



DAICON 월간 데이콘 21 | Code NLP

# 코드 유사성 판단

# AI 경진대회



# 목차

**1. 대회 소개 및 데이터 설명**

**2. 데이터 전처리**

**3. 모델 소개**

**4. 실험 구성 및 결과**

**5. 결론 및 한계점**

01

# 대회 소개 및 데이터 설명

## 대회 소개



DAICON

커뮤니티

대회

교육

랭킹

더보기



em

### 코드 유사성 판단 AI 경진대회

월간데이콘21 | AI프렌즈 | 대전 AI | Code NLP

₩ 상금 : 총 600만 원

🕒 2022.05.02 ~ 2022.06.10 16:59

+ Google Calendar

👤 720명

📅 마감



참여중

두 코드간 유사성(동일 결과물 산출 가능한지) 여부를 판단할 수 있는 AI 알고리즘 개발

## 데이터 설명

300개의 문제에 대한 코드 중, 17,970개의 code pair

- 서로 **다른** 문제를 해결하는 코드일 경우: **0**
- 서로 **같은** 문제를 해결하는 코드일 경우: **1**

	code1	code2	similar
0	flag = "go"\n cnt = 0\n while flag == "go":\n ...	# Python 3+\n#-----...	1
1	b, c = map(int, input().split())\n\n print(b * c)	import numpy as np\n\n nn = int(input())\n na = np...	0
2	import numpy as np\n\n import sys\n\n read = sys.std...	N, M = map(int, input().split())\n\n if M%2 != 0:...	0
3	b, c = map(int, input().split())\n\n print(b * c)	n,m=map(int,input().split())\n\n h=list(map(int,i...	0
4	s=input()\n t=input()\n\n ans=0\n\n for i in range(le...	import math\n\n a,b,h,m=map(int,input().split())\n...	0

```
flag = "go"
cnt = 0
while flag == "go":
    cnt += 1
    x = int(input())
    if x == 0:
        flag = "stop"
    else:
        print("Case " + str(cnt) + ": " + str(x))
```

```
Python 3+
#-----
import sys

ff = open("test.txt", "r")
ff = sys.stdin
cnt = 1
for line in ff:
    if int(line) == 0:
        break
    print("Case {0}: {1}".format(cnt, line), end="")
    cnt += 1
```

## 추가 데이터 설명

- ✓ 학습용으로 주어지는 **300개의 문제**에 대한 코드

```
code [Folder] : 학습용으로 주어지는 300개의 문제에 대한 코드
  problem001 : 문제 번호
    problem001_1.py : 문제(001)를 해결하는 솔루션 코드 1
    problem001_2.py : 문제(001)를 해결하는 솔루션 코드 2
  problem002 : 문제 번호
    problem002_1.py : 문제(002)를 해결하는 솔루션 코드 1
    problem002_2.py : 문제(002)를 해결하는 솔루션 코드 2
  problem002_...
  ...
```

- ✓ 평가 지표 : **Accuracy**

- 전체 샘플의 개수들 중에서 얼마나 나의 알고리즘이 정답이라고 예측한 샘플이 포함되었는지의 비율

## 데이터 전처리의 필요성

```
import sys
input = sys.stdin.readline
```

```
def I(): return int(input())
def MI(): return map(int, input().split())
def LI(): return list(map(int, input().split()))
```

```
def main():
    mod=998244353
    N,S=MI()
```

```
    dp=[[0]*(S+1) for _ in range(N+1)]
    #dp[i][j]はi番目までで、和がjになるのが何個作れるか。
    #ただし、その数を使わない場合、選択するかしないか選べるのでダブルでカウント
```

```
    dp[0][0]=1
```

```
    for i in range(N):
        #dp[i][0]+=1
        for j in range(S+1):
            dp[i+1][j]+=dp[i][j]*2
            dp[i+1][j]%=mod
            if j+A[i]<=S:
                dp[i+1][j+A[i]]+=dp[i][j]
                dp[i+1][j+A[i]]%=mod
```

```
    print(dp[-1][-1])
```

```
main()
```

