BERT

노민준

NLP Study

2024/04/30

TRAIN AND TEST

Contents

- Introduction
- BERT
- Experiments

TRAIN AND TEST

BERT

BERT

Bidirectional

Encoder

Representation

Transformer

➡ Transformer의 encoder를 이용해 양방향 representation을 얻어내는 방법

TRAIN AND TEST

Introduction

Natural language processing task

- Sentence-level task

➢ 문장들 전체적으로 분석 -> 문장 간의 관계를 예측

Ex) NLI (자연어 추론) : 주어진 문장 (hypothesis)과 다른 문장(premise)에 대해 논리적으로 들어맞는지 판단하는 것을 목표

Ex) Paraphrasing (의역) : 문장들을 다른 방식으로 표현

TRAIN AND TEST

Introduction

Natural language processing task

- Token-level task

Ex) NER (개체명 인식) : 단어를 문장에서 추출해, 인명/지명/기관명 등으로 분류하는 것

Ex) Question Answering (질의 응답) : 질문에 대한 답변 추출

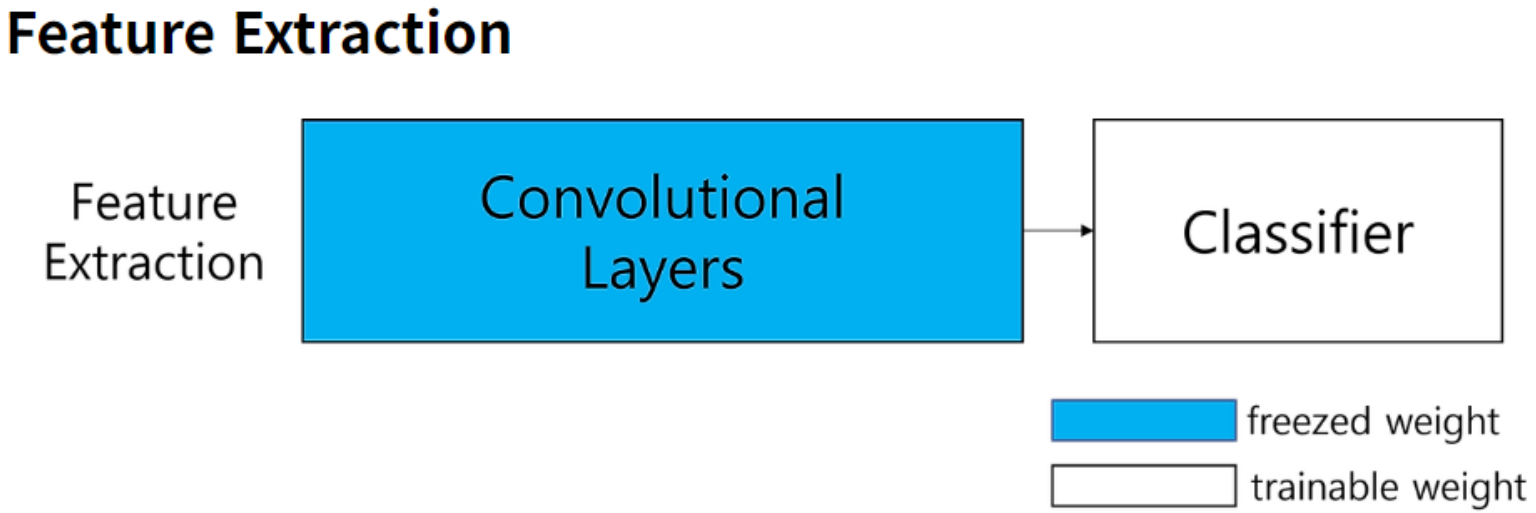
TRAIN AND TEST

Introduction

Two Strategies for pre-training
1. feature-based

- Embedding은 그대로 두고, 그 위의 layer들만 학습하는 방법

Ex) Elmo



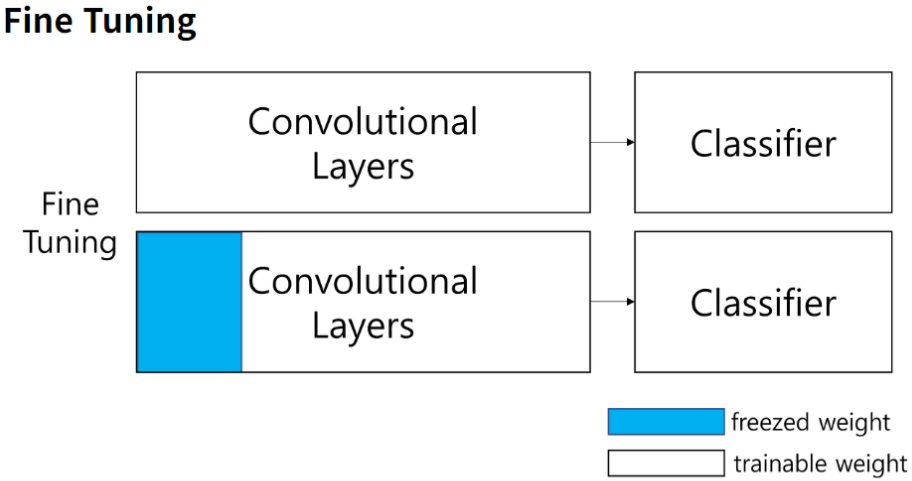
Feature Extraction

Introduction

Two Strategies for pre-training
2. fine-tuning

- Embedding까지 모두 update하는 방식
- 이 때, 기존의 Weight를 최대한 망가뜨리지 않고,
  새로운 데이터들의 feature들을 조금씩만 학습하기 위해 learning rate를 아주 작게 설정해줌

Ex) Open AI GPT-1



Fine Tuning

TRAIN AND TEST

Introduction

Fine – tuning에서 GPT의 한계점

- OpenAI GPT는 left-to-right의 모델을 사용
  - ➢ Self – attention layer에서 이전 token들만 이용하게 됨
  - ➢ Sentence-level task나 Question Answering과 같은 Token-level task에서는 양방향의 문맥을 모두 포함해야 함
  - ➢ 위와 같은 Task에서는 적용이 어려움

