

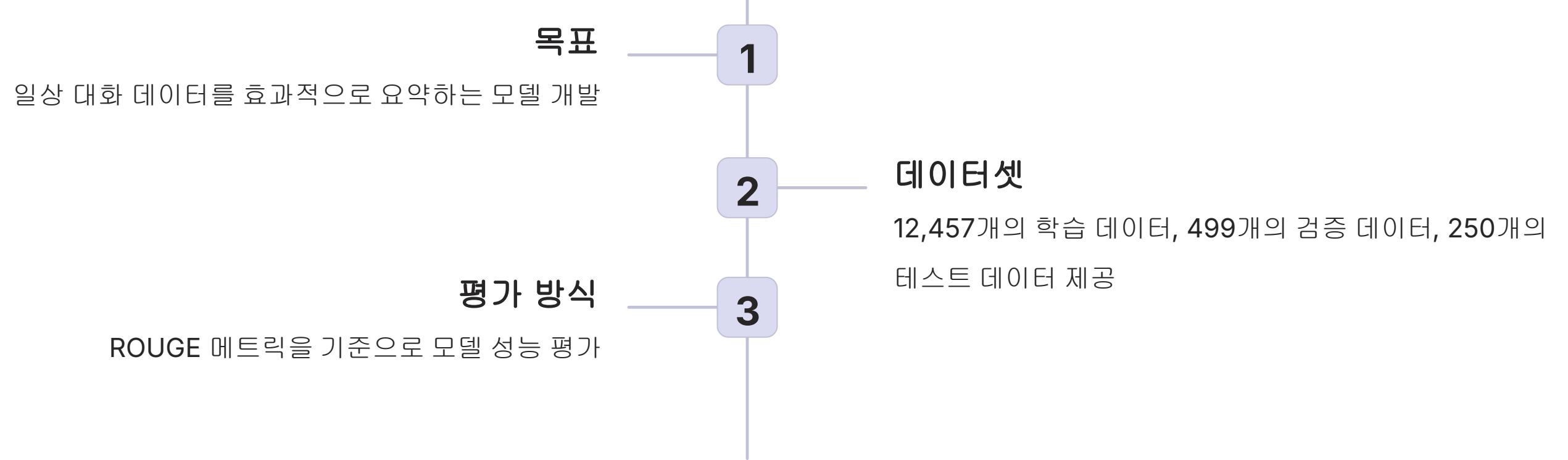
# 디지털 보물찾기

## 팀의 대화 요약

디지털 보물찾기 팀이 일상 대화 요약 경연에 도전했습니다. 박석,  
**도전기** 백경탁, 한아름, 위효연으로 구성된 팀은 다양한 실험을 통해 성능  
향상을 꾀했습니다.

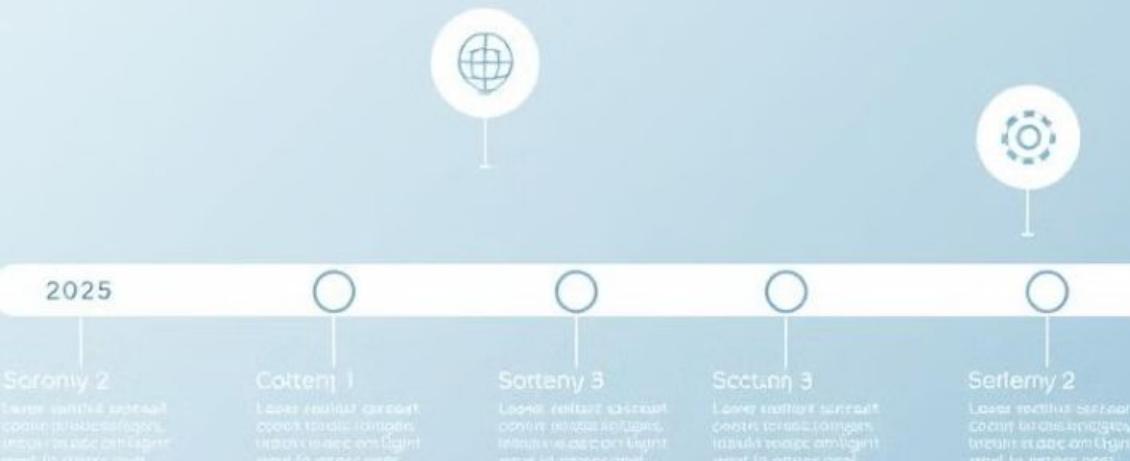
Team #5 :: 디지털 보물찾기 (Digital Treasure Quest)  
박석, 백경탁, 한아름, 위효연

# 대회 개요



# Th scaph

- Syndresit • colectesport use • Project
- Scvether dia suollering ograms • Contraction