## Code Data

주식 및 픽어쓰기 존재

-> 언어 모델 성능에 영향 줄 수 있으므로 제거 필요

(1) 주식 삭제

(2) ‘ ‘ tab 변환

(3) 다중 개행 한 번으로 변환

## 데이터 전처리의 방식

- ✓ Sample pair data 외에, 추가로 제공된 Code 활용하여 추가 Pair data 생성

### I 키워드 기반의 랭킹 알고리즘 - BM25 사용

- 주어진 쿼리에 대해 문서와의 연관성을 평가하는 랭킹 함수로 사용되는 알고리즘
- Bag-of-words 개념을 사용하여 쿼리에 있는 용어가 각각의 문서에 얼마나 자주 등장하는지를 평가

### I Pair Data

- negative pair: 두 피처의 연관도가 낮음 -> 멀어지도록 학습
  - positive pair: 두 피처의 연관도가 높음 -> 가까워지도록 학습
- ✓ 각 문제의 첫번째 코드를 기준으로 BM25 알고리즘 적용  
1번 코드와 연관성이 얼마나 있는지, 얼마나 비슷한지에 따라 scoring 된 후 ranking 매겨짐  
해당 ranking을 가지고 다른 코드들과의 pair 구성 및 labeling 진행

## 데이터 전처리 결과

	code1	code2	similar
0	N = int(input())\nn = list(map(int,input().spl...	from time import time\nfrom random import rand...	0
1	n=int(input())\ns=[input() for i in range(n)]\...	N = int(input())\nans_ac = 0\nans_wa = 0\nans_...	1
2	import math\ndef koch(start, end, n, r):\n\tif...	import math\ndef triangle(co1, co2):\n\tans = ...	1
3	s = input()\nl = len(s)\nlst = [0] * (l+1)\nfo...	N, K, C = map(int, input().split())\nS = input...	0
4	def bubbleSort(a, n):\n\tflag = True\n\tcount ...	def pnt(s):\n for i in xrange(len(s)):\n prin...	1
...	...	...	...
5133762	import numpy as np\nfrom numpy.fft import rfft...	import numpy as np\nimport sys\ninput = sys.st...	1
5133763	N, X, M = map(int, input().split())\nl = [-1] ...	N, X, M = map(int, input().split())\npt = [[0]...	1
5133764	import sys\ninput= lambda: sys.stdin.readline(...	n,k=map(int,input().split())\nd=0\nfor i in ra...	0
5133765	import sys\nn = int(input())\nA = [int(e)for e...	n,m = map(int,input().split())\nc = list(map(i...	0
5133766	n, m = map(int,input().split())\nlst = [-1 for...	from sys import setrecursionlimit\nsetrecursio...	1