TRAIN AND TEST

Introduction

BERT는 이렇다

- Masked Language Model (MLM)의 사용
- GPT같은 Left – to – right model과는 다르게 left & right 문맥을 융합한 representation 생성 가능함
- Next Sentence Prediction (NSP)의 사용
- Text – pair도 jointly하게 학습이 가능함

TRAIN AND TEST

BERT

Two Step

1. **Pre-training**

- 각 pre-training task에 대해 unlabeled data로 training

2. **Fine-tuning**

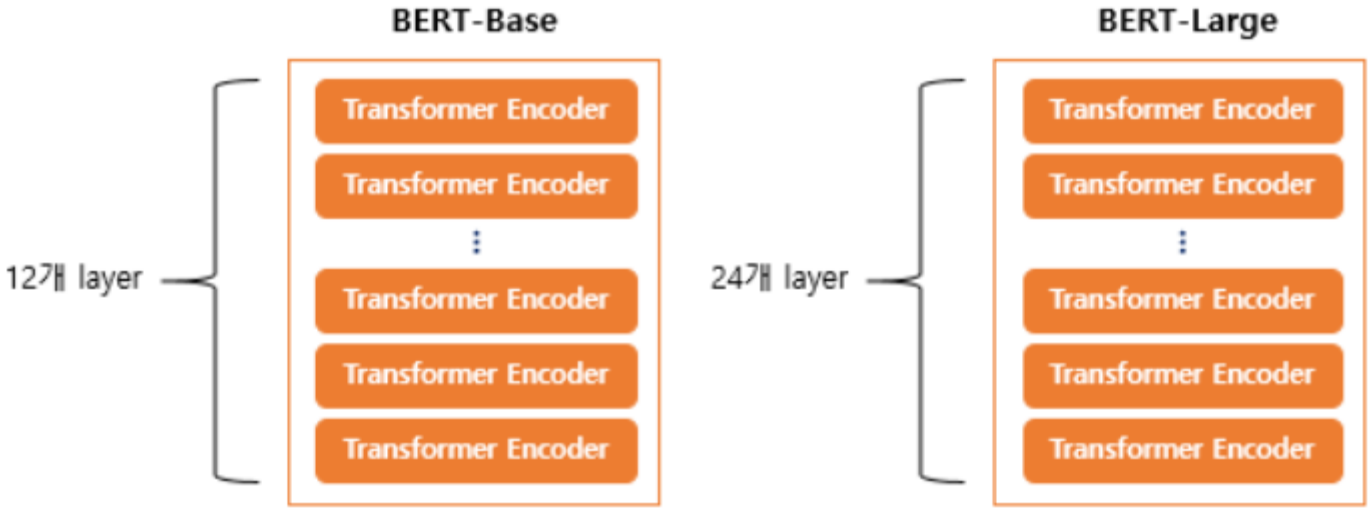
- Pre-trained된 parameter에서 시작해, downstream task의 labeled data를 이용해 fine-tuning해 나감
- 같은 parameter로부터 시작했을지라도, 각 downstream task들은 각자의 fine-tuned된 model을 갖게 됨

TRAIN AND TEST

BERT

Model Architecture

- 2가지 model size를 이용
- ✓ $BERT_{base}$
- ✓ $BERT_{large}$
- $BERT_{base}$ 는 OpenAI GPT와 비교를 위해 동일한 size로 설정함



TRAIN AND TEST

BERT

Model Architecture

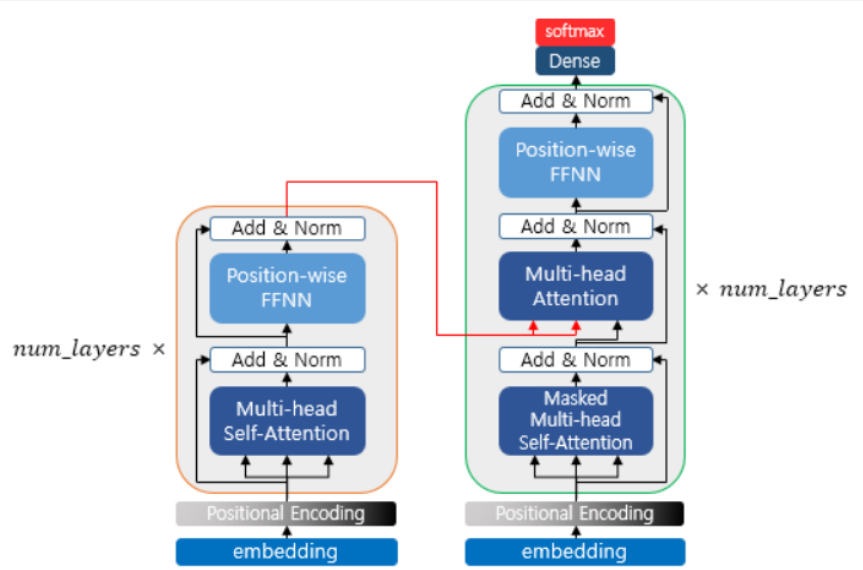
BERT에서 Decoder가 아니라 Encoder를 사용하는 이유?

TRAIN AND TEST

BERT

Model Architecture

- **Decoder**의 Masked Multi-Head Self-Attention에서, 현재 시점보다 미래 시점의 단어들을 참고하지 못하도록 look-ahead mask 사용함
- **Encoder**에선 masking없이 전체 단어를 이용해 self-attention 진행 => MLM 이용이 가능해짐



