Guidelines

특허분야 사전학습 언어모델

(KorPatBERT)

사용자 매뉴얼

제·개정 이력현황

개정버전	개정내용	작성자	승인자	제·개정일자
V1.0	최초 작성	박진우	노한성	2021.09.01

CONTENTS

목 차

제1장 개요

1. 특허분야 사전학습 모델 개요

1.1 KorPatBERT 개요

2018.11월 공개된 Google BERT Base 모델의 아키텍쳐를 기반으로, 한국어 특허문헌 코퍼스를 사전 학습한 언어모델로, 다양한 특허 자연어처리 태스크에 활용이 가능한 모델이다.

1.2 학습데이터

약 406만개의 특허공보의 내용 중 서지사항을 제외한 자연어로 구성된 모든 구성요소를 정제 및 전처리 후 학습하였으며, 학습데이터의 레이아웃은 Google에서 공개한 대로 한 라인에 한 문장씩 & 문단이 바뀌는 경우 하나의 빈 라인을 삽입하는 형태를 그대로 이용하였다.

[KorPatBERT 학습데이터 예시]

합성세제액포

제 1 도는 본 고안의 일부를 분리한 전체 사시도 제 2 도는 두겹으로 열중착된 세제액포를 한장으로 등분한 ㅌ

본 고안은 주로 일회용 합성세제액을 집어넣어 밀봉하는 세제액포의 내부를 원호상으로 열중착하되 세제액이
종래의 세제액포는 두장의 합성수지 필름을 중합하여 그 바깥둘레를 4각 또는 구형으로 열중착하였기 때문에
이와 같은 폐단을 시정하기 위하여 안출한 본원은, 세제액포의 내벽을 세제액이 배출하는 절단부 쪽으로 협소
구형으로 형성된 세제액포의 바깥둘레를 열중착하되, 절단부쪽으로 향하여 벽선을 원호상으로 형성함과 동시
이와같이 구성된 본 고안은, 세제액포의 벽선이 원호상으로 형성되어 있을뿐만 아니라, 내측모퉁이에 있어서도

세제액포의 바깥둘레를 열중착하되, 절단부쪽으로 향하여 벽선을 원호상으로 형성함과 동시에 내측 모퉁이도

회전 가능한 헤드를 구비한 칫솔 {TOOTHBRUSH HAVING ROTATABLE HEAD}

본 발명은 회전 가능한 헤드를 구비한 칫솔에 관한 것으로, 더욱 상세하게는, 칫솔 헤드가 상하 방향에서 좌우
도 1은 종래의 일반적인 칫솔을 도시한 도면이다.
도 1에 도시된 바와 같이, 종래의 일반적인 칫솔은 앞니에서부터 어금니 방향으로 또는 어금니에서 앞니 방향
하지만, 일반적으로 칫솔질은 잇몸 측에서부터 시작하여 치아 단부 측으로 상하 운동을 하는 칫솔질 방법이 ㅎ
종래의 일반적인 칫솔로 이와 같은 상하 운동을 하기 위해서는 칫솔의 손잡이를 잇몸 측에서부터 치아 단부 ㅎ
한편, 전동식으로 칫솔의 헤드 부분을 360 도 회전시키는 전동 칫솔이 제안되어 널리 판매되고 있다.
하지만, 이러한 전동식 회전 칫솔은 제조 단가가 높을 뿐만 아니라 원 형상의 칫솔모가 회전할 뿐이어서, 잇몸
한편, 한국 공개특허공보 10-2006-0020065 호에는 힌지를 구비한 칫솔이 개시되어 있다.
도 2는 이러한 종래 기술에 따라 칫솔대가 힌지를 중심으로 회전하는 동작을 도시한 도면이다.
이러한 종래기술에 따른 칫솔은 칫솔 헤드가 힌지를 중심으로 자유롭게 360 도 회전함으로써, 칫솔질시에 손의
하지만, 상기한 바와 같은 칫솔은 칫솔 헤드가 칫솔대에 대해 고정되지 않고 자유롭게 회전 운동을 하기 때문
또한, 칫솔 헤드가 360 도 회전함으로 인해서 좌우 방향의 칫솔질과 상하 방향의 칫솔질이 모두 불편하여 널리