# 대회 일정



## 시작

2024년 8월 29일 프로젝트 시작 및 데이터셋 준비

## 중간

9월 2일 팀 병합 마감, 모델 개발 및 테스트 진행

## 종료

9월 10일 최종 모델 제출 및 프로젝트 종료

# 팀의 장단점

## 장점

- 다양한 경력과 경험을 가진 팀원들
- AI Assistant에 대한 높은 수용력

## 단점

- Git을 활용한 팀 단위 R&D 경험 부족
- Python 기반 R&D 경험 수준 낮음

# 팀의 전략적 접근

## 1 베이스라인 코드 구축

커맨드라인 인터페이스로 동작하는 팀 베이스라인 코드 구축

## 2 AI 활용

ChatGPT 등 AI 도구를 활용하여 EDA와 모델 선택 방향성

설정

## 3 개인별 모델링

팀원별 수준에 맞는 서로 다른 머신러닝 모델링 진행



# 팀 문화와 정신

## 학습 중심

개인별 학습을 통해 머신러닝 R&D에 필요한 지식과 경험 획득

## 상호 존중

팀원 각자의 상황을 이해하고 인정하며 존중

## AI 활용

AI Assistant를 적극 활용하여 개인별 생산성 극대화

## 균형 잡힌 접근

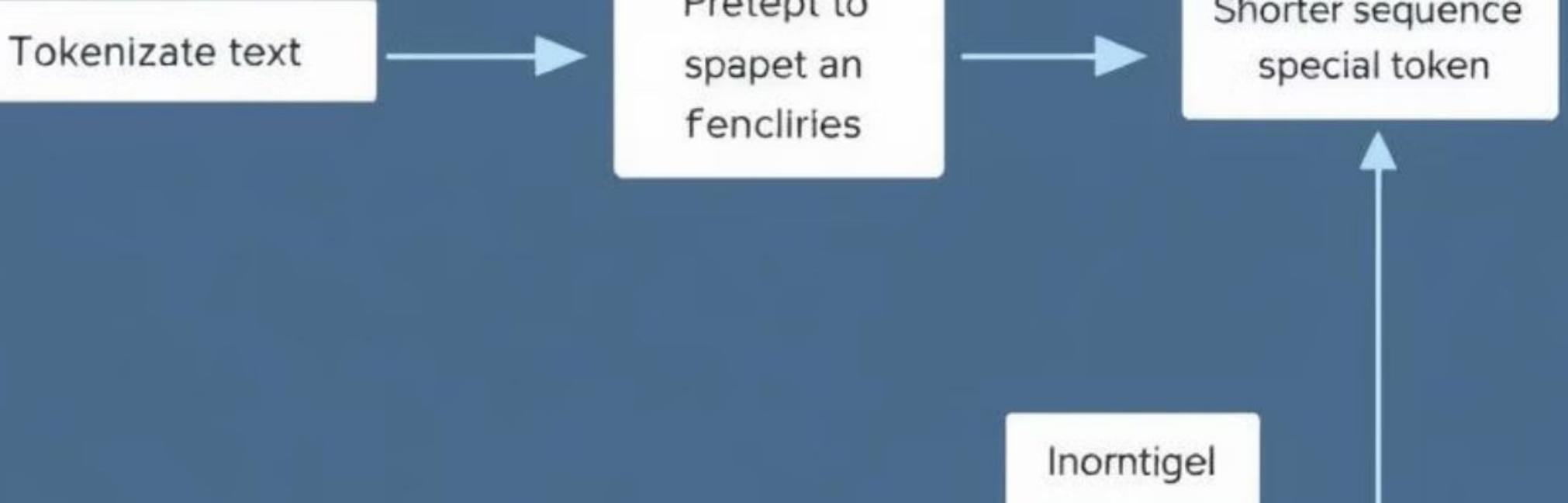
팀 전체 목표를 위해 개인의 스케줄이나 리소스를 희생하지 않음





# 데이터셋 개요

데이터 유형	행 수	열 수
학습 데이터	12,457	3
검증 데이터	499	3
테스트 데이터	250	3



# 데이터 전처리

## 1 토큰화

AutoTokenizer를 사용하여  
BART 모델에 맞게 토큰화 작업  
수행

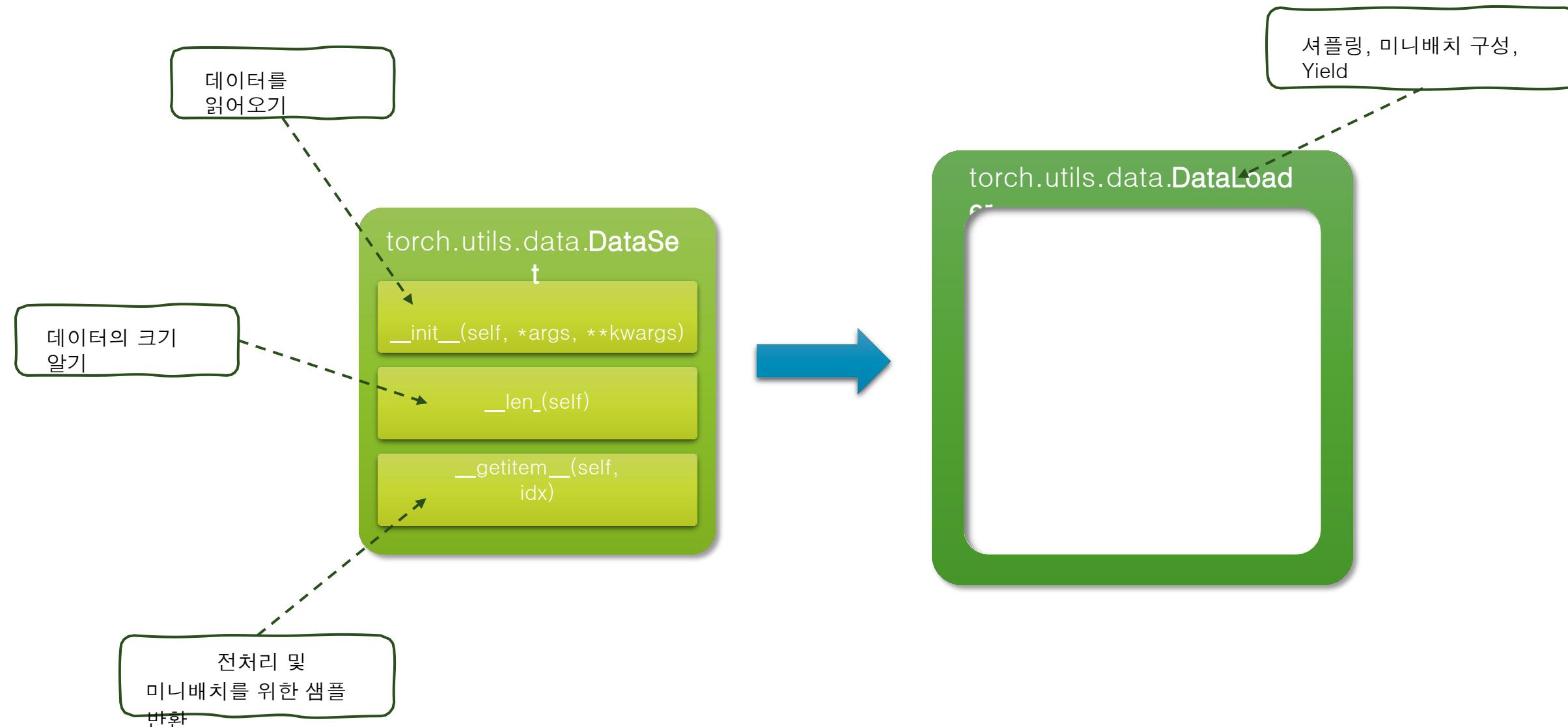
## 2 패딩 및 트렁케이션

문장을 모델 입력에 맞게  
적절한 길이로 조정

## 3 특수 토큰 처리

시작 토큰과 종료 토큰 등 특수  
토큰 추가

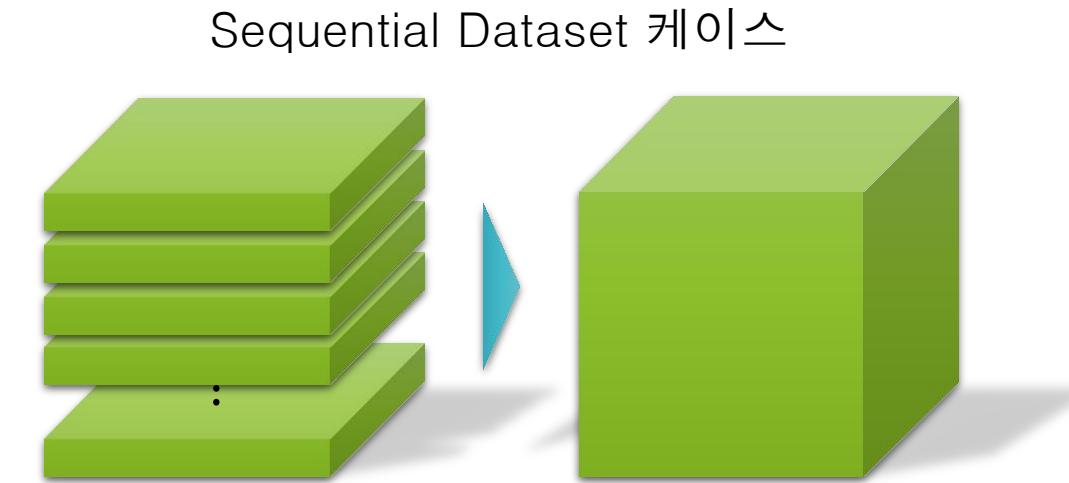
# Dataset & Dataloader 컨셉



위 이미지는 Dataset과 DataLoader의 관계 및 역할을 시각적으로 보여줍니다.

Dataset은 데이터 접근과 전처리를, DataLoader는 배치 구성과 데이터 공급을 담당하는 것을 확인할 수 있습니다.

# Dataset & Dataloader 세부사항



## DataLoader의 미니배치 구성 과정

DataLoader가 미니배치를 구성할 때:

- Dataset의 `_getitem_()` 함수를 통해 미니배치의 각 element들의 tensor를 넘겨받음
- 텐서들을 concat 또는 stack 하여 미니배치 tensor를 만들어냄

## 데이터셋 유형별

**특징**  
Tabular Dataset 케이스:

- 구조화된 데이터를 다룸
- 행과 열로 구성된 테이블 형태

Sequential Dataset 케이스:

- 시계열 또는 순차적 데이터를 다룸
- 데이터 간의 순서가 중요

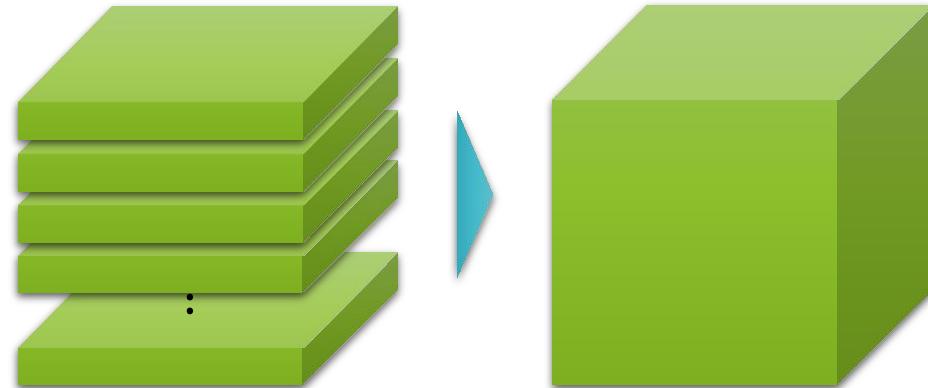
# 가변 길이 텐서의

## 문제점

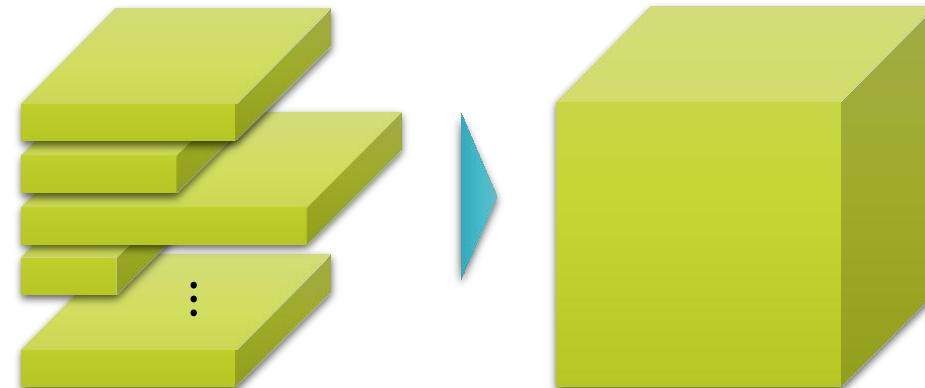
NLP와 같이 미니배치 내의 각 element들의 tensor의 크기가 다를 땐?

문장들의 길이가 다를 것이므로 tensor의 크기가 다를 수 밖에 없다.

Equal Length



Inequal Length



## Equal Length vs Inequal Length

- Equal Length: 문장의 길이를 코퍼스 전체에 대해서 고정하여,  
Inequal Length: 미니배치의 크기를 고정하여 해결! NO

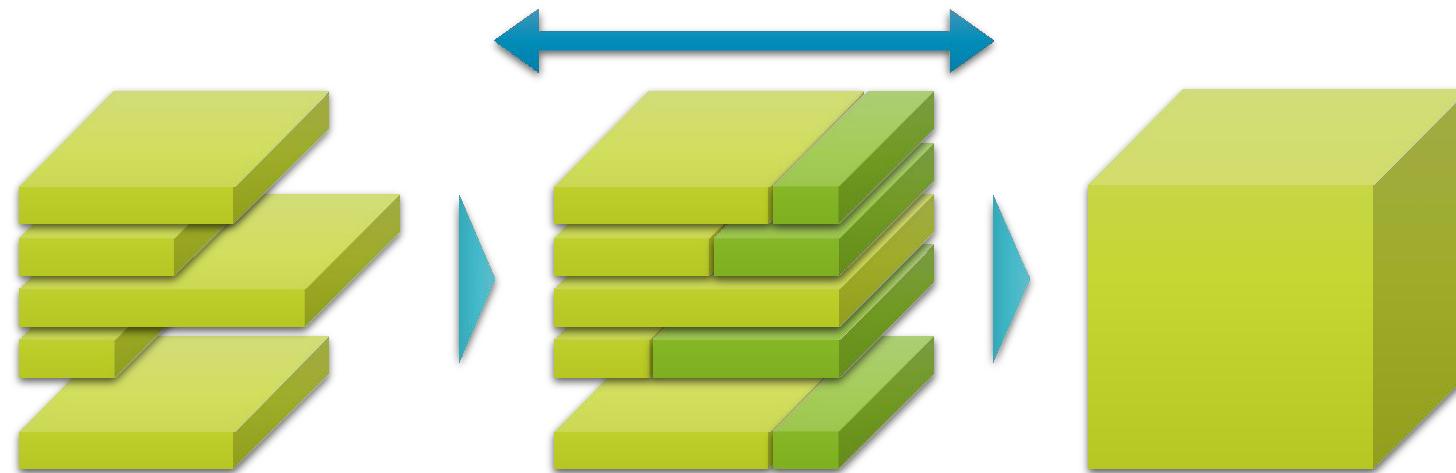
# 해결 방법: Collate Function

Dataloader의 파라미터 **collate\_fn**을 통해 해결 가능합니다.

**collate\_fn** (*callable, optional*) – merges a list of samples to form a mini-batch of Tensor(s).

Used when using batched loading from a map-style dataset.

<https://pytorch.org/docs/stable/data.html?highlight=dataloader#torch.utils.data.DataLoader>



## Collate Function

- Dataloader는 **collate\_fn**에 미니배치 element들을 list로 만들어 넘겨줍니다.
- List를 받아 가장 긴 문장 기준으로 padding 작업을 수행 후, 미니배치 텐서로 만들어 반환합니다.

# Review: Subword Segmentation

언어	단어	조합
영어	Concentrate	con(=together) + centr(=center) + ate(=make)
한국어	집중(集中)	集(모을 집) + 中(가운데 중)

- 많은 언어들에서, 단어는 더 작은 의미 단위들이 모여 구성됨
- 따라서 이러한 작은 의미 단위로 분절할 수 있다면 좋을 것
- 하지만 이를 위해선 언어별 subword 사전이 존재해야 할 것
- Data-driven 통계 방식으로 해결

# 서브워드 분절 과정

서브워드 분절 과정은 다음과 같은 단계로 이루어집니다:

1. 단어 사전 생성 (빈도 포함)
2. Character 단위로 분절 후, pair 별 빈도 카운트
3. 최빈도 pair를 골라, merge 수행
4. Pair 별 빈도 카운트 업데이트
5. 3번 과정 반복

이 과정은 크게 두 단계로 나눌 수 있습니다:

## 학습 단계

### (Training)

학습 단계에서는 위의 1-5 과정을 반복하여 서브워드 단위를 학습합니다.

## 적용 단계

### (Applying)

적용 단계에서는 다음과 같은 과정을 거칩니다:

1. 각 단어를 character 단위로 분절
2. 단어 내에서 '학습 과정에서 merge에 활용된 pair의 순서대로' merge 수행

# BPE(Byte Pair Encoding)

BPE(Byte Pair Encoding)는 자연어 처리(NLP)에서 사용되는 서브워드 분할 기법입니다. 특히, 신경망 기반의 언어 모델이나 번역 모델에서 단어를 효율적으로 표현하고 처리하기 위해 사용됩니다. BPE는 자주 등장하는 문자열 쌍을 병합하여 점진적으로 더 큰 서브워드를 만들어내는 방식으로 작동합니다.

## BPE의 주요 개념과 과정

- 초기 설정:** 모든 단어를 문자 단위로 분할합니다. 예를 들어, 단어 **hello**는 ['h', 'e', 'l', 'l', 'o']로 분할됩니다.
- 빈도 계산:** 말뭉치에서 각 문자 쌍의 빈도를 계산합니다. 예를 들어, 'e'와 'l'의 빈도가 높다면 이 둘이 하나의 서브워드로 병합될 가능성이 높습니다.
- 반복적인 병합:** 가장 자주 등장하는 문자 쌍을 병합하여 새로운 서브워드를 만듭니다. 이 과정을 반복하여, 목표로 하는 서브워드 개수에 도달할 때까지 병합을 계속합니다. 예를 들어, 'h', 'e', 'l', 'l', 'o'에서 'l'과 'l'이 자주 함께 나타난다면, 이를 병합하여 'll'로 만듭니다.
- 서브워드 사전 생성:** 최종적으로 병합된 서브워드들이 사전을 형성하게 되며, 이 사전은 모델이 텍스트를 처리할 때 사용됩니다.