TRAIN AND TEST

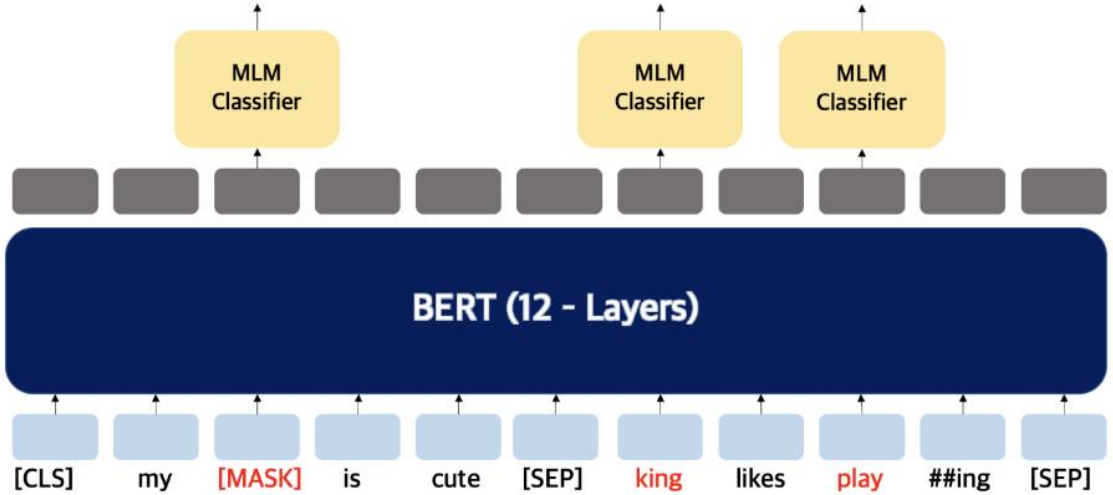
BERT

I/O Representation

- Input에서 single sentence뿐 아니라 pair of sentences도 one token sequence로 만들어 이용할 수 있음

✓ 일반적 의미의 Sentence : “S + V + O”로 이루어진, 의미를 가지는 문장

✓ BERT에서의 Sentence : 연속적인 text들의 span



TRAIN AND TEST

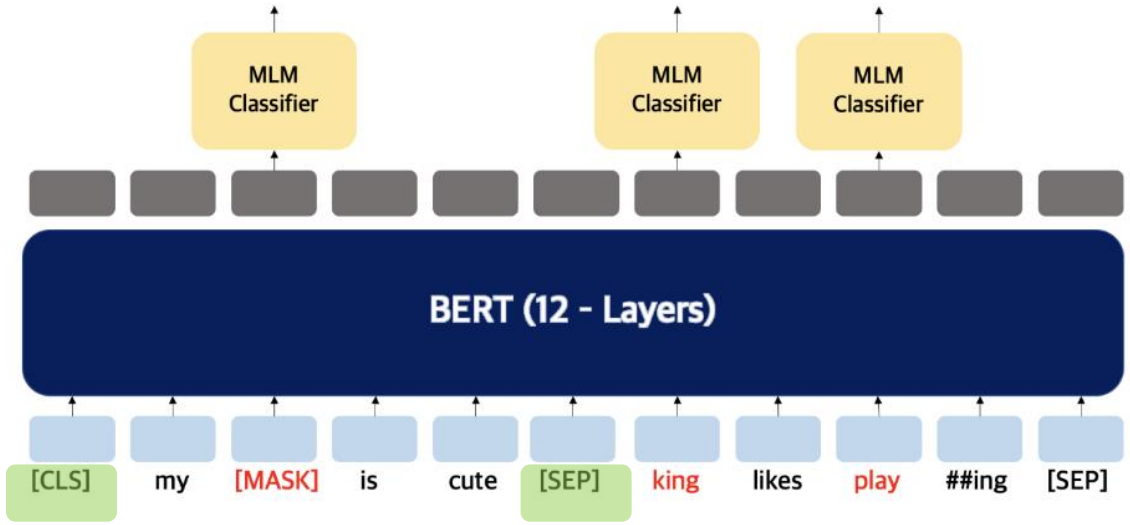
BERT

I/O Representation

- [CLS] : 모든 sequence들에서 처음으로 사용되는 special token

✓ 이 token의 final hidden state는 classification task를 수행할 때 사용됨

- [SEP] : single sentence가 아니라, sentence 쌍들을 input으로 넣어줄 때, 두 sentence들을 구분하기 위해 사용됨



TRAIN AND TEST

BERT

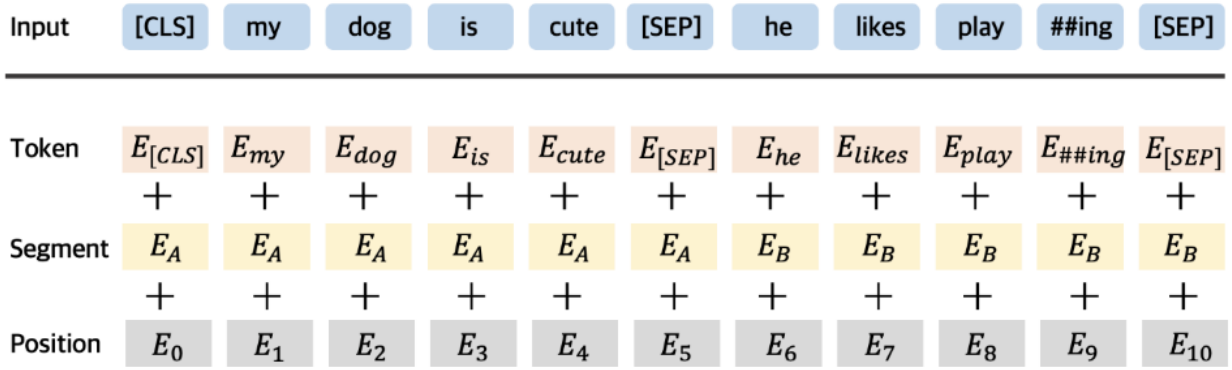
Input Representation

- Token Embedding, Segment Embedding, Position Embedding을 다 더해서 넣어줌

✓ Token Embedding : Word Piece tokenizer을 이용해 토큰화해준 부분

✓ Segment Embedding : 각 token이 몇 번째 문장에 속하는지 알려주는 부분

✓ Position Embedding : 각 token이 몇 번째 token인지 알려주는 부분



TRAIN AND TEST

BERT

Word Piece Tokenizer

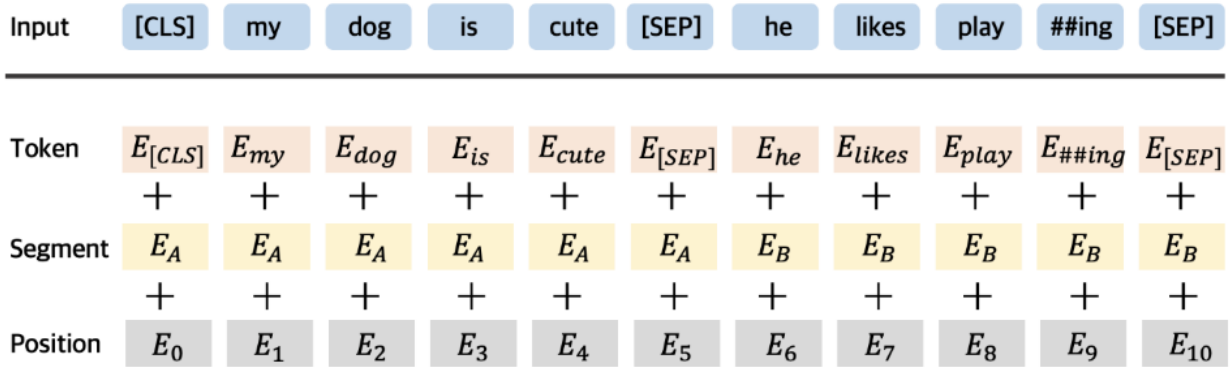
- Subword tokenizer : 단어보다 더 작은 단위로 쪼개주는 것