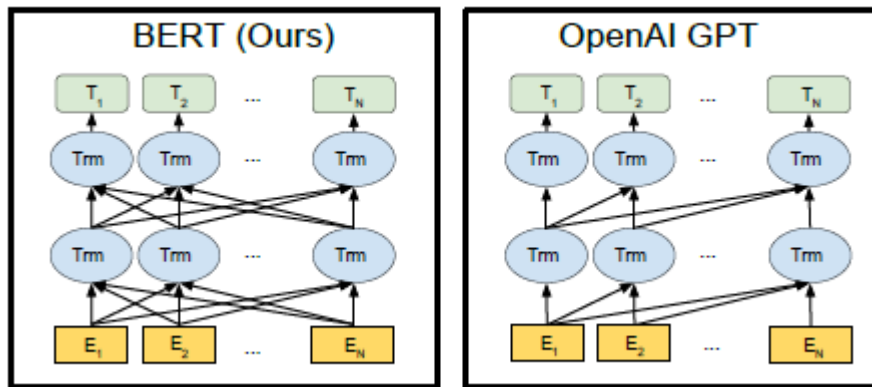
5133767 rows x 3 columns

추가로 5,133,767개의 데이터 추가 생성

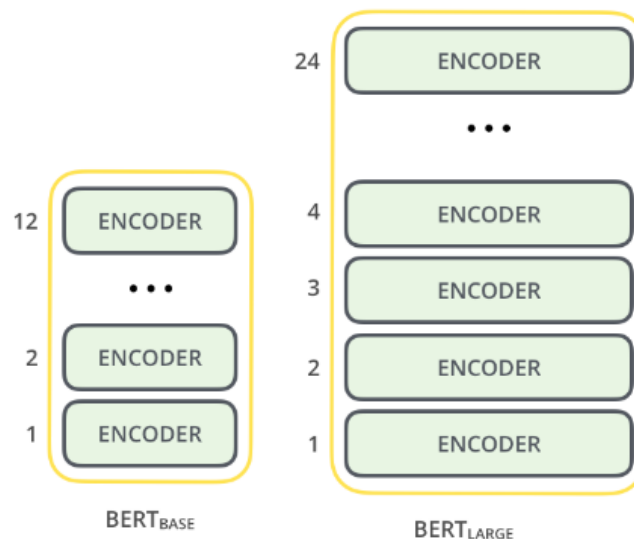


## BERT(Bidirectional Encoder Representations from Transformers)

- 다양한 task에 사용할 수 있는 pre-trained LM
- GPT는 한쪽 방향에 대해서만 고려하는 구조(left-to-right)로 제한적인 모델
- 양방향을 고려한 fine-tuning based approach!
- Transformer encoder를 12/24개 쌓아서 만든 구조



BERT vs GPT



BERT 구조

## BERT

다양한 task에 사용하려면 ?

## (1) 학습

## ✓ Masked LM

- 80% of the time: Replace the word with the [MASK] token, e.g., my dog is hairy → my dog is [MASK]
- 10% of the time: Replace the word with a random word, e.g., my dog is hairy → my dog is apple
- 10% of the time: Keep the word unchanged, e.g., my dog is hairy → my dog is hairy. The purpose of this is to bias the representation towards the actual observed word.

- 무작위로 15%의 단어를 선택해서 masking
- Masked word를 제대로 예측하도록 학습
- fine-tuning 과정에서도 좋은 성능을 위해, 80%는 masking, 10%는 무작위로 다른 단어로, 10%는 바꾸지 않고 학습

## ✓ Next Sentence Prediction(NSP)

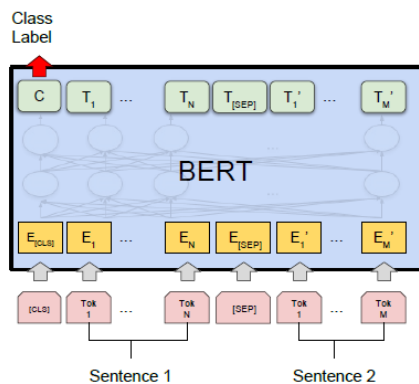
```
Input = [CLS] the man went to [MASK] store [SEP]
        he bought a gallon [MASK] milk [SEP]
Label = IsNext

Input = [CLS] the man [MASK] to the store [SEP]
        penguin [MASK] are flight ##less birds [SEP]
Label = NotNext
```

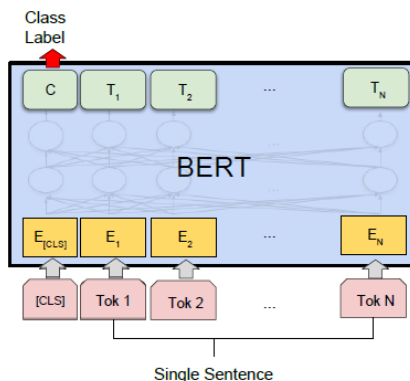
- Question Answering(QA), Natural Language Inference(NLI)와 같은 task들을 위한 학습 방법
- 문장의 짝을 지을 때, 50%는 실제로 다음 문장인 것, 50%는 무작위로 추출된 문장을 선택

## BERT

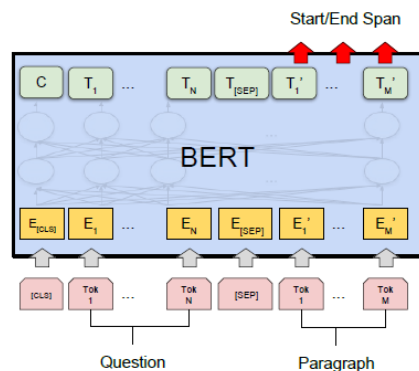
## Fine-tuning



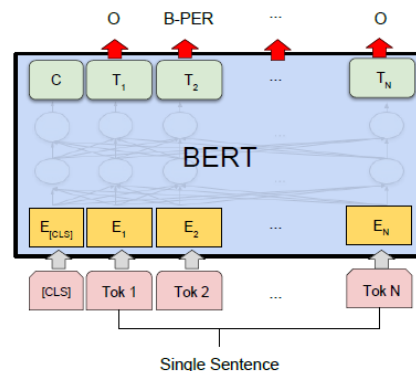
(a) Sentence Pair Classification Tasks:  
MNLI, QQP, QNLI, STS-B, MRPC,  
RTE, SWAG