[KorPatBERT 학습데이터 통계]

섹션	데이터 수	섹션	데이터 수
A(생활 필수품)	564,606	F(기계공학; 조명; 가열; 무기;)	367,676
B(처리조작; 수송)	690,645	G(물리)	757,660
C(화학; 야금)	416,332	H(전기)	1,013,023
D(섬유; 종이)	69,896	Y(새로운 기술)	664
E(고정 구조물)	185,017	-	-
데이터 수		**4,065,519건**	
문장 수		**460,448,189건**	
형태소 개수		**24,949,133,115건**	
코퍼스 크기		**117 GB**	

1.3 학습 파라미터

Google BERT Base 모델의 12개 인코더 및 사전학습 프로그램을 그대로 활용하였으며, 모델학습에 사용된 파라미터는 아래와 같다.

・파라미터 설정 파일명 : korpat_bert_config.json

[KorPatBERT Parameters]

파라미터	값	파라미터	값
attention_probs_dropout_prob	0.1	num_hidden_layers	12
directionality	"bidi"	pooler_fc_size	768
hidden_act	"gelu"	pooler_num_attention_heads	12
hidden_dropout_prob	0.1	pooler_num_fc_layers	3
hidden_size	768	pooler_size_per_head	128
initializer_range	0.02	pooler_type	"first_token_transform"
intermediate_size	3072	type_vocab_size	2
max_position_embeddings	512	vocab_size	21400
num_attention_heads	12	-	-

1.4 사용 토크나이저 (KorPat Tokenizer)

KorPatBERT 사전학습을 위하여, 언어모델 학습에 사용된 특허공보 406만건을 대상으로, Soynlp 라이브러리를 이용하여 약 660만개의 명사 및 복합명사를 추출하였으며, 이를 한국어 형태소분석기 Mecab-ko 의 사용자 사전에 추가 후 Google SentencePiece를 통하여 Subword로 분할하는 방식의 토크나이저를 사용하였으며, 이를 KorPat Tokenizer라 한다. 관련 정보는 아래와 같다.

- Mecab-ko 특허 사용자 사전파일명 : pat_all_mecab_dic.csv (6,663,693개 용어)
- SentencePiece 사전파일명 : korpat_vocab.txt (21,400개 토큰)
- SentencePiece 스페셜 토큰 : [PAD], [UNK], [CLS], [SEP], [MASK]
- KorPat Tokenizer 파일명 : korpat_tokenizer.py

1.5 언어모델 파일

KorPartBERT 언어모델은 Google에서 공개한 사전학습 프로그램을 그대로 활용하였으며, 해당 프로그램은 Tensorflow 프레임워크 기반으로 작성되어 있으므로, 학습완료된 언어모델 파일은 Tensorflow Checkpoint 파일 형태로 되어 있다.

- 언어모델 파일명 : model.ckpt-381250.meta
  model.ckpt-381250.index
  model.ckpt-381250.data-00000-of-00001

1.6 사전학습 환경

KorPartBERT 언어모델의 사전학습은 Google BERT 논문에서 제안된 방식대로 128 Sequence의 학습데이터와 512 Sequence의 학습데이터를 연이어서 학습하였으며, 그 환경은 아래와 같다.

- 128 Sequence 2,300,000 Step 학습 + 512 Sequence 750,000 Step 학습
- 특허공보 117 GB 코퍼스의 4억 6천만 문장 데이터 학습
- NVIDIA V100 32GB GPU 8개 및 분산학습 라이브러리 Horovod를 이용하여 학습
- NVIDIA AMP (Automated Mixed Precision) 방식을 활용하여, 메모리 최적화

제2장 사용안내

2. 언어모델 사용 안내

2.1 사전설치 요구사항

KorPatBERT 언어모델 및 토크나이저를 사용하기 위해서는 아래와 같은 환경 및 라이브러리들이 사전에 설치 요구되어진다.