✓ Token이 단어 집합에 존재 => 분리 X

✓ Token이 단어 집합에 존재 X => 해당 토큰을 subword로 분리

✓ 첫 번째 subword를 제외한 나머지는 앞에 ##을 붙여줌

Ex) Embeddings = > Em, ##bed, ##ding, #s 로 분리



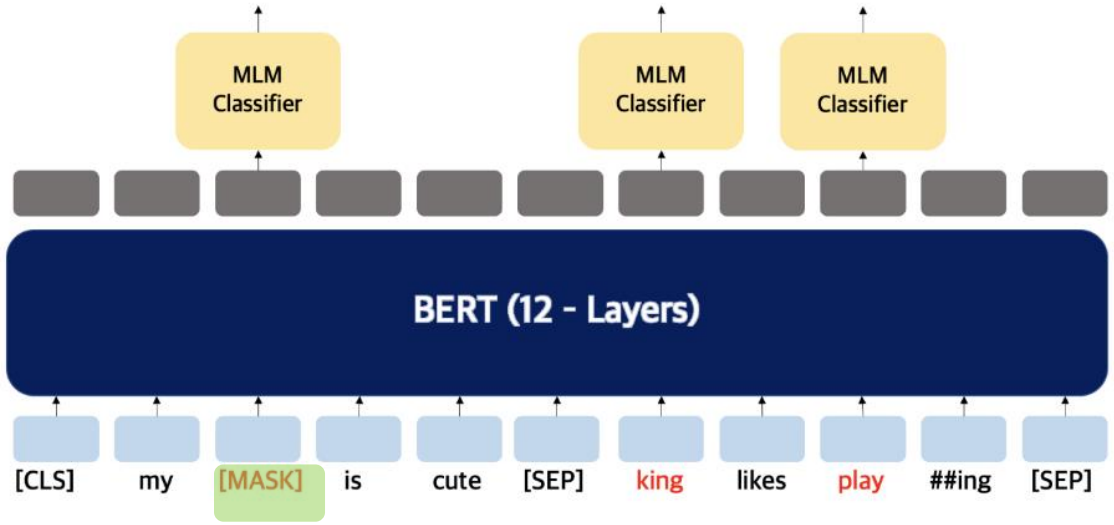
TRAIN AND TEST

BERT

Masked LM

- Input으로 들어온 sequence에서 일정 비율의 token들을 가려줌
- 이후 가려준 token들을 cross entropy loss를 이용해 예측하면서 학습을 진행

✓ BERT에서는 15% 비율로 token을 가려줌



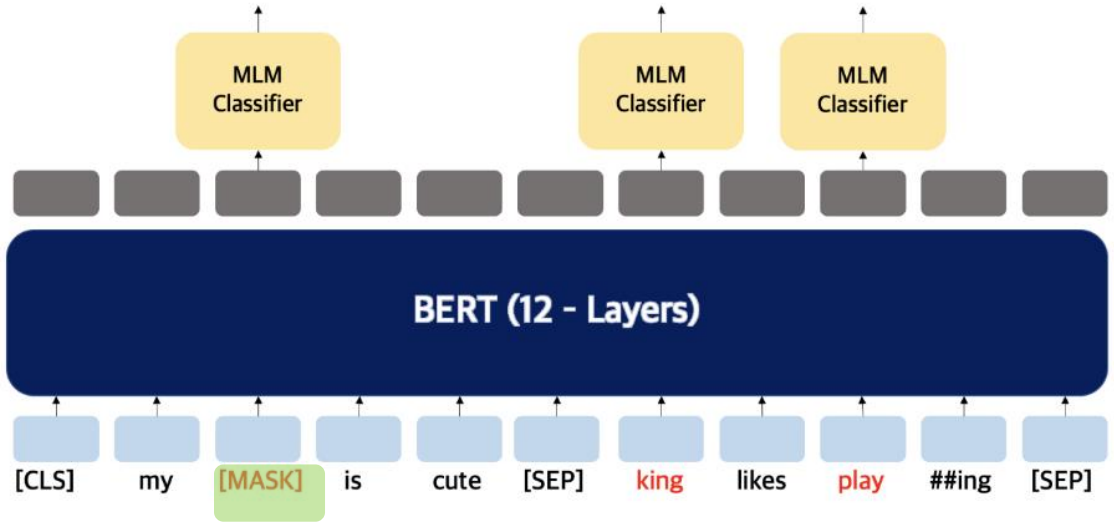
TRAIN AND TEST

BERT

Masked LM Problem

- **Pre-training**에서는 일정 비율만큼 [MASK] token으로 바꿔주면서 학습을 진행
- **Fine-tuning**에서는 이와 달리 [MASK] token이 아예 존재 x