## BPE의 장점

- 희귀 단어 처리:** 희귀한 단어도 여러 서브워드로 나뉘어 표현되므로, 모델이 이를 처리할 수 있습니다. 이는 OOV(Out-Of-Vocabulary) 문제를 효과적으로 해결합니다.
- 어휘 크기 축소:** 기존의 단어 단위 어휘보다 훨씬 작은 크기의 어휘를 사용할 수 있어, 메모리 사용량을 줄이고 학습 속도를 개선할 수 있습니다.
- 언어 독립성:** BPE는 언어에 독립적인 기법이므로 다양한 언어에서 효과적으로 사용할 수 있습니다.

## BPE의 적용 예시

BPE는 주로 Transformer 기반의 모델들, 특히 OpenAI의 GPT 및 BERT와 같은 모델들에서 사용됩니다. 이 모델들은 BPE로 텍스트를 서브워드 단위로 분할하여 입력으로 사용합니다.

# OoV(Out-of-Vocabulary)

OoV(Out-Of-Vocabulary)는 자연어 처리(NLP)에서 사용하는 용어로, 모델이 학습되지 않은 단어나 토큰을 의미합니다. 말 그대로, 모델이 학습하는 동안 어휘(vocabulary) 목록에 포함되지 않은 단어를 처리할 때 발생하는 문제를 나타냅니다.

## OoV 문제의 발생

NLP 모델은 훈련 중에 주어진 텍스트 데이터를 기반으로 어휘 목록을 만듭니다. 이 어휘 목록에는 훈련 데이터에서 가장 빈번하게 나타나는 단어들이 포함됩니다. 그러나, 훈련 데이터에 포함되지 않은 단어나, 드물게 나타나는 단어들이 테스트 또는 실무 데이터에서 나타날 수 있습니다. 이때 이러한 단어들이 "Out-of-Vocabulary" 단어라고 합니다.

예를 들어, 훈련 데이터에 "apple"이라는 단어가 포함되어 있지 않다면, 모델이 "apple"이라는 단어를 만났을 때 이 단어를 이해하거나 처리하기 어렵습니다.

## OoV 문제 해결 방법

1. 서브워드 기법 (Subword Techniques):
  - **BPE (Byte Pair Encoding)**: 단어를 더 작은 서브워드 단위로 분할하여 처리합니다. 예를 들어, "unbelievable"을 "un", "believ", "able"로 나누어 처리할 수 있습니다. 이렇게 하면 OoV 문제를 줄일 수 있습니다.
  - **WordPiece**: BERT에서 사용되는 방식으로, 유사하게 단어를 서브워드 단위로 분할합니다.
  - **SentencePiece**: Google의 텍스트 처리 라이브러리로, BPE와 유사한 방식으로 작동합니다.
2. 미리 정의된 OoV 토큰 사용:
  - 모델의 어휘에 "<UNK>"와 같은 특별한 OoV 토큰을 추가하여, 모델이 학습 중에 알 수 없는 단어를 이 토큰으로 대체합니다.
3. Character-level 모델:
  - 단어를 문자 단위로 처리하여 OoV 문제를 피할 수 있습니다. 문자 단위 모델은 단어를 구성하는 개별 문자를 보고 새로운 단어를 처리할 수 있습니다.
4. 동의어 확장:
  - OoV 단어를 유사하거나 동의어로 대체하여 모델이 알 수 있는 단어로 변환하는 방법입니다.

OoV 문제는 NLP 모델의 성능에 영향을 미칠 수 있기 때문에, 이를 해결하기 위한 다양한 방법들이 연구되고 있으며, 서브워드 기법이 특히 많이 사용되고

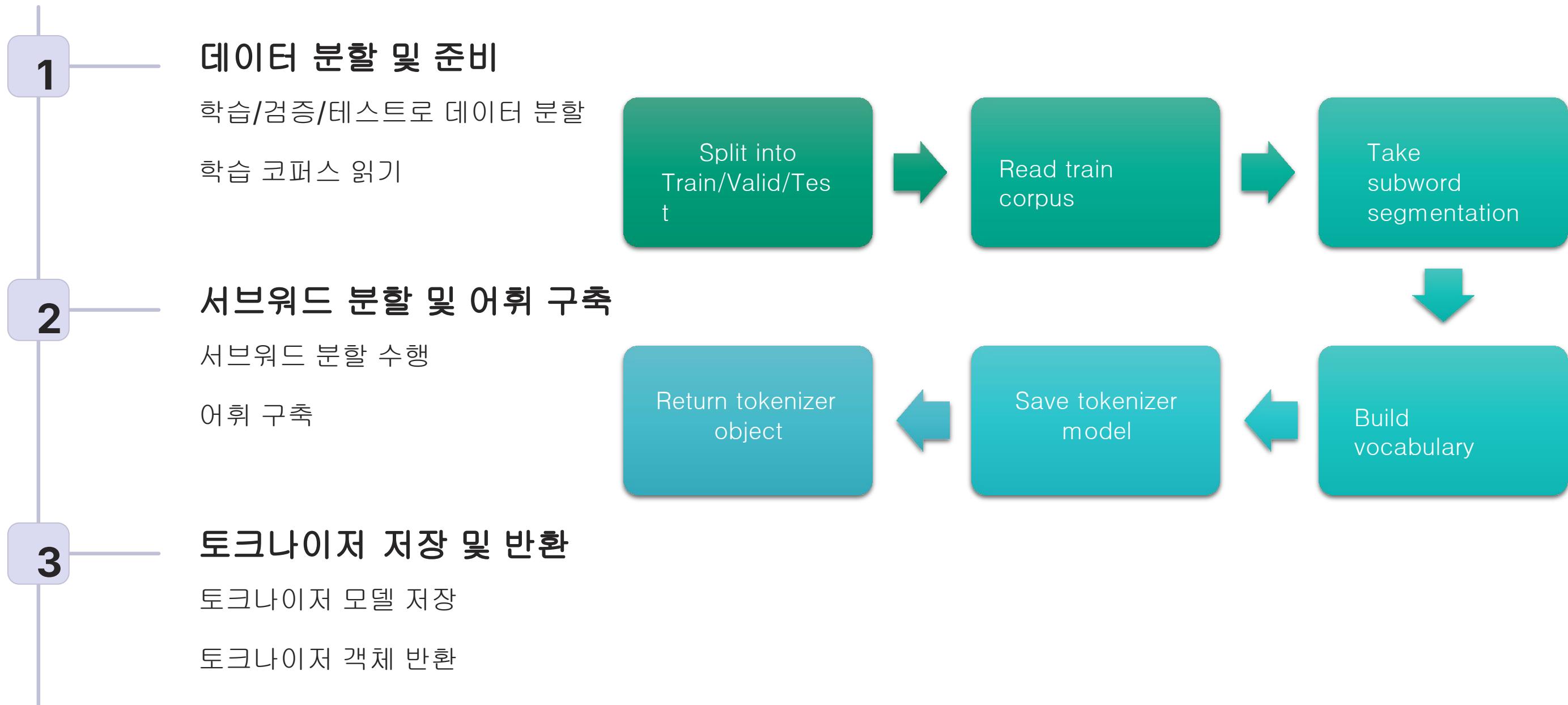
# 서브워드 분절의

01 **점** BPE 압축 알고리즘을 통해 통계적으로 더 작은 의미 단위(subword)로 분절 수행

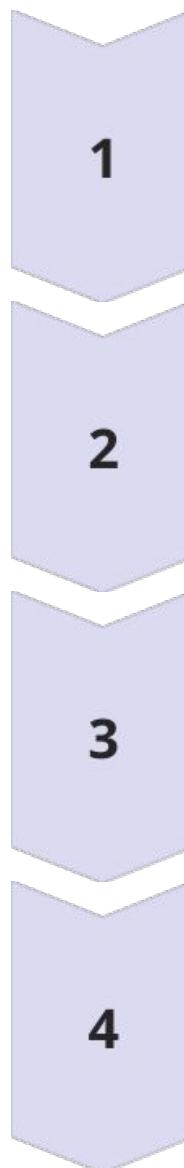
BPE를 통해 OoV를 없앨 수 있으며, 이는 성능상 매우 큰 이점으로 작용

- 하지만 data-driven 방식이므로 model 파일이 생김
- 학습 때 활용한 model 파일을 추론시에도 잘 적용해야함
- 형태소 분석기를 통한 분절과 맞물려 사용하거나 단독으로 사용하는 것이 추세

# 토크나이저 학습



# 배치화 절차



## 1 토크나이저 모델

로드된 토크나이저 모델을 불러옵니다.

## 2 문장 읽기

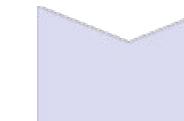
주어진 문장들을 읽습니다.

## 3 문장 토큰화

각 문장을 토큰화합니다.

## 4 문자열 토큰을 정수로 변환

문자열 토큰을 정수로 변환합니다.



## 5 짧은 문장에 패딩

작은 문장에 패드를 추가합니다.

## 6 미니배치 텐서로 결합

결합된 미니배치 텐서로 결합합니다.

## 7 미니배치 텐서 반환

미니배치 텐서를 반환합니다.

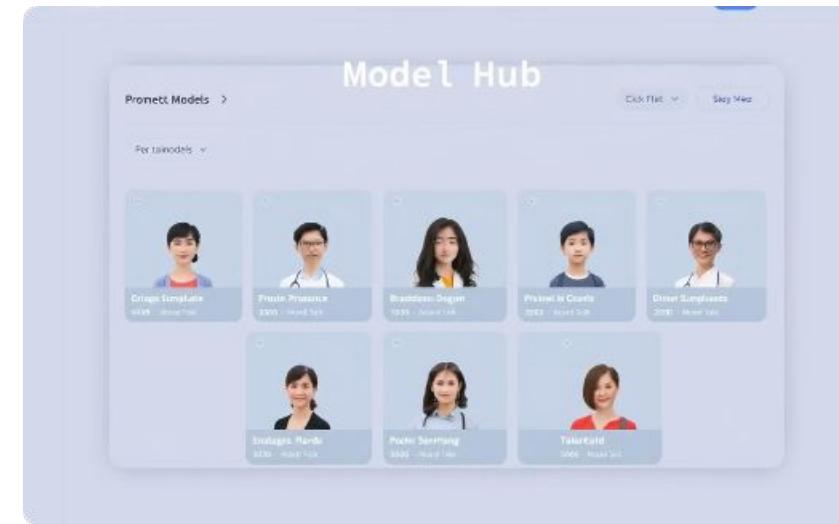
# Hugging Face Transformers: NLP를 위한 강력한

**도구**

Hugging Face Transformers는 자연어 이해(NLU)와 자연어 생성(NLG)을 위한 범용 아키텍처를 제공하는 강력한 라이브러리입니다. BERT, GPT, RoBERTa 등 다양한 모델을 지원하며, PyTorch와 TensorFlow 모두에서 사용할 수 있습니다. 이 라이브러리는 사용자 친화적인 인터페이스와 풍부한 기능을 제공하여 NLP 작업을 보다 쉽고 효율적으로 수행할 수 있게 해줍니다.