(b) Single Sentence Classification Tasks:  
SST-2, CoLA



(c) Question Answering Tasks:  
SQuAD v1.1



(d) Single Sentence Tagging Tasks:  
CoNLL-2003 NER

## RoBERTa (Robustly optimized BERT approach)

BERT의 성능을 높인 기술 중 하나로 BERT의 parameter 및 training 방법의 변화를 통해 성능을 향상시킨 모델

### (1) Simple modifications

- Static Masking을 Dynamic Masking으로 바꿈
- NSP(Next Sentence Prediction) objective 제거
- 짧은 sequence를 배제하고 더 긴 sequence로 학습
- 더 많은 데이터를 사용하여 더 오래, 더 큰 batch로 학습

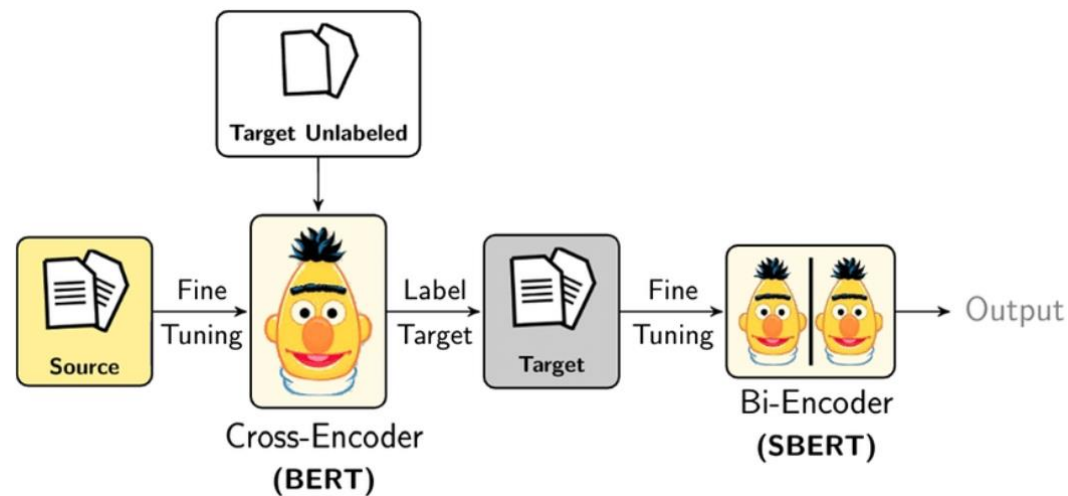
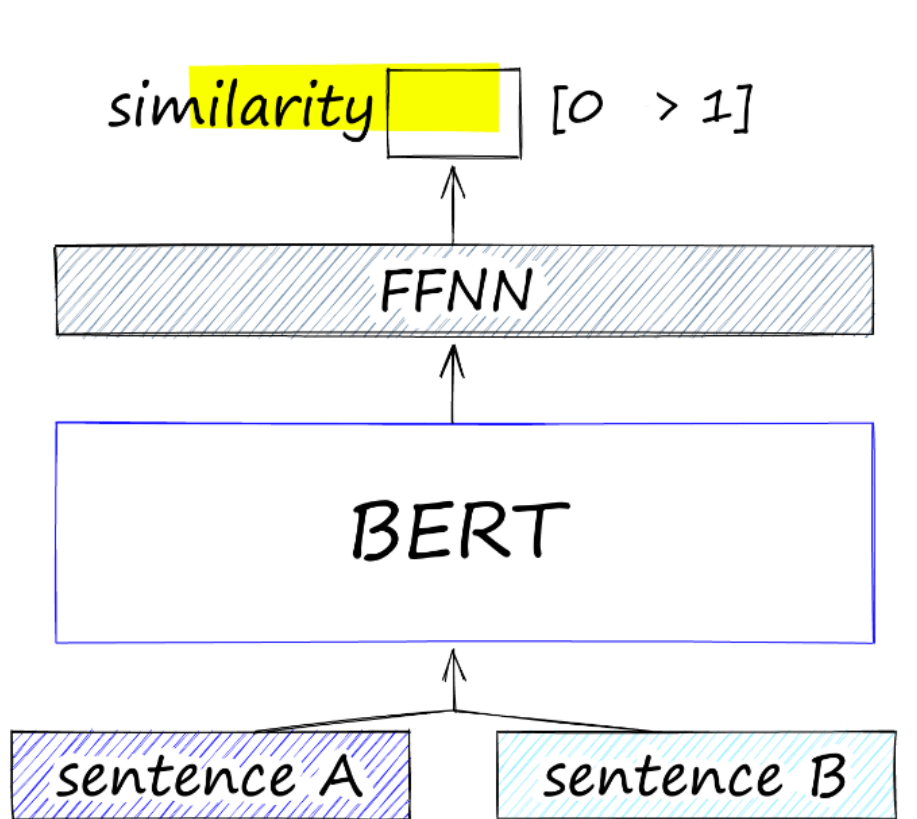
## RoBERTa