➢ Pre-training과 Fine-tuning 사이에 **Mismatch** 발생



TRAIN AND TEST

BERT

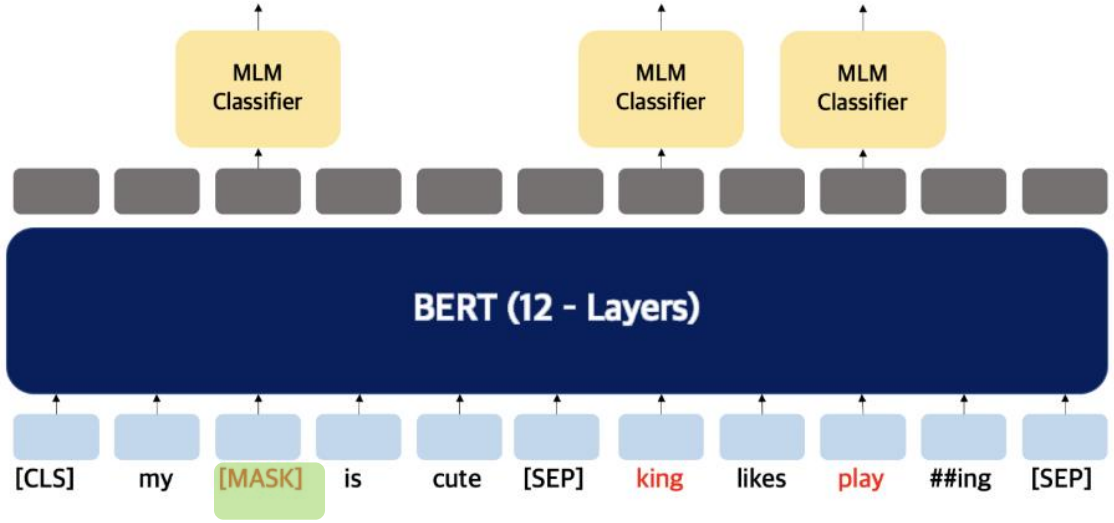
Solution

- 15%의 선택된 token들을 모두 [MASK] token으로 바꾸지 X

✓ **80%** : 예정대로 [MASK] token으로 바꿔줌

✓ **10%** : 원래 token 유지

✓ **10%** : random token으로 change



TRAIN AND TEST

BERT

Solution

- Masking Rates를 다양하게 설정해 실험
- 80 / 10 / 10 %로 나누었을 때의 Performance가 가장 좋아서 채택

Masking Rates			Dev Set Results		
MASK	SAME	RND	MNLI	NER	
			Fine-tune	Fine-tune	Feature-based
80%	10%	10%	84.2	95.4	94.9
100%	0%	0%	84.3	94.9	94.0
80%	0%	20%	84.1	95.2	94.6
80%	20%	0%	84.4	95.2	94.7
0%	20%	80%	83.7	94.8	94.6
0%	0%	100%	83.6	94.9	94.6

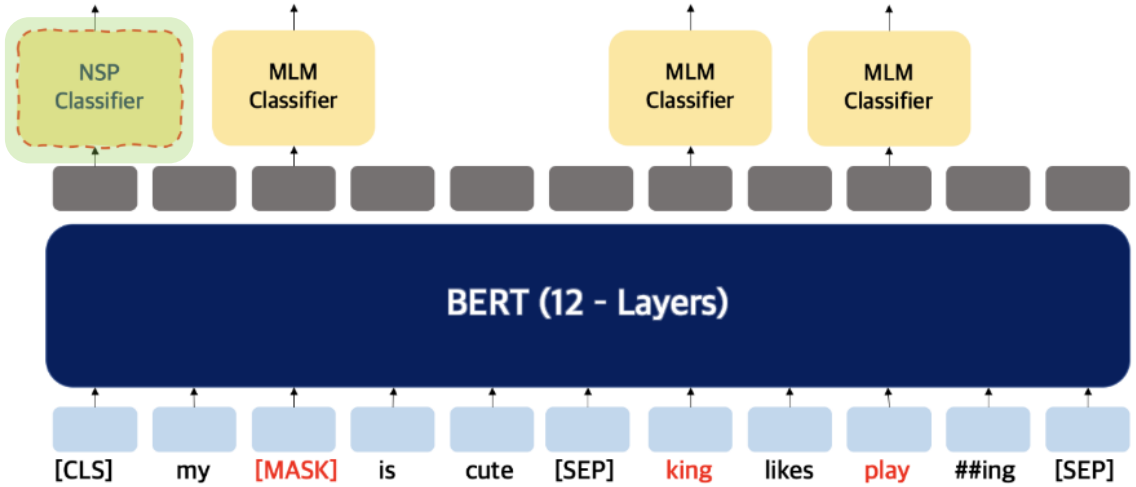
TRAIN AND TEST

BERT

NSP

- NLI (자연어 추론)이나 QA (질의 응답)과 같은 task : 두 문장 간의 유기성을 이해하는 것을 기반으로 함
- 하지만 직접적으로 포착하기엔 어려움이 있음

➢ Binarized된 next sentence prediction task를 pre-train하면서, 문장간 유기성 이해하게 만듦

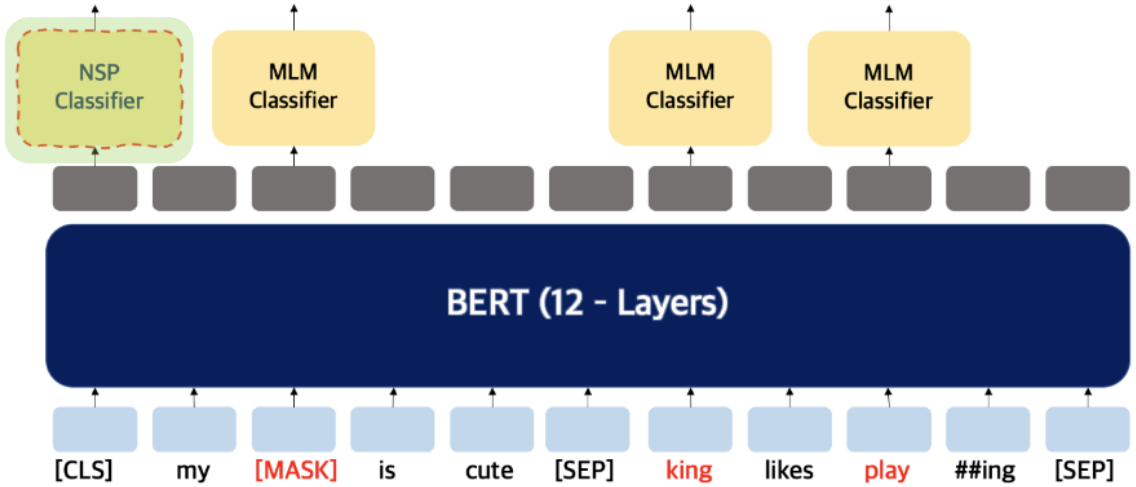


TRAIN AND TEST

BERT

NSP

- ✓ 50% : 두 sentence A, B에 대해, B가 실제로 A의 next sentence임
- ✓ 50% : corpus에서 random한 문장을 가져와 B로 사용 (B가 A의 next sentence가 아님)
- ✓ NSP Classifier : next sentence가 맞는지 예측하는데 사용됨