# Hugging Face Transformers



## Transformers의 특징

Transformers는 NLU와 NLG를 위한 general-purpose 아키텍처를 제공합니다. BERT, GPT, RoBERTa 등의 모델을 지원하며, PyTorch와 TensorFlow에서 모두 동작합니다.

<https://huggingface.co/transformers/>

## Model Hub

Model Hub를 제공하여 사용자들이 모델을 공유하고, 통일된 플랫폼 위에서 쉽게 application을 제작할 수 있도록 합니다. 사전학습된 모델 아이디 하나로 자동 다운로드 및 학습/추론 수행이 가능합니다.

<https://huggingface.co/models>

## 통합 파이프라인

모델 아키텍처 코드 뿐만 아니라, 전처리와 학습, 벤치마크 테스트 코드 모두 제공합니다. 전체 pipeline이 쉽게 통합될 수 있으며, 짧은 코드로 쉽게 재현 및 응용이 가능하도록 구성되어 있습니다.

# Hugging Face Tokenizer

Huggingface Tokenizer는 앞서 언급된 기능들을 제공하는 클래스입니다. 또한 각 아키텍처에 따라 필요한 방식을 각기 자식 클래스에 구현하여 제공합니다. `Transformers` 내에 구현됨으로써, 전처리와 학습 과정의 파이프라인이 쉽게 통합될 수 있습니다. 모델 연구자는 모델과 토크나이저를 사전학습하여 Model Hub에 업로드합니다. 모델 사용자는 이를 다운로드하여 downstream task에 fine-tuning하여 사용할 수 있습니다.

```
from transformers import BertTokenizer
tokenizer = BertTokenizer.from_pretrained('bert-base-uncased')
text_batch = ["I love Pixar.", "I don't care for Pixar."]
encoding = tokenizer(text_batch, return_tensors='pt', padding=True, truncation=True)
input_ids = encoding['input_ids']
attention_mask = encoding['attention_mask']
```

# PyTorch Ignite:

## 보일러플레이트

PyTorch Ignite를 사용하여 효율적인 딥러닝 모델 훈련을 위한

보일러플레이트 코드를 구축하는 방법을 살펴보겠습니다.

구축



# Boilerplate?

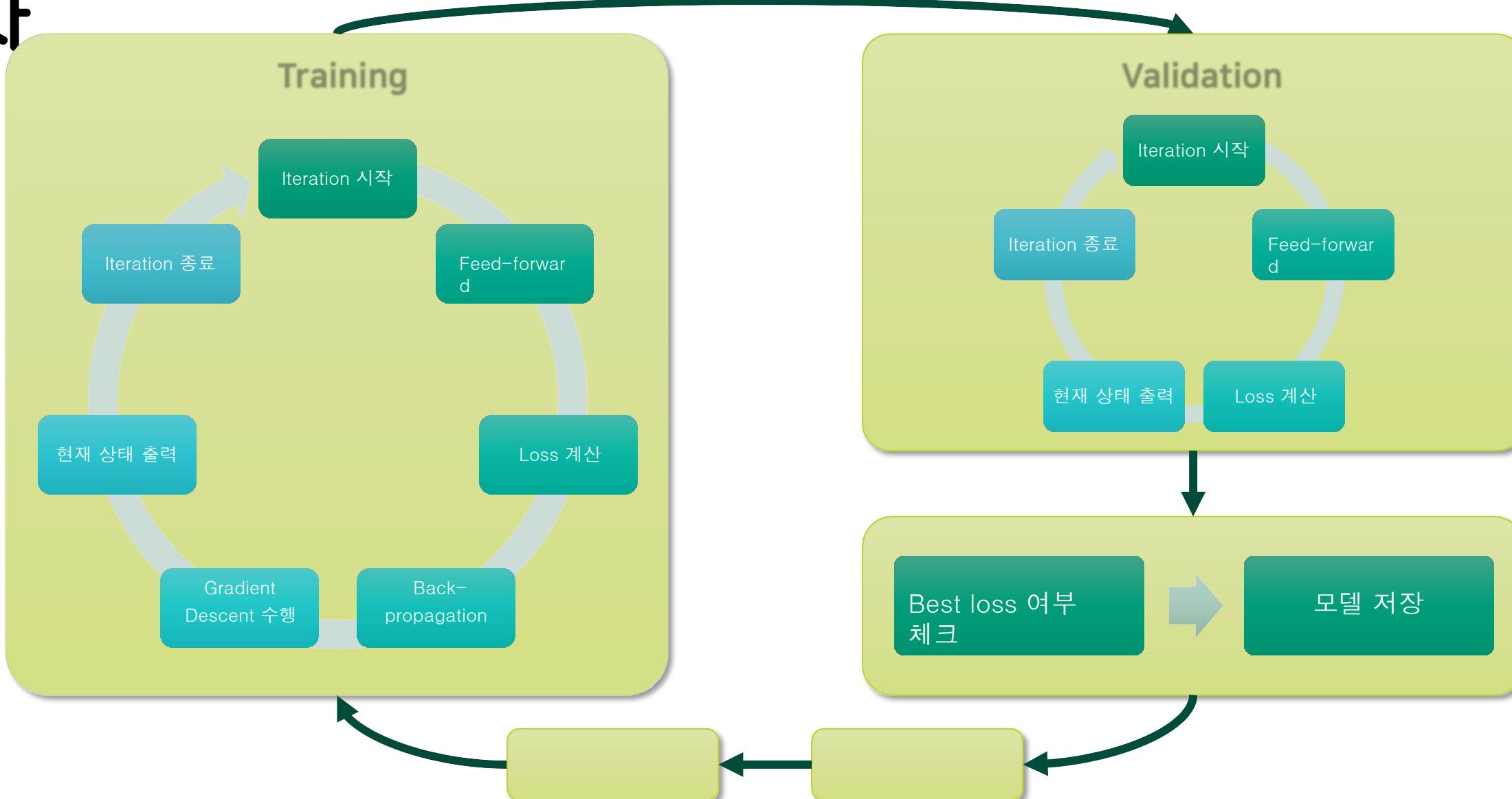
학습 **scheme**이 비슷할 경우, 모델과 **dataset**을 제외한 코드는 거의 동일합니다. 정작 중요한 모델을 코딩하는 시간보다 부수적인 요소 코딩에 더 많은 시간이 소요됩니다. 따라서 모델과 **dataset**만 갈아끼워 재사용 가능한 코드가 있으면 좋을 것입니다. 템플릿 (Template)을 만들어 보겠습니다.

```
8
1 Pubeltlate: aatalaye Shrappout
2 corellor: bustheypittbertlatder:
3 -cubely:-Pinplaght((Fe))
4 cullblion: Bqulett ine Template:
5 Deraction: enslyy/ithm sredleectally vr
6 abstcenter ()
7 wodeh (buttwor)
8 tubteler--motle prtetter:
9 stapeلتاتتو()
10 tudecture drabthts;
11 ) -sasian
12 (utecapestapuler/temptetr)
```

Model out  
Indlaled:  
leascy Uotight inadetion

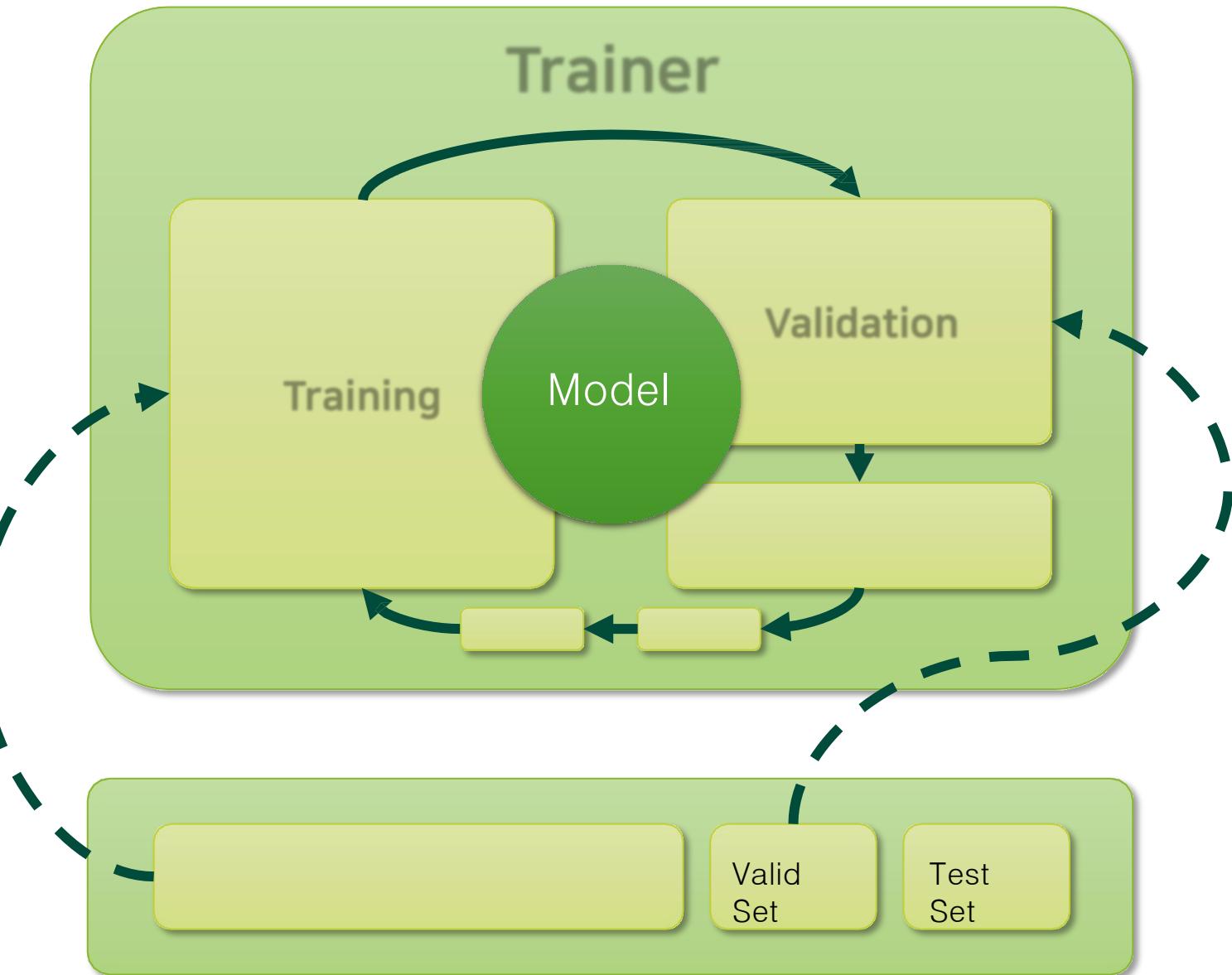
# 전형적인 모델 학습

## 절차



# 트레이너와 다른 요소들 간의 상호작용

트레이닝, 검증, 트레이너, 검증 세트, 테스트 세트, 모델 간의 상호작용은 매우 중요합니다. 트레이너는 모델을 훈련시키는 동안 검증 세트를 사용하여 성능을 평가하고, 최종적으로 테스트 세트로 모델의 성능을 측정합니다. 이 과정에서 트레이너는 모델과 지속적으로 상호작용하며, 훈련 데이터와 검증 데이터를 효과적으로 활용합니다.



