 1 min 18.39 sec
Average execution time over 11 runs: 0 min 7.13 sec
Execution time: 1 min 25.51 sec
Average execution time over 12 runs: 0 min 7.13 sec
Execution time: 1 min 32.43 sec
Average execution time over 13 runs: 0 min 7.11 sec
Execution time: 1 min 39.51 sec
Average execution time over 14 runs: 0 min 7.11 sec
Execution time: 1 min 46.43 sec
Average execution time over 15 runs: 0 min 7.10 sec
Execution time: 1 min 53.62 sec
Average execution time over 16 runs: 0 min 7.10 sec
Execution time: 2 min 0.49 sec
Average execution time over 17 runs: 0 min 7.09 sec
Execution time: 2 min 7.32 sec
Average execution time over 18 runs: 0 min 7.07 sec
Execution time: 2 min 14.49 sec
Average execution time over 19 runs: 0 min 7.08 sec
Execution time: 2 min 21.38 sec
Average execution time over 20 runs: 0 min 7.07 sec
Step 20/208
Execution time: 2 min 28.43 sec
Average execution time over 21 runs: 0 min 7.07 sec
Execution time: 2 min 35.40 sec
Average execution time over 22 runs: 0 min 7.06 sec
Execution time: 2 min 42.31 sec
Average execution time over 23 runs: 0 min 7.06 sec
Execution time: 2 min 49.16 sec
Average execution time over 24 runs: 0 min 7.05 sec
```

단일 BERT 모델

단일 RoBERTa만 사용하였을 경우, 오른쪽의 사진과 같이 뉴스 본문만 이용하여 판별하고, 추가적으로 주어진 evidence 문장을 활용하는것은 찾기 어려웠다.