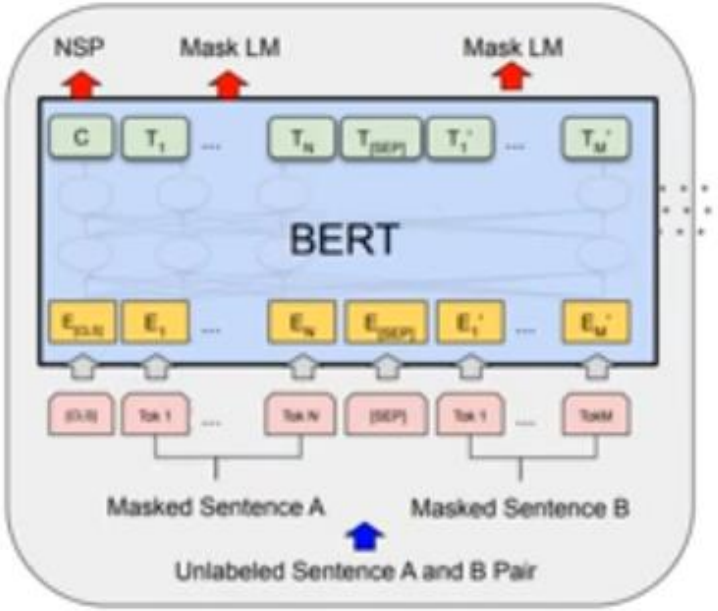


TRAIN AND TEST

BERT

Pre-training 요약

- ✓ Masked Sentence A : N개의 token을 가짐
- ✓ Masked Sentence B : M개의 token을 가짐
- ✓ Unlabeled Sentence A and B pair => Sequence
- ✓ [CLS] : Sequence의 시작을 알려줌
- ✓ [SEP] : Sentence A 와 B 사이를 구분해줌
- ✓ C : Special Token인 [CLS]의 final hidden vector
- ➢ 이후 감정분석 or simularity분석과 같은 binary classification을 수행할 때 사용됨
- ✓ Mask LM : [MASK] token에 들어갈 token을 예측하는데 사용됨



TRAIN AND TEST

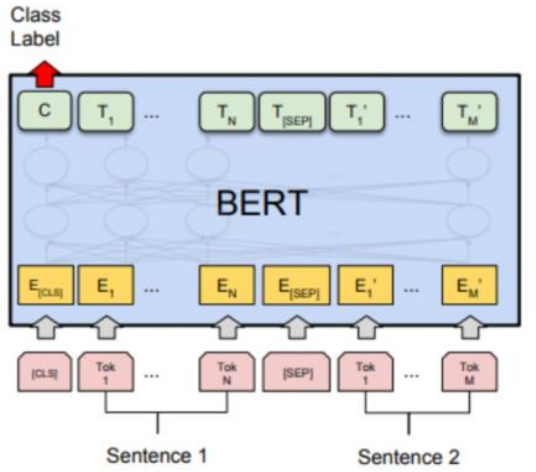
BERT

Different Tasks

1. Sentence Pair Classification Tasks

- Sentence 2개를 input으로 받음
- 해당 Sequence가 특정 class에 속하는지 아닌지에 대한 Task 수행

Ex) 두 문장 사이의 Similarity 측정, 모순 관계(contradiction) 측정, 함의 관계(entailment) 측정



TRAIN AND TEST

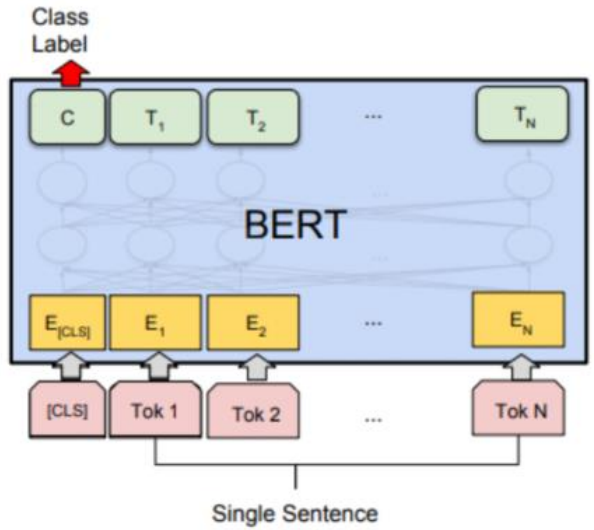
BERT

Different Tasks

2. Single Sentence Classification Tasks

- Sentence 1개를 input으로 받음
- 해당 Sequence가 어떤 class에 속하는지에 대한 Task 수행

Ex) 영화 리뷰 감성 분류, 뉴스 분류



TRAIN AND TEST

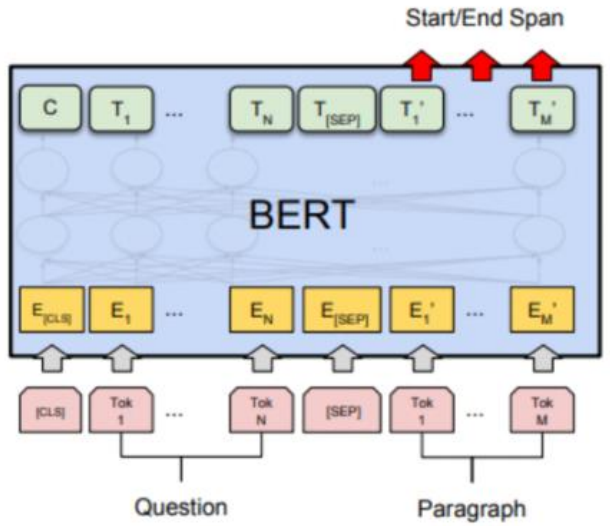
BERT

Different Tasks

3. Question Answering Tasks

- Input sequence에 대해 문장을 생성

Ex) 질의응답(QA)