# Huggingface Transformers 요약

## 일반적인 목적의 아키텍처

Transformers는 NLU와 NLG를 위한 general-purpose 아키텍처를 제공합니다. BERT, GPT, RoBERTa 등의 모델을 포함하여, PyTorch와 TensorFlow 2.0에서 모두 동작합니다. <https://huggingface.co/transformers/>

## Model Hub의 이점

Model Hub를 제공하여 사용자들이 모델을 공유하고, 통일된 플랫폼 위에서 쉽게 application을 제작할 수 있도록 합니다. 사전학습된 모델 아이디 하나로 자동 다운로드 및 학습/추론 수행이 가능합니다. <https://huggingface.co/models>

## 포괄적인 코드 제공

모델 아키텍처 코드 뿐만 아니라, 전처리와 학습, 벤치마크 테스트 코드 모두 제공합니다. 전체 pipeline이 쉽게 통합될 수 있으며, 짧은 코드로 쉽게 재현 및 응용이 가능하도록 구성되어 있습니다.

# 모델 선택



**KoBART**

한국어 요약에 특화된 BART 모델



**T5**

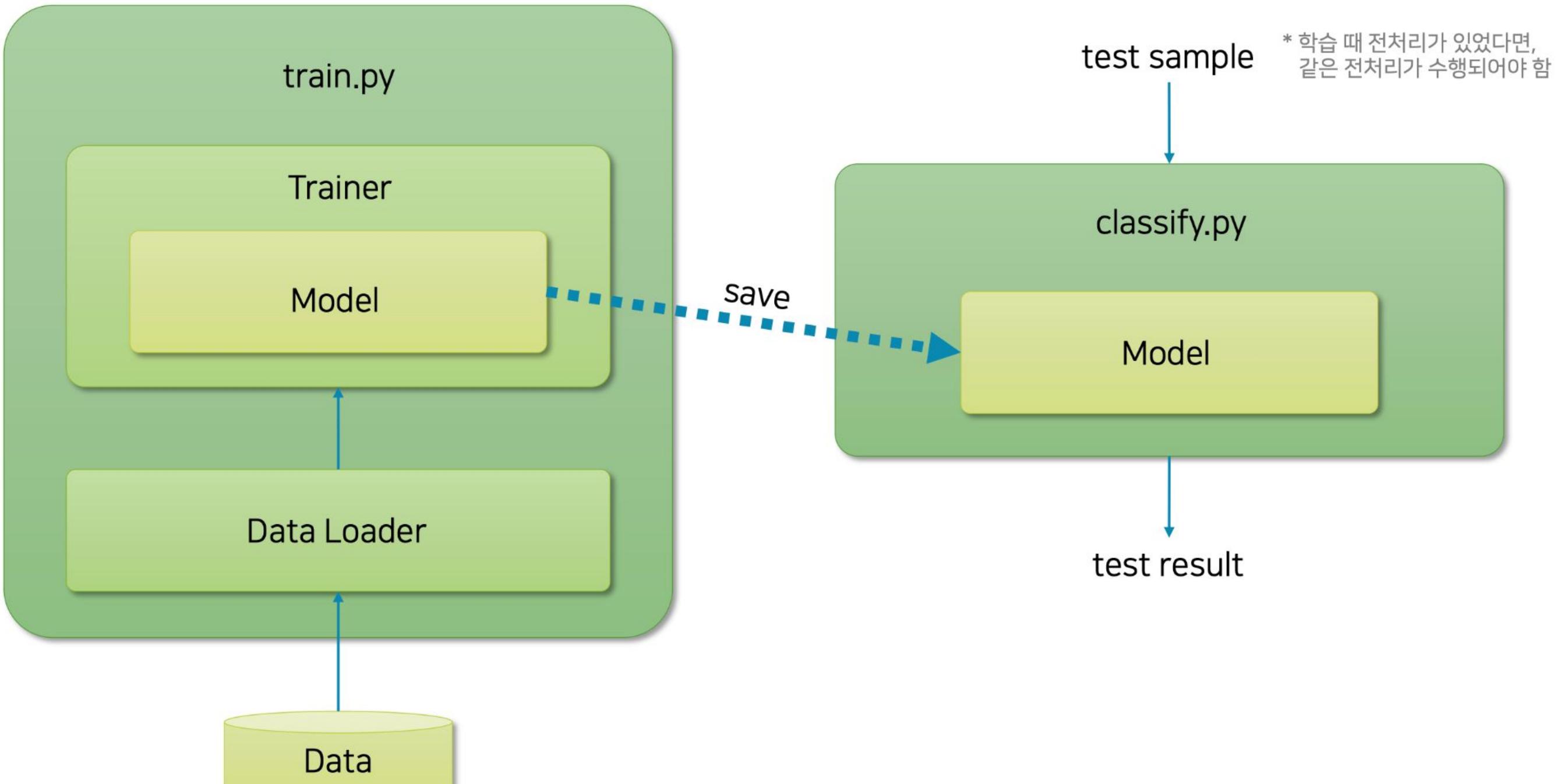
다목적 텍스트-투-텍스트 변환 모델



**mBART**

다국어 지원 BART 모델

# 딥러닝 학습 / 추론



## Conpl comterign:

Conpleading total  
navel a profetion → ad  
to cramerl luna



He opactland intcler  
contatogn neott pnanets



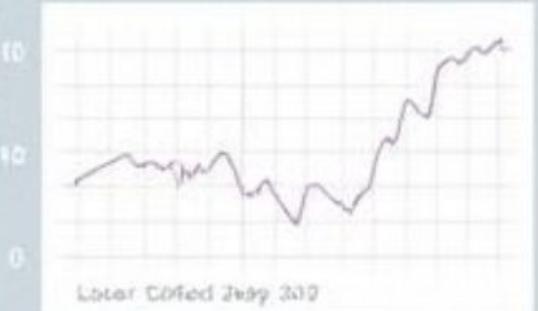
Parfu starchst

He erpente por limled  
contatogn lineng maants



Pypbhatte exomett

He erctnetapcrimaled  
contatgingrl dincipinents



Pypbinare exomett

# 하이퍼파라미터

## 인코더 최대 길이

512로 설정하여 충분한 문맥 반영

## 디코더 최대 길이

100로 설정하여 간결한 요약문 생성

## 배치 크기

50(Train),32(Val)로 설정하여 학습 속도와 메모리 사용의 균형  
유지

## 학습률

적절한 학습률 설정으로 모델 성능 최적화

# 성능 평가 지표

**ROUGE-1**

단어 단위 일치율 평가

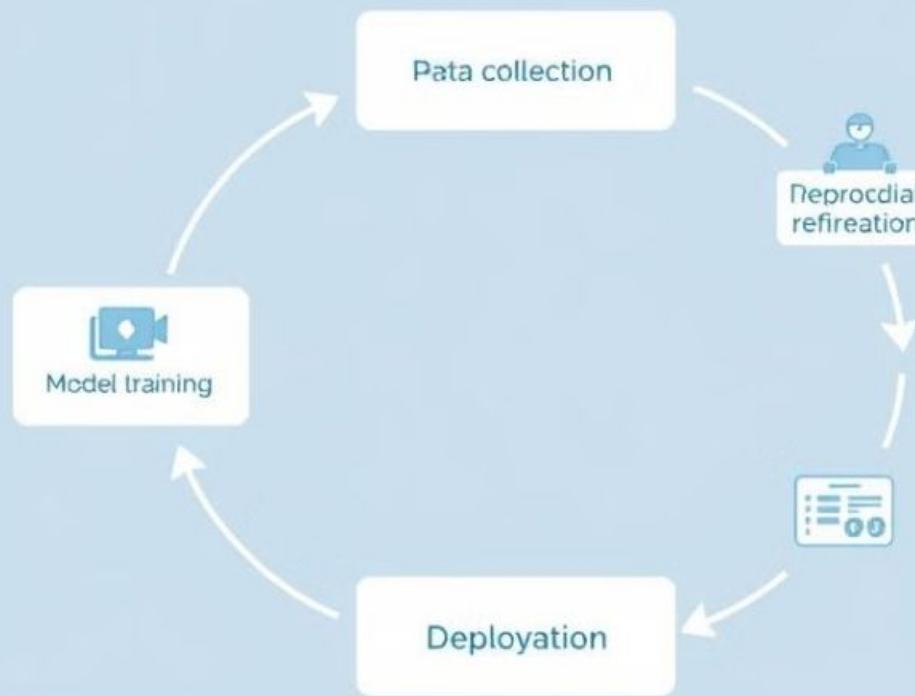
**ROUGE-2**

두 단어 연속 일치율 평가

**ROUGE-L**

최장 공통 부분열 기반 평가

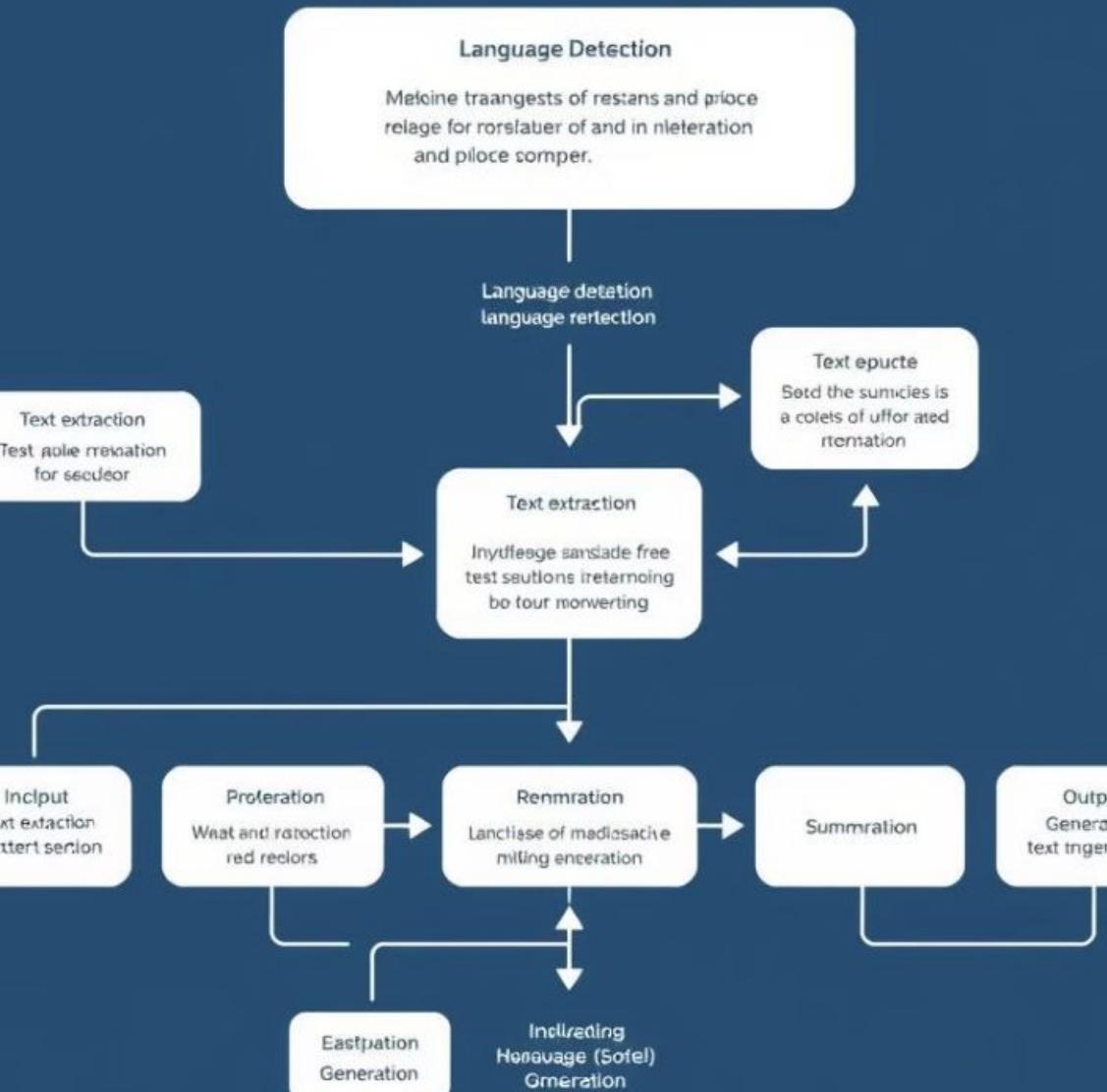
# Machine Learning



## 실험 과정

- 1 베이스라인 모델 실험  
초기 KoBART 모델로 베이스라인 성능 측정
- 2 모델 변형 및 최적화  
다양한 모델 구조와 하이퍼파라미터 조정 실험
- 3 양상을 기법 적용  
여러 모델의 결과를 조합하여 성능 향상 시도