※ Mecab 설치 후 특허사용자 사전은 필수로 추가 필요 (pat_all_mecab_dic.csv 파일)

[사전 설치 요구사항 목록]

프로그램명	버전	설치 안내 경로	필수
python	3.6 이상	https://www.python.org/	Y
anaconda	4.6.8 이상	https://www.anaconda.com/	N
tensorflow	2.2.0 이상	https://www.tensorflow.org/install/pip?hl=ko	Y
sentencepiece	0.1.96 이상	https://github.com/google/sentencepiece	N
mecab-ko	0.996-ko-0.0.2	https://bitbucket.org/eunjeon/mecab-ko-dic/src/master/	Y
mecab-ko-dic	2.1.1		Y
mecab-python	0.996-ko-0.9.2		Y
python-mecab-ko	1.0.11 이상	https://pypi.org/project/python-mecab-ko/	Y
keras	2.4.3 이상	https://github.com/keras-team/keras	N
bert_for_tf2	0.14.4 이상	https://github.com/kpe/bert-for-tf2	N
tqdm	4.59.0 이상	https://github.com/tqdm/tqdm	N
soynlp	0.0.493 이상	https://github.com/lovit/soynlp	N

2.2 KorPat Tokenizer 사용 예시

KorPat Tokenizer를 사용하기 위해서는 2.1에서 기술된 것처럼, Mecab-ko 프로그램이 필수적으로 설치 요구되며, 제공된 특허사용자사전이 Mecab에 추가되어야 한다. 자세한 사용 안내는 아래에 작성된 테스트 샘플 프로그램을 참고하면 된다.

* Tokenizer 파일명 : korpat_tokenizer.py
* Tokenizer 테스트 샘플 : test_tokenize.py

[test_tokenize.py 세부내용]

```
# coding=utf-8
from korpat_tokenizer import Tokenizer

# 메인 프로그램 시작
if __name__ == '__main__':

    # 토크나이저 선언
    # (vocab_path=Vocabulary 파일 경로, cased=한글->True, 영문-> False)
    tokenizer = Tokenizer(vocab_path="./korpat_vocab.txt", cased=True)

    # 테스트 샘플 문장
    example = "본 고안은 주로 일회용 합성세제액을 집어넣어 밀봉하는 세제액포의 내부를 원호상으로
              열중착하되 세제액이 배출되는 절단부 쪽으로 내벽을 협소하게 형성하여서 내부에
              들어있는 세제액을 잘짜질 수 있도록 하는 합성세제 액포에 관한 것이다."

    # 샘플 토크나이즈
    tokens = tokenizer.tokenize(example)

    # 샘플 인코딩 (max_len=토큰 최대 길이)
    ids, _ = tokenizer.encode(example, max_len=256)

    # 샘플 디코딩
    decoded_tokens = tokenizer.decode(ids)

    # 결과 출력
    print("Length of Token dictionary ===>", len(tokenizer._token_dict.keys()))
    print("Input example ===>", example)
    print("Tokenized example ===>", tokens)
    print("Converted example to IDs ===>", ids)
    print("Converted IDs to example ===>", decoded_tokens)
```

[test_tokenize.py 실행결과]

Length of Token dictionary ===> 21400

Input example ===> 본 고안은 주로 일회용 합성세제액을 집어넣어 밀봉하는 세제액포의 내부를 원호상으로 열중착하되 세제액이 배출되는 절단부 쪽으로 내벽을 협소하게 형성하여서 내부에 들어있는 세제액을 잘짜질 수 있도록 하는 합성세제 액포에 관한 것이다.

Tokenized example ===> ['[CLS]', '본', '고안', '은', '주로', '일회용', '합성', '##세', '##제', '##액', '을', '집', '##어넣', '어', '밀봉', '하', '는', '세제', '##액', '##포', '의', '내부', '를', '원호', '상', '으로', '열', '##중', '착하', '되', '세제', '##액', '이', '배출', '되', '는', '절단부', '쪽', '으로', '내벽', '을', '협소', '하', '게', '형성', '하', '여서', '내부', '에', '들', '어', '있', '는', '세제', '##액', '을', '잘', '짜', '질', '수', '있', '도록', '하', '는', '합성', '##세', '##제', '액', '##포', '에', '관한', '것', '이', '다', '.', '[SEP]']