TRAIN AND TEST

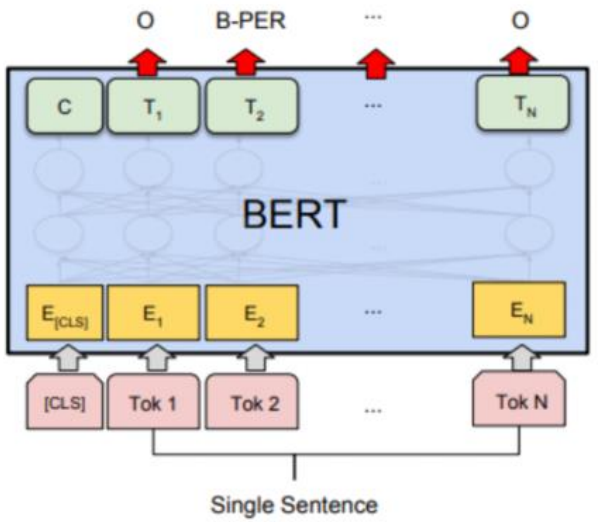
BERT

Different Tasks

4. Single Sentence Tagging Tasks

- Sentence 1개를 input으로 받음
- 각각의 token에 대해 output token 도출

Ex) 형태소 분석, 개체명 인식



TRAIN AND TEST

Experiments

Model Evaluation

- ELMO < OpenAI GPT < BERT_base < BERT_large

System	MNLI-(m/mm) 392k	QQP 363k	QNLI 108k	SST-2 67k	CoLA 8.5k	STS-B 5.7k	MRPC 3.5k	RTE 2.5k	Average -
Pre-OpenAI SOTA	80.6/80.1	66.1	82.3	93.2	35.0	81.0	86.0	61.7	74.0
BiLSTM+ELMo+Attn	76.4/76.1	64.8	79.9	90.4	36.0	73.3	84.9	56.8	71.0
OpenAI GPT	82.1/81.4	70.3	88.1	91.3	45.4	80.0	82.3	56.0	75.2
$BERT_{BASE}$	84.6/83.4	71.2	90.1	93.5	52.1	85.8	88.9	66.4	79.6
$BERT_{LARGE}$	**86.7/85.9**	**72.1**	**91.1**	**94.9**	**60.5**	**86.5**	**89.3**	**70.1**	**81.9**

TRAIN AND TEST

Experiments

About NSP

- 기존 BERT_base
- 기존 BERT_base에서 NSP과정 제외

✓ 모든 Task에서 조금씩 score가 감소

- Left-to-right model + NSP 과정 제외

✓ 마찬가지로 모든 task에서 조금씩 score가 감소

- Left-to-right model + NSP 과정 제외 + 임의로 초기화된 BiLSTM 추가

✓ SQuAD에서 점수 크게 향상

Tasks	Dev Set MNLI-m (Acc)	QNLI (Acc)	MRPC (Acc)	SST-2 (Acc)	SQuAD (F1)
$\text{BERT}_{\text{BASE}}$	84.4	88.4	86.7	92.7	88.5
No NSP	83.9	84.9	86.5	92.6	87.9
LTR & No NSP	82.1	84.3	77.5	92.1	77.8
+ BiLSTM	82.1	84.1	75.7	91.6	84.9

TRAIN AND TEST

Experiments

About NSP

- **MNLI** : 한 쌍의 문장에 대해, 두 번쨰 문장이 entailment/contradiction/neutra인지 분류
- **QNLI** : SQuAD의 이진분류 버전
- **MRPC** : 두 문장이 의미적으로 유사한지 판단
- **SST-2** : 영화리뷰에 대한 sentiment analysis
- **SQuAD** : Stanford Qu estion Answering Dataset

✓ 3번째 task type으로 사용할 때, 유용할 수 있음

	Dev Set				
Tasks	MNLI-m (Acc)	QNLI (Acc)	MRPC (Acc)	SST-2 (Acc)	SQuAD (F1)
$\text{BERT}_{\text{BASE}}$	84.4	88.4	86.7	92.7	88.5
No NSP	83.9	84.9	86.5	92.6	87.9
LTR & No NSP	82.1	84.3	77.5	92.1	77.8
+ BiLSTM	82.1	84.1	75.7	91.6	84.9

TRAIN AND TEST

Experiments

Hyperparameter

- 4번째 : BERT base
- 6번째 : BERT large

✓ 모델의 크기가 커질수록 성능이 높아지는 것 확인 가능

Hyperparams			Dev Set Accuracy			
#L	#H	#A	LM (ppl)	MNLI-m	MRPC	SST-2
3	768	12	5.84	77.9	79.8	88.4
6	768	3	5.24	80.6	82.2	90.7
6	768	12	4.68	81.9	84.8	91.3
12	768	12	3.99	84.4	86.7	92.9
12	1024	16	3.54	85.7	86.9	93.3
24	1024	16	3.23	86.6	87.8	93.7

TRAIN AND TEST

Experiments

About Fine-tuning

<Fine-tuning하지 않았을 때>

- Embedding만 진행
- 2번째부터 마지막 hidden layer까지 사용
- 마지막 hidden layer만 사용
- 마지막 4개 hidden layer들의 output들의 weighted sum
- 마지막 4개 hidden layer들의 concat
- 12개 hidden layer들의 output들의 weighted sum

✓ 마지막 3가지는 fine-tuning을 진행한 BERT 자체 성능과 큰 차이 X

System	Dev F1	Test F1
ELMo (Peters et al., 2018a)	95.7	92.2
CVT (Clark et al., 2018)	-	92.6
CSE (Akbik et al., 2018)	-	**93.1**
Fine-tuning approach		
$\text{BERT}_{\text{LARGE}}$	96.6	92.8
$\text{BERT}_{\text{BASE}}$	96.4	92.4
Feature-based approach ($\text{BERT}_{\text{BASE}}$)		
Embeddings	91.0	-
Second-to-Last Hidden	95.6	-
Last Hidden	94.9	-
Weighted Sum Last Four Hidden	95.9	-
Concat Last Four Hidden	96.1	-
Weighted Sum All 12 Layers	95.5	-