BERT의 성능을 높인 기술 중 하나로 BERT의 파라미터 및 트레이닝 방법의 변화를 통해 성능을 향상시킨 모델

### (2) 세부 내용

- **Static vs Dynamic Masking**  
: original BERT와 다르게 매 epoch마다 새로운 masking을 시키는 task
- **Model Input Format and NSP**  
: 기존 BERT에는 두 문장이 이어졌는지 판단하는 pretraining 과정인 NSP(next sentence prediction)가 존재  
→ NSP가 없을 때 성능이 비슷하거나 조금 향상
- **Training with large batches**  
: 아주 큰 mini-batch로 학습하면 적절한 learning-rate을 사용할 때  
optimization speed와 end-task performance 모두 향상

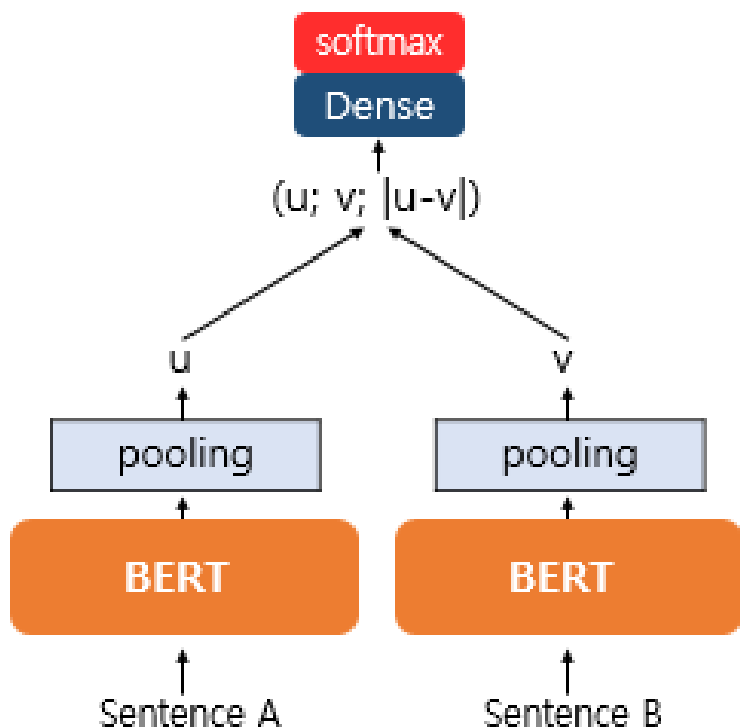
## SBERT(Sentence-BERT)



- SBERT는 BERT의 문장 임베딩을 응용하여 Finetuning 시킨 모델
- BERT의 문장 임베딩 성능을 우수하게 개선시킨 모델
- 문장 간 관계 파악에 용이