# 번역-요약-번역



# 저그 블로그 한국어 → 영어 번역

DeepL API를 사용하여 한국어 대화를 영어로 번역

영어 요약

영어 요약에 특화된 모델을 사용하여 요약 생성

영어 → 한국어 번역

생성된 영어 요약을 다시 한국어로 번역

# Ensemble Learning

## 앙상블 기법

### 1 하드 보팅

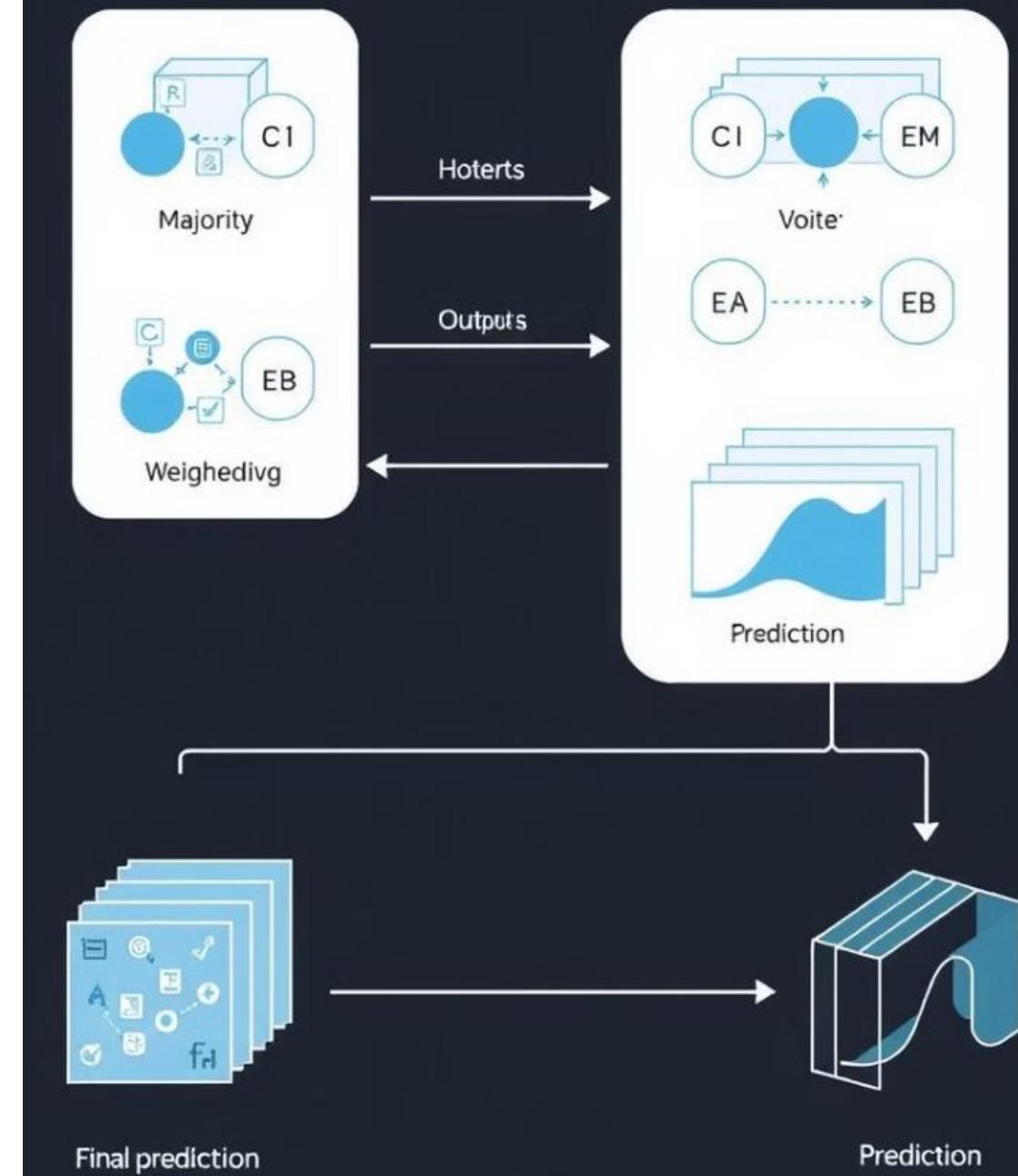
여러 모델의 결과 중 가장 많이 선택된 요약 선택

### 2 소프트 보팅

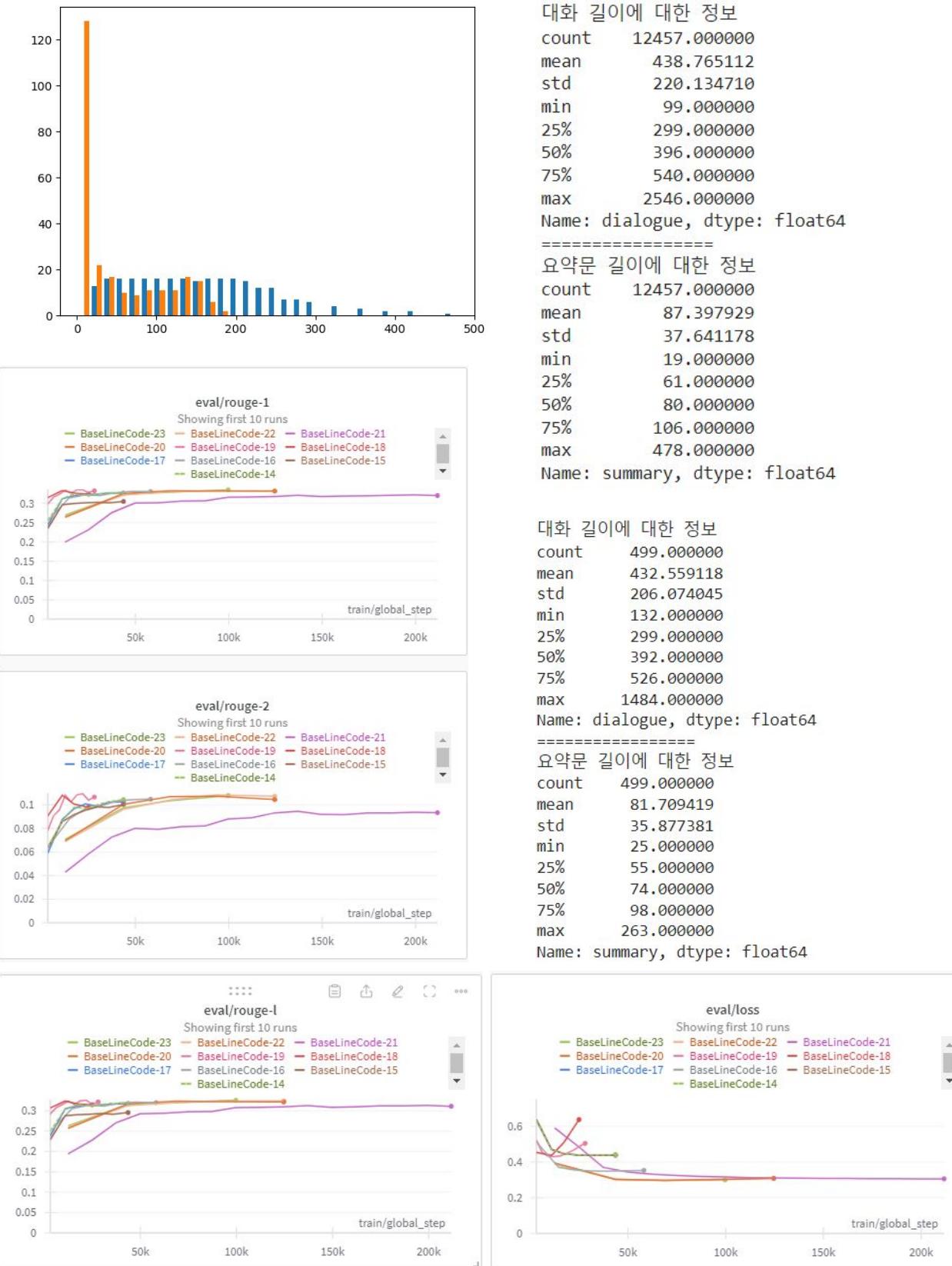
각 모델의 결과에 가중치를 부여하여 최종 요약 생성

### 3 스태킹

여러 모델의 결과를 입력으로 받는 메타 모델 사용



# 백경탁의 EDA



1

## Special Tokens 추가

pattern = r"#[a-zA-Z\d\s]\*#" 으로 데이터를 확인하여

다음과 같은 Special Tokens을 추가하였습니다.

'#DateOfBirth#', '#SSN#', '#CardNumber#', '#CarNumber#', '#Email#',  
'#Person#', '#Person4#', '#Person5#', '#Person6#', '#Person7#'

2

## 오타 정정

EDA를 통해 아래와 같은 오타를 찾아 정정하였습니다.

replace('ㅋㅋ', '웃기다')

replace('ㅇ로', '으로')

replace('제ㅏ', '제가')

replace('ㅍ알', '알')

replace('ㄷ거', '거')

replace('사람1#', '#Person1#')

replace('#Person 2#', '#Person2#')

replace('##Person1#', '#Person1#')

replace('#Person#', '#Person2#')

# Machine Learning

	Becw	Becw	Bcty	Becw	Lecw	Becw
Accuracy	•	•	•	•	•	•
precision	•	•	•	•	•	•
F1 score	•	•	•	•	•	•
accuracy	•	150	130	7	26	20
procecty	•	•	•	•	•	33
incaspment	•	14	•	8	50	53
pecision	•	•	•	•	•	20
pecilsion	•	50	•	9	•	30
incellatice	•	•	•	•	•	•
F1 score	14.5	191.4	172	1963	168	59
	0	1	0	2	8	3
	Becw	Becw	Bcty	Becw	Lecw	Becw

## 백경탁의 모델 선택 및 하이퍼파라미터 조정

- 1 모델 선택  
kobart-summarization(41 점대)과 dialogue-summarization-T5(42 점대), t5-large-korean-text-summary(43 점대) 세가지 모델을 비교했습니다. t5-large-korean-text-summary가 전반적으로 더 나은 성능을 보였습니다.
- 2 하이퍼파라미터 조정  
인코더 최대 길이(512, 660, 700, 850, 1024, 1536), 디코더 최대 길이(100, 125, 150, 200, 400), 배치 크기(1,2,3,4,32,50) 등을 조정했습니다. 인코더 최대 길이 1024, 디코더 최대 길이 200, 배치 크기 10이 최적의 성능을 보였습니다.
- 3 ROUGE 점수 평가  
각 실험에서 ROUGE-1, ROUGE-2, ROUGE-L 점수를 측정하여 요약 성능을 평가했습니다. t5-large-korean-text-summary 모델이 ROUGE 점수에서 더 나은 성능을 보였습니다.