Converted example to IDs ===> [5, 58, 554, 32, 2716, 6554, 817, 20418, 20308, 20514, 15, 732, 15572, 39, 1634, 12, 11, 5934, 20514, 20367, 9, 315, 16, 5922, 17, 33, 279, 20399, 16971, 26, 5934, 20514, 13, 674, 26, 11, 10132, 1686, 33, 3781, 15, 11950, 12, 64, 87, 12, 3958, 315, 10, 51, 39, 25, 11, 5934, 20514, 15, 1803, 12889, 399, 24, 25, 118, 12, 11, 817, 20418, 20308, 299, 20367, 10, 439, 56, 13, 18, 14, 6, 0]

Converted IDs to example ===> ['본', '고안', '은', '주로', '일회용', '합성', '##세', '##제', '##액', '을', '집', '##어넣', '어', '밀봉', '하', '는', '세제', '##액', '##포', '의', '내부', '를', '원호', '상', '으로', '열', '##중', '착하', '되', '세제', '##액', '이', '배출', '되', '는', '절단부', '쪽', '으로', '내벽', '을', '협소', '하', '게', '형성', '하', '여서', '내부', '에', '들', '어', '있', '는', '세제', '##액', '을', '잘', '짜', '질', '수', '있', '도록', '하', '는', '합성', '##세', '##제', '액', '##포', '에', '관한', '것', '이', '다', '.']

2.3 KorPatBERT 언어모델 사용 예시

KorPatBERT는 Google에서 공개한 Tensorflow 기반의 사전학습 프로그램으로 학습되어졌으며, 모델 파일 또한 Tensorflow 체크포인트 파일로 작성되어졌다. 해당 BERT 언어모델 파일을 활용하는 방법에 대해서는 Google 공식 Github를 비롯하여, 많은 방법들이 존재하며, 온라인상에서 검색을 통해 쉽게 찾아볼 수 있다.

본 매뉴얼에서는 파이썬 딥러닝 프레임워크 중 가장 활발히 사용되어지고 있는 Keras 프레임워크와 제공 되어진 KorPatBERT 언어모델을 활용하여 간단한 분류문제를 해결하는 샘플 프로그램을 예시로 들었으며, 자세한 사용 안내는 아래에 작성된 분류 샘플 프로그램 파일 내 주석을 참고하면 된다.

※ 샘플 프로그램 구동을 위해선, Keras 및 bert_for_tf2가 필수 설치되어 있어야 함

* 분류 샘플 프로그램 파일명 : test_lm.py
* 분류 샘플 데이터셋 파일명 : lm_test_data.tsv

[분류 샘플 데이터셋 구성]