## SBERT

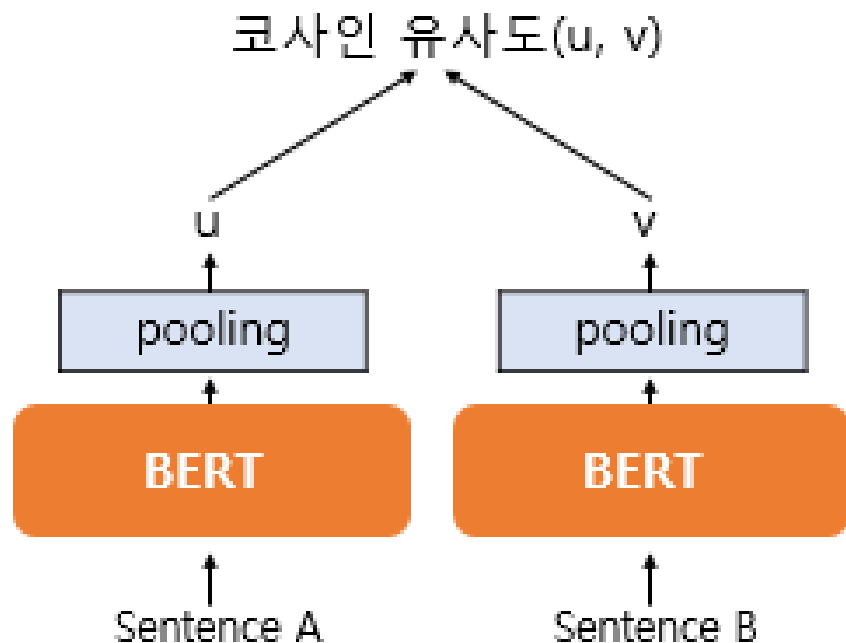
## 1) 문장 쌍 classification



- NLI 문제를 푸는 것에 집중하여 fine tuning
- 두 개의 문장이 주어졌을 때, 수반 관계인지, 모순 관계인지, 중립 관계인지를 맞추는 muticlass classification task

## SBERT

## 2) 문장 쌍 regression task





- 대표적으로 STS(Semantic Textual Similarity) 가 있으며, 이는 두 개의 문장으로부터 의미적 유사성을 구하는 문제
- 유사도(0~5)가 높을 수록 5에 가까운 label을 가짐



## 실험구성

사용 데이터의 종류와 모델의 종류에 따라 조합하여 실험 진행

Model 	Data 
<ul style="list-style-type: none"><li>- BERT</li><li>- SBERT</li><li>- RoBERTa</li></ul>	<ul style="list-style-type: none"><li>- Sample Data</li><li>- 생성 Pair Data<ul style="list-style-type: none"><li>- 2/3 Random</li><li>- Early Stopping</li></ul></li></ul>

추가로 생성한 Pair Data는 개수가 너무 많아 2가지 방식 적용

- 2/3만큼 random하게 추출하여 사용
- 모델 fitting 과정에서 Early stopping 적용

## 결과

	Sample Data	2/3 Random Data	Early stopping
BERT	0.7916	-	-
SBERT	0.9073	-	0.6805
RoBERTa	0.7801	0.8919	-

Metric: accuracy

## 결과

	Sample Data	2/3 Random Data	Early stopping
BERT	0.7916	-	-
SBERT	0.9073	-	0.6805
RoBERTa	0.7801	0.78919	-

678488

submission\_06051544.csv

[edit](#)

2022-06-05 15:45:35

0.9073641254



SBERT와 Sample Data 조합에서 가장 좋은 성능을 보임

Metric: accuracy

## 한계점

- 데이터 수를 늘렸더라도 너무 정체되지 않은 데이터일 경우 오히려 성능 저하
- 다양한 tokenizer와 전처리 방식을 통해 성능 향상을 이룰 수 있을 것이라 기대
- 알고리즘에 기반한 pair data를 생성하여 모델링하였으나 성능을 더 많이 향상시키지 못한 것에 대한 아쉬움
- SBERT 두 문장의 유사도를 기반으로 학습시킨 모델이기에 대회 task에서 높은 성능을 달성했을 것



**감사합니다.**