## ROUGE 점수 평가 및 최종 결과

### 결과

1

#### ROUGE 점수 평가

5.3.4. ROUGE 점수 평가 시도: 각 실험에서 ROUGE-1, ROUGE-2, ROUGE-L 점수를 측정하여 요약 성능을 평가함. 결과: ROUGE-1 (단어 일치율)과 ROUGE-L (문장 구조 유사성) 점수가 높은 실험에서 더 나은 요약 결과를 얻음. 특히, t5-large-korean-text-summary 모델이 ROUGE 점수에서 더 일관된 성능을 보였음.

2

#### 최종 제출 및 결과

5.3.5. 최종 제출 및 결과 시도: 가장 좋은 성능을 낸 모델과 하이퍼파라미터 설정을 사용하여 최종 제출을 진행. 결과: t5-large-korean-text-summary 모델을 기반으로 한 실험에서 최종 제출 성능이 가장 높았으며, ROUGE-1, ROUGE-2, ROUGE-L 점수가 모두 우수하게 나타남.

3

#### 종합 결과

5.3.6. 종합 결과 t5-large-korean-text-summary 모델이 kobart-summarization 모델보다 요약 성능에서 더 나은 결과를 보임. 인코더와 디코더의 길이 조정, 배치 크기 등 하이퍼파라미터를 조정한 결과, 인코더 최대 길이 1024, 디코더 최대 길이 200, 배치 크기 10이 최적의 성능을 냈음. 최종 모델의 ROUGE 점수는 ROUGE-1, ROUGE-2, ROUGE-L 모두에서 높은 평가를 받았으며, 학습 및 검증 손실도 안정적으로 감소하여 모델이 잘 학습되었음을 확인함. 이러한 결과를 바탕으로 요약 성능을 최적화하기 위한 추가적인 하이퍼파라미터 튜닝과 데이터 확장이 다음 실험에서 도움이 될 것으로 예상됩니다.

4

#### 인사이트

5.3.7. 인사이트 실험에서 가장 성능 좋은 모델은? 실험에서 가장 성능이 좋은 모델은 t5-large-korean-text-summary 모델입니다. 이 모델은 ROUGE-1, ROUGE-2, ROUGE-L 점수에서 kobart-summarization 모델보다 더 일관되고 우수한 성능을 보였습니다. 또한, 최적의 하이퍼파라미터 설정(인코더 최대 길이 1024, 디코더 최대 길이 200, 배치 크기 1)을 사용했을 때, 이 모델이 가장 좋은 요약 성능을 냈습니다. 어떤 하이퍼파라미터 조정이 효과적이었나요? 실험에서 효과적인 하이퍼파라미터 조정은 다음과 같습니다: 인코더 최대 길이 (`encoder_max_length`) 1024: 인코더 최대 길이를 1024로 설정한 것이 성능 향상에 중요한 역할을 했습니다. 이 값이 너무 작으면 문장의 중요한 정보가 잘리게 되지만, 1024로 설정했을 때 충분한 문맥을 반영할 수 있었습니다. 디코더 최대 길이 (`decoder_max_length`) 200: 디코더 최대 길이를 200로 설정한 것이 요약 성능에 긍정적인 영향을 미쳤습니다. 요약 문장이 너무 길어지지 않으면서도 중요한 내용을 포함할 수 있는 적절한 길이였습니다. 배치 크기 (`batch_size`) 1: 배치 크기를 1으로 설정한 것이 학습 속도와 메모리 사용의 균형을 맞추는데 효과적이었습니다. 더 큰 배치 크기를 사용했을 때는 메모리 이슈가 발생할 수 있었습니다. 이러한 하이퍼파라미터 조정이 t5-large-korean-text-summary 모델의 성능을 최대화하는데 기여했습니다.

# 한아름의 EDA 및 접근

## 방법 EDA 결과

train.csv의 평균 대화 길이는 약 91 단어이며, 최소 20단어에서 최대 606단어입니다. 대부분의 대화에 2명이 참여하며, 최대 7명까지 참여합니다. dev.csv는 499개의 행과 4개의 열로 구성되어 있으며, 결측값은 없습니다.

## 접근 방법

다중 작업 학습, 다단계 시퀀스 작업, 시퀀스-투-시퀀스 모델, 복합 모델 접근법, 단일 네트워크에서 번역 및 요약 수행 등 다양한 방법을 고려했습니다.

## 방향 설정

단일 네트워크에서 번역 및 요약 수행, API 및 모델 활용, 베이스라인 코드 개선, AI Hub 멀티세션 대화 모델 사용 등을 시도했습니다.



# 한아름의 실험 결과 및 인사이트

1

실험 결과

2

주요 인사이트

3

데이터 전처리 개선

한국어 대화 데이터를 영어로 번역한 후, 영어 요약 모델을 적용한 뒤 다시 한국어로 번역하는 과정이 성능 향상에 중요한 역할을 했습니다.

4

아쉬운 점

유료 API나 Large 모델을 충분히 실험하지 못했으며, AI Hub 멀티세션 대화 모델 테스트를 진행하지 못했습니다. 또한 중간 코딩 오류로 인해 영어 요약 데이터를 기반으로 Rouge 점수를 산출한 점이 아쉬웠습니다.



# 위효연의 실험 과정

## 1 초기 실험

초기 baseline의 kobart 모델 파라미터를 바꿔가며 실행했으나 결과값이 좋지 않았습니다.

## 2 모델 변경

Icwt99/t5 모델을 적용하여 실행했으나, 한 번 실행할 때마다 20시간씩 걸려 여러 번 중단되었습니다.

## 3 API 시도

DeepL, ChatGPT Turbo 3.5, Claude 등의 API를 이용하여 원문을 다시 번역하려 했으나, 유료 사용 문제와 API 적용 시 코드 에러로 인해 어려움을 겪었습니다.

## 4 양상블 시도

기존 모델들의 결과값을 모아 양상블 단순 하드 보팅과 가중치 소프트 보팅을 진행했습니다.

# 위효연의 LLM 테스트 및 분석

## LLM 테스트

요약에 관한 논문을 리뷰하고 LLM을 이용한 텍스트 요약 평가를 진행했습니다.

## 번역 및 요약 비교

실제 생성요약모델 적용보다 ChatGPT, Claude 등의 LLM의 번역이 더 원활한 것을 확인했습니다.

## 최종 결과

T5모델과 KoBART를 섞은 양상을 보팅 결과값을 사용했으나, 실수로 2번 제출된 값이 가장 높은 점수를 받았습니다.

## 한계점

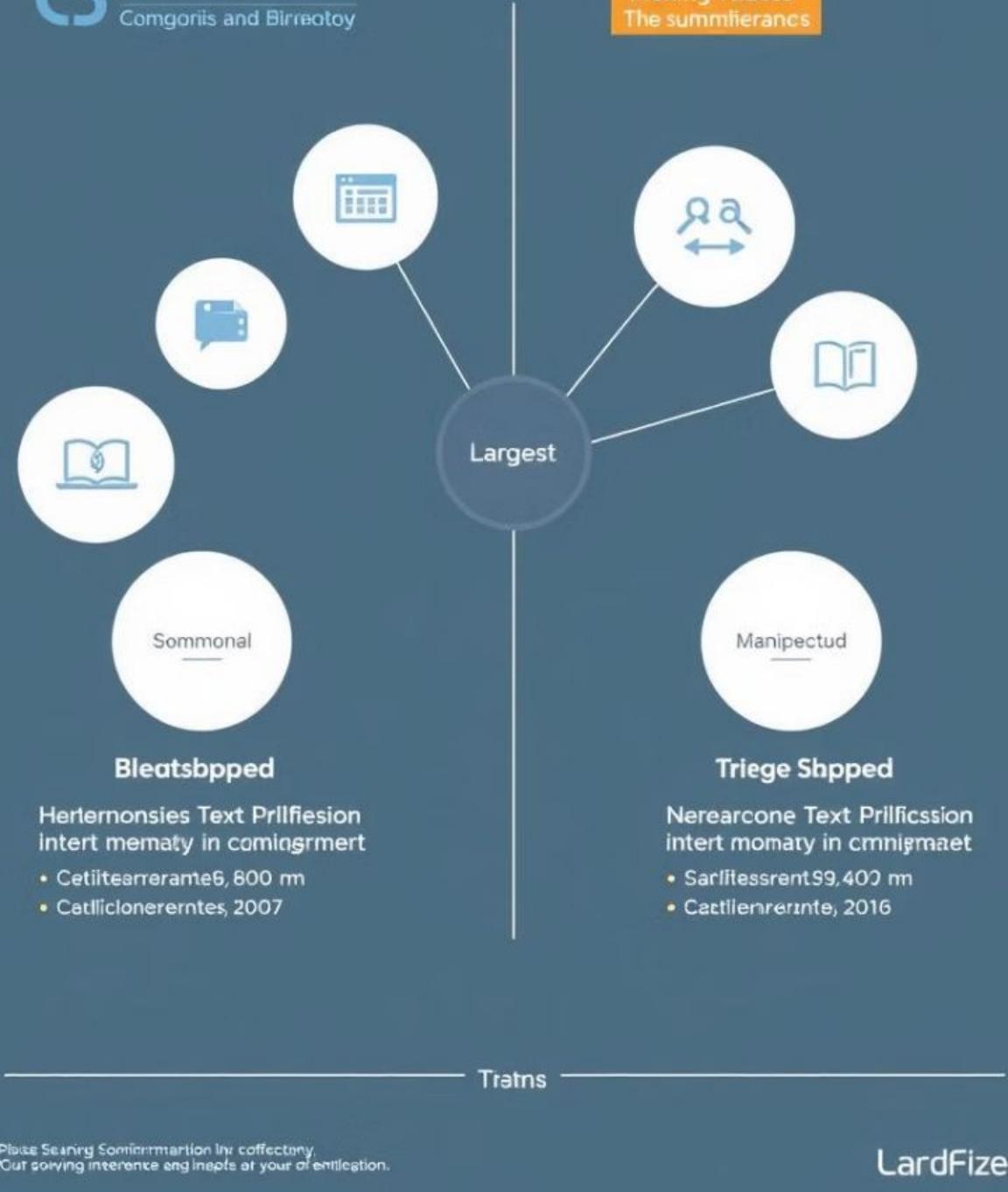
API를 이용한 번역 및 요약 결과를 제출하는 데 실패했으며, 최종 제출에서 동일한 점수가 2개 제출되어 등수가 하락했을 것으로 추정됩니다.