컬럼	label	sentence
내용	특허분류코드 3종 (A, B, C)	특허 자연어 문장

[test_lm.py 세부내용]

```
#-*- coding: utf-8 -*-
# 프로그램 목적 : KorPatBERT 와 Keras를 활용하여, 3개의 클래스로 문장을 분류하는 테스트 프로그램
# 사용되어진 데이터 셋은 label, sentence 두개의 컬럼으로 구성되어 있으며,
# 클래스별 300개씩 총 900개의 데이터로 이루어져 있다.
# label은 특허 CPC코드 섹션 "A", "B", "C" 3개의 클래스로 구성되어 있고,
# sentence 는 특허기술에 대한 문장으로 이루어져 있다.

# 필요 라이브러리 임포트
import tensorflow as tf
import bert
import os
import mecab
import pandas as pd
import numpy as np
from tensorflow import keras
from tensorflow.keras.layers import Dense
from tqdm import tqdm
from korpat_tokenizer import Tokenizer
```

```
# 필요 환경변수 설정

os.environ['TF_KERAS'] = '1'      # Keras Tensorflow 설정
config_path = "./pretrained/korpat_bert_config.json" # KorpatBert Config 파일 경로
vocab_path = "./pretrained/korpat_vocab.txt"          # KorpatTokenizer Vocabulary 파일 경로
checkpoint_path = "./pretrained/model.ckpt-381250"  # KorpatBert 모델파일 경로
pretreind_model_dir = "./pretrained/"                 # KorpatBert 모델 디렉토리 경로
dataset_path = "./lm_test_data.tsv"                   # 사용할 데이터셋 경로
save_model_path = "./korpat_bert_test_model.h5"       # 학습완료 모델 저장 경로

MAX_SEQ_LEN = 256    # 학습 최대 토큰 갯수
BATCH_SIZE = 8       # 학습 배치 사이즈
EPOCHS = 5           # 학습 에폭
LR = 0.00003         # 학습률



# 분리 비율에 따른 데이터셋 분리 함수
# 입력 : dataset(분리 대상 데이터셋), split_val(분리 비율)
# 출력 : train_data(분리 데이터셋), dev_data(분리 데이터셋)

def dataset_split(dataset, split_val):
    lengths = int(len(dataset) * split_val)
    train_data = dataset[:lengths]
    dev_data = dataset[lengths:]
    return train_data, dev_data



# 데이터 셋 로드 함수
# 입력 : dataset_url (전체 데이터셋 경로)
# 출력 : train_data(학습데이터) dev_data(검증데이터), test_data(평가데이터)

def dataset_load(dataset_url):
    all_data = pd.read_csv(dataset_url, sep='₩t')
    all_data = all_data.sample(frac=1).reset_index(drop=True)
    train_data, test_data = dataset_split(dataset=all_data, split_val=0.9)
    train_data, dev_data = dataset_split(dataset=train_data, split_val=0.9)
    return train_data, dev_data, test_data
```

```
# 학습을 위한 데이터 셋 전처리 함수
# 입력 : dataset(전처리 대상 데이터셋)
# 출력 : tokens(토큰화 결과), x_data(입력데이터), y_data(정답데이터)

def preprocessing_dataset(dataset):
    tokens, indices, labels = [], [], []

    # 문장을 토큰화 및 인코딩 처리하고, 라벨을 One hot 벡터 변환으로 처리한다.
    for label, sentence in tqdm(zip(dataset['label'], dataset['sentence']), desc="전처리 진행중"):
        tokens.append(tokenizer.tokenize(sentence))
        ids, _ = tokenizer.encode(sentence, max_len=MAX_SEQ_LEN)
        indices.append(ids)
        if label == "A":
            labels.append([1, 0, 0])
        elif label == "B":
            labels.append([0, 1, 0])
        else:
            labels.append([0, 0, 1])
    x_data = np.array(indices)
    y_data = np.array(labels)
    return tokens, x_data, y_data



# 메인 프로그램 시작

if __name__ == '__main__':
    # KorPat Tokenier 선언
    tokenizer = Tokenizer(vocab_path=vocab_path, cased=True)

    # 필요 데이터셋 로드
    train_data, dev_data, test_data = dataset_load(dataset_path)

    # 모든 데이터셋 전처리
    print("===> 학습데이터 샘플 출력 및 전처리 시작 <===")
    print(train_data[:10])
    train_tokens, train_x, train_y = preprocessing_dataset(train_data)

    print("₩n₩n===> 검증데이터 샘플 출력 및 전처리 시작 <===")
    print(dev_data[:10])
    dev_tokens, dev_x, dev_y = preprocessing_dataset(dev_data)

    print("₩n₩n===> 평가데이터 샘플 출력 및 전처리 시작 <===")
    print(test_data[:10])
    test_tokens, test_x, test_y = preprocessing_dataset(test_data)



    # 라벨의 크기 정의

    label_size = train_y.shape[1]
    print("===> 라벨 사이즈 : ", label_size)
```

```
# 분류 모델 네트워크 정의, ※ MirroredStrategy (분산처리 포함)
mirrored_strategy = tf.distribute.MirroredStrategy()
with mirrored_strategy.scope():

    # 입력 레이어 생성
    input_ids  = keras.layers.Input(shape=(MAX_SEQ_LEN,), dtype='int32')

    # BERT 언어모델 레이어 생성
    bert_params = bert.params_from_pretrained_ckpt(pretreind_model_dir)
    l_bert = bert.BertModelLayer.from_params(bert_params, name="bert")

    # 입력레이어와 BERT 레이어 연결
    bert_output = l_bert(input_ids)
    print("bert shape", bert_output.shape)

    # BERT 출력 중 Context 정보가 담긴, [CLS] 토큰 정보를 추출
    cls_out = keras.layers.Lambda(lambda seq: seq[:, 0, :])(bert_output)

    # 최종 분류를 위한 클래스 개수로 구성된 Dense 레이어를 연결
    outputs = Dense(units=label_size, activation='softmax')(cls_out)

    # 모델 빌딩
    model = keras.Model(inputs=input_ids, outputs=outputs)
    model.build(input_shape=(None, MAX_SEQ_LEN))

    # BERT 언어모델의 초기 가중치 로드
    bert.load_stock_weights(l_bert, checkpoint_path)

    # 모델 컴파일 과정
    model.compile(
        optimizer=keras.optimizers.Adam(learning_rate=LR),
        loss='categorical_crossentropy',
        metrics=['accuracy'],
    )

# 모델 요약 출력
model.summary()

# 모델 학습 시작
history = model.fit(
    train_x,
    train_y,
    epochs=EPOCHS,
    batch_size=BATCH_SIZE,
    validation_data=(dev_x, dev_y)
)

# 모델 저장 및 테스트 데이터를 이용한 평가
model.save(save_model_path)
eval_result = model.evaluate(test_x, test_y)

print('₩n===> 평가 결과 출력')
print("Accuracy : %.4f" % (eval_result[1]))
```

[test_lm.py 실행결과]

```
---------------------------------------- 일부 생략 ----------------------------------------

Model: "model"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 256)]             0
_________________________________________________________________
bert (BertModelLayer)        (None, 256, 768)          101884416
_________________________________________________________________
lambda (Lambda)              (None, 768)               0
_________________________________________________________________
dense (Dense)                (None, 3)                 2307
=================================================================
Total params: 101,886,723
Trainable params: 101,886,723
Non-trainable params: 0
_________________________________________________________________


Epoch 1/5
2021-09-01 13:30:56.395359: I tensorflow/stream_executor/platform/default/dso_loader.cc:44]
Successfully opened dynamic library libcublas.so.10
92/92 [==============================] - 59s 640ms/step - accuracy: 0.5350 - loss:
0.9620 - val_accuracy: 0.8642 - val_loss: 0.3982
Epoch 2/5
92/92 [==============================] - 49s 535ms/step - accuracy: 0.9246 - loss:
0.1948 - val_accuracy: 0.9259 - val_loss: 0.2275
Epoch 3/5
92/92 [==============================] - 49s 534ms/step - accuracy: 0.9931 - loss:
0.0401 - val_accuracy: 0.8025 - val_loss: 0.6616
Epoch 4/5
92/92 [==============================] - 49s 535ms/step - accuracy: 0.9918 - loss:
0.0278 - val_accuracy: 0.9506 - val_loss: 0.2411
Epoch 5/5
92/92 [==============================] - 49s 537ms/step - accuracy: 1.0000 - loss:
0.0020 - val_accuracy: 0.9506 - val_loss: 0.2404

3/3 [==============================] - 1s 317ms/step - accuracy: 0.9778 - loss: 0.0583


===> 평가 결과 출력

Accuracy : 0.9778
```

제3장 제공내역

3. 제공 파일 목록

3.1 경로별 제공 파일 및 설명

경로	파일명	설명
/	pat_all_mecab_dic.csv	Mecab 특허사용자사전
	lm_test_data.tsv	분류 샘플 데이터 셋
	korpat_tokenizer.py	KorPat Tokenizer 프로그램
	test_tokenize.py	Tokenizer 사용 샘플
	test_tokenize.ipynb	Tokenizer 사용 샘플 (주피터)
	test_lm.py	언어모델 사용 샘플
	test_lm.ipynb	언어모델 사용 샘플 (주피터)
pretrained/	korpat_bert_config.json	KorPatBERT Config 파일
	korpat_vocab.txt	KorPatBERT Vocabulary 파일
	model.ckpt-