TRAIN AND TEST

TRAIN AND TEST

RoBERTa

노민준

NLP Study

2024/04/30

TRAIN AND TEST

Contents

- Introduction
- Background
- Experimental Setup

TRAIN AND TEST

Introduction

BERT

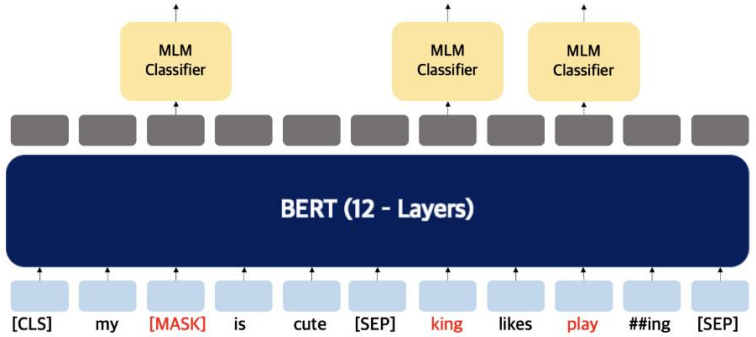
- BERT는 상당히 undertrained 되어있음
- ➢ 4가지 수정 사항을 통해 개선이 가능함
- ✓ Batch의 크기를 키우고, 더 많은 data를 이용해 더 오래 학습해야 함
- ✓ NSP 삭제
- ✓ 더 긴 Sequence를 이용해 학습
- ✓ Masking pattern을 바꿔줌

TRAIN AND TEST

Background

BERT

- Two sentence를 합쳐 sequence로 만들고 input으로 사용
- [CLS], [SEP]의 special token을 사용
- Unlabeled corpus를 통해 먼저 pretrain 한 후에, labeled된 data를 이용해 fine-tuning 진행
- Transformer의 encoder를 이용
- Bookcorpus와 Wikipedia를 이용해 훈련 진행
- $\beta_1 = 0.9, \beta_2 = 0.999, \epsilon = 1e-6, L2\ weight\ decay(L2\ penalty) = 0.01$로 Adam 최적화 진행



Background

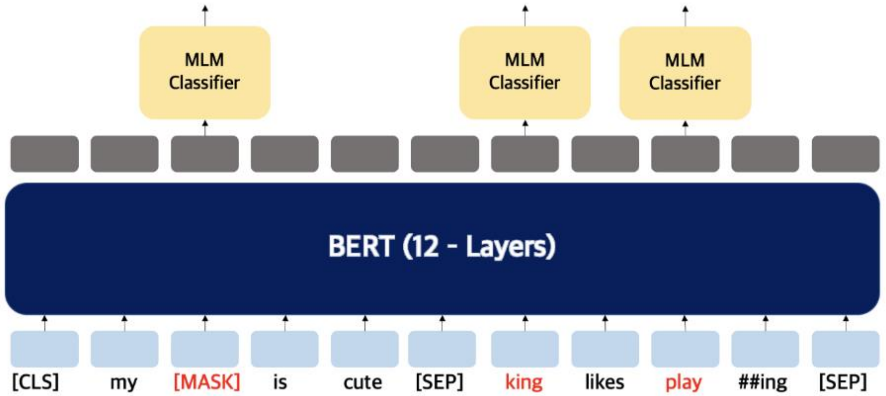
BERT

- Pretraining과정에서 MLM, NSP의 두 과정 진행
- MLM

✓ Sequence의 input token들 중에서 15%를 random하게 선택해 [MASK] token으로 바꾸어 주고, 원래 token을 예측하면서 학습하는 과정

✓ Pretraining과 fine-tuning 과정 사이에 mismatch가 존재하기 때문에 15% 전부를 [MASK] token으로 바꿔주지 X

✓ 80% : [MASK], 10% : unchanged, 10% : randomly selected



TRAIN AND TEST

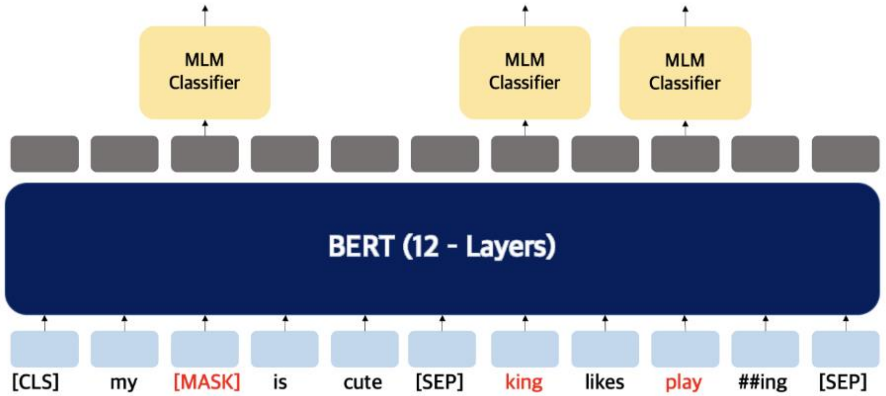
Background

BERT

- Pretraining과정에서 MLM, NSP의 두 과정 진행
- NSP

✓ Input으로 들어온 두 sentence 중 B가 A 다음 문장인지 예측해주는 binary classification 수행

✓ 두 문장 사이의 관계를 추론해야 하는 task에서의 성능을 높여줌



TRAIN AND TEST

Experimental Setup

Implementation

- Adam 최적화를 진행할 때, training 과정이 epsilon값에 매우 민감하게 반응함을 발견
  - ➢ 살짝 tuning 한 후에 더 좋은 performance를 가지고 stability가 증가했음을 확인
- β_2 값 역시 더 큰 batch size로 훈련할 때 0.98로 수정해주면 stability가 증가함도 확인
- Full length sequence만을 이용해 training 진행

TRAIN AND TEST

Experimental Setup

Data

- Bert와 같은 model들은 data의 양에 의존적임
- 2019년 논문에서 data size를 키우면 end-task performance가 향상됨이 밝혀짐

➢ 더 크고 다양한 dataset 이용

✓ Bookcorpus, Wikipedia

✓ CC-News : 6300만개의 news article

✓ OpenWebText : Reddit의 web content

✓ Stories : CommonCrawl에서 가져온 dataset

TRAIN AND TEST

TRAIN AND TEST