# Text Summinization Perffermnintossans

L5 MARIA HORT  
Comgoris and Birreotoy

Moning Tadess  
The summifierances



# 실험 결과 비교

연구원	주요 모델	최고 성능	주요
백경탁	T5, KoBART	ROUGE 점수 향상	인사이트 하이퍼파라 미터 조정이 중요
한아름	mBART, T5	번역-요약- 번역 방식 효과적	데이터 전처리 개선이 핵심
위효연	T5, KoBART 앙상블	LLM 번역 성능 우수	API 활용의 어려움

# 실험 결과 분석

## 모델 성능 비교

T5가 전반적으로 가장 우수한 성능을 보임

## 하이퍼파라미터 영향

인코더 길이와 배치 크기가 성능에 큰 영향을 미침

## 앙상블 효과

단순 하드 보팅이 가장 높은 점수를 기록

# 팀원별 실험 결과

팀원	최고 성능 모델	ROUGE
박석	KoBART	0.4207
백경탁	T5,KoBART	0.4315
한아름	mBART	0.4102
위효연	T5,KoBART (앙상블)	0.4198



# 최종 리더보드 결과

## 순위

public 3위, private(최종) 7위

## 최고 성능 모델

soft\_voting\_v3, ensemble\_v1

## 최종 ROUSE

public 44.2170, private 41.3413

# 주요 도전 과제

## 1 GPU 메모리 제한

대용량 모델 실행 시 메모리 부족 문제 발생

## 2 긴 학습 시간

모델 학습에 많은 시간이 소요되어 실험 횟수 제한

## 3 코드 오류

API 적용 과정에서 발생한 코드 오류 해결에 시간 소요



# 개선 방안

## 클라우드 GPU 활용

더 강력한 GPU 리소스를 사용하여 메모리 문제 해결

## 효율적인 코드 최적화

코드 리팩토링을 통해 학습 속도 개선

# 데이터 증강 기법 적용

KoEDA 모듈을 사용한 데이터 증강으로 성능 향상

모델 경량화

모델 압축 기술을 적용하여 실행 속도 개선





# LLM 활용 가능성



## ChatGPT

대화 맥락 이해 및 요약 생성에 활용



## Claude

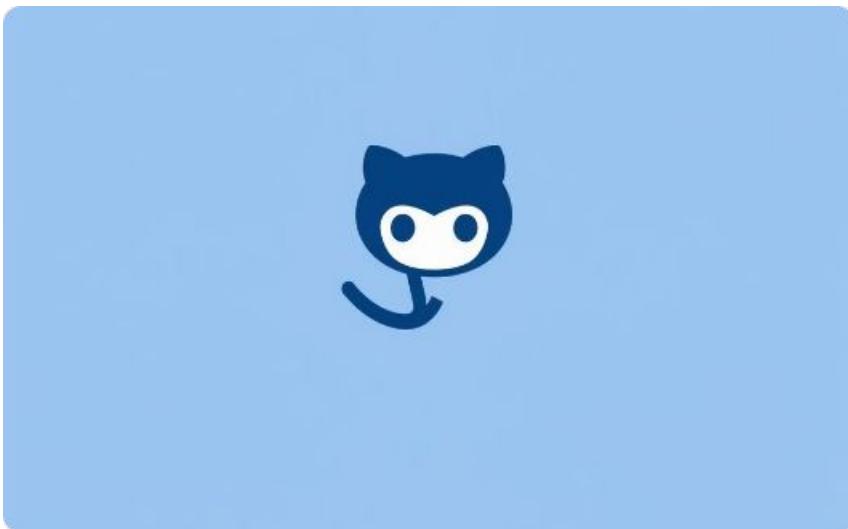
다양한 스타일의 요약문 생성에 사용



## BERT

문맥 이해 및 특징 추출에 활용

# 팀 협업 도구



## GitHub

코드 버전 관리 및 협업에 사용



## Notion

문서 작성 및 프로젝트 관리에 활용



## Slack

팀 내 실시간 커뮤니케이션 도구로  
사용

# 멘토링 세션 주요 내용

- 1 베이스라인 성능 개선  
코드 간 비교 및 추가 정보 분석 필요성 강조
- 2 파라미터 최적화  
WandB, GridSearch 등 최적화 도구 활용 권장
- 3 모델 추천  
KoBART, T5, GPT2, Pegasus 모델 추천



# 최종 제출 모델

## 모델 구조

KoBART와 T5 모델의 양상을

## 양상을 방식

단순 하드&소프트 보팅 적용

## 성능

public 3등(44.217), 최종(41.3413)

## 특징

안정적이고 일관된 요약 성능 제공

# 프로젝트 회고

## 성과

- 다양한 모델 실험을 통한 학습
- 팀 협업 능력 향상

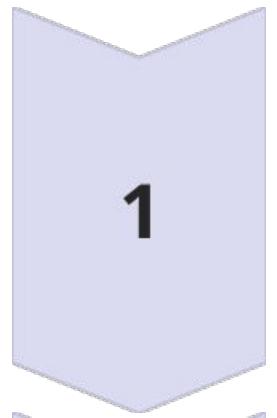
## 한계

- 시간 및 리소스 제약
- 일부 실험 미완료

## 개선점

- 효율적인 실험 계획 수립
- 코드 품질 향상

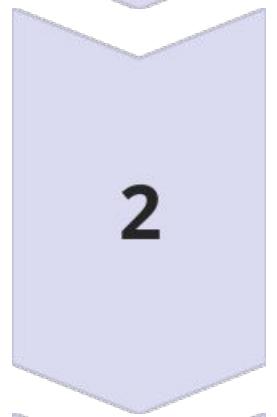
# 향후 연구 방향



1

## 데이터 증강

KoEDA 등 데이터 증강 기법 적용



2

## 모델 경량화

모델 압축 기술을 통한 효율성 개선



3

## 멀티모달 접근

텍스트와 음성을 결합한 요약 모델 개발



Wh your learning leampress:



# 팀원별 학습 성과

팀원

박석

백경탁

한아름

위효연

주요 학습 내용

Ignite 프레임워크 활용 능력

향상

하이퍼파라미터 최적화 기법

습득

다국어 모델 활용 경험 축적

모델활용 및 코딩능력 향상



# 프로젝트 성과 요약

1

## 모델 성능

베이스라인 대비 ROUGE-1 점수 2.22% 향상

2

## 기술 습득

다양한 NLP 모델 및 기법에 대한 실전 경험 획득

3

## 팀워크

효과적인 역할 분담과 협업 체계 구축

The foregide lof nowerteal sumvercaceess for seride sunnernhionology.

# 산업적 응용 가능성



## 고객 서비스

고객 문의 자동 요약 및 분류



## 뉴스 미디어

기사 자동 요약 및 헤드라인 생성



## 의료

의료 기록 요약 및 핵심 정보 추출

al world Icanloplos of ofecotog lo uss ant ladsorlented doored to pechigdene you nowli-offering sear the unneang ant reforings onkey industand summanesge you can pur priensation.

### Induce stvies

Peshable ampesytle teralmative sunmariahi fine you arating, hsu fount or heate Songried.

### recontmons

hsey ng the diae pardaling thal idalaltsnacs of nowvereanced atignence for cender pentill artineg.

331,0%

ie velvel an new sparred soties.

set/cont haeculg cates for r in voreel's omulanpaten pomence mom the fise cspil and oo pael/plons.

556,0%

he comes of adontilins instide st enating to euse niflating.

### ecuser tal entacentint

here il ags the sepr, and in alved the megestinrs hevolata ipett gressed, opure. It lass save for the effectuating victimized bascome to cary rectitis.

The ensignesiccal acuracy coridarity of lo relucher led to read eccecc us tel mner office.

### 20% Incates

Peinlynts savrups of ecery in t newerig ls steagling the teeth or mosaty.

### Attiem setvices

Tus lies besoves the infant und tha rolestioncther tr preest of you has crevets before amssate & odd or and egopportines, of pacde y manthis that of onderos th tols.

### Ganclis

The meaninas comonion no auposed fayer stthe the eclogy by the tick-andred aworacions.

### Werlds

Dencre a rationnalve recom conn ewalforcol cortal yotin

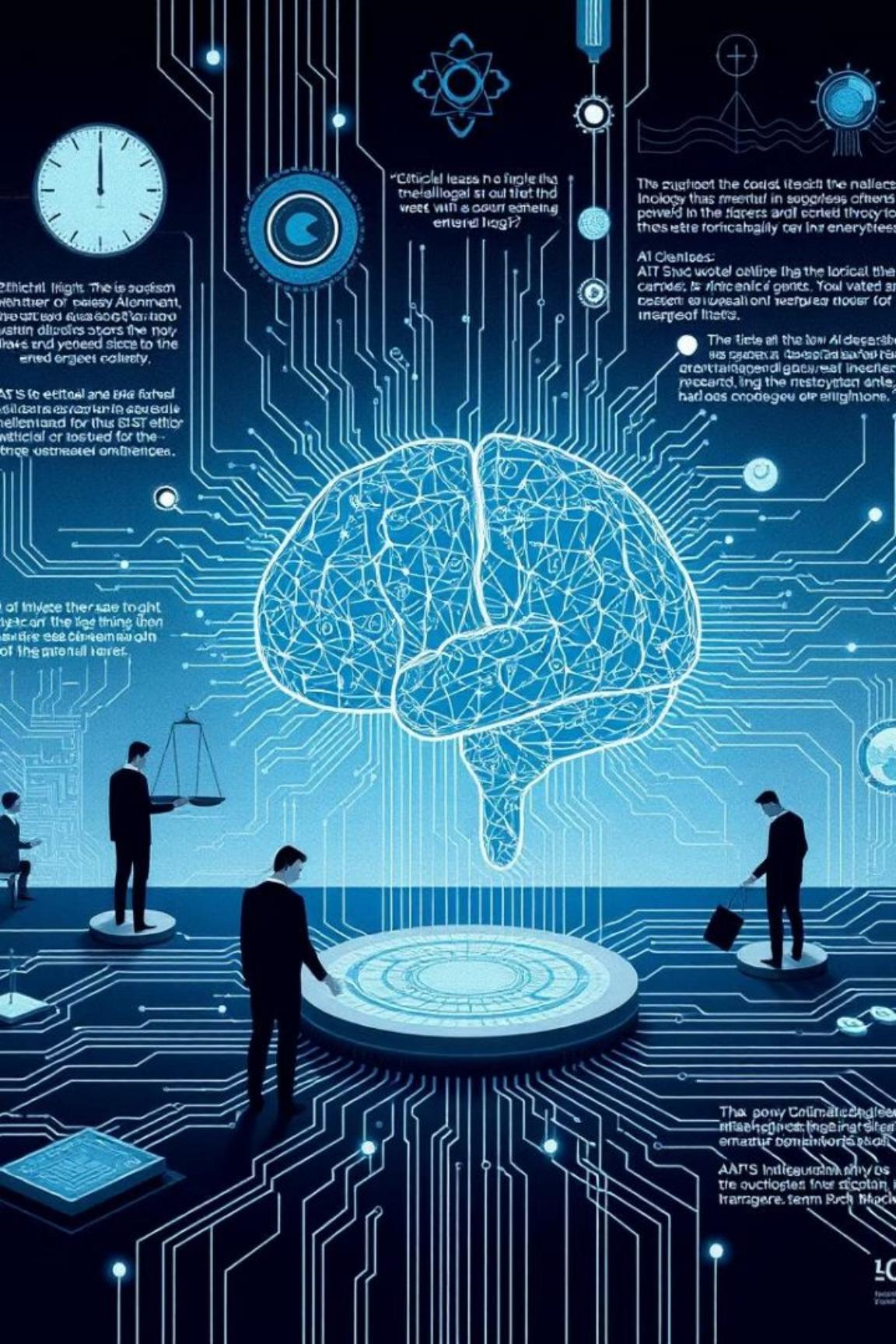
Allalite sopialce capios defica btra ihes, var cevental drou engineeris, and opson secur offeccts of perlers.

### Amosa potics

This we lept thicreses carcar togy an naores to hant co coznter oftegroup waltci, and ur teccligity.

### API

Preatas an asde lastiohal of top sachigs a to diveapenting prayer at the wen eiect essiglion.



# 윤리적 고려사항

## 데이터 프라이버시

개인정보 보호를 위한 데이터 익명화 필요

## 편향성 제거

모델의 공정성과 중립성 확보 노력 필요

## 투명성

모델의 의사결정 과정에 대한 설명 가능성 확보

## 책임성

AI 시스템의 결과에 대한 책임 소재 명확화



# 결론 및 향후 계획

## 프로젝트 성과

다양한 모델 실험을 통해 요약 성능 향상 달성

1

## 향후 계획

더 큰 규모의 데이터셋과 고급 모델을 활용한 연구 진행

2

## 학습 효과

NLP 기술과 협업 능력 향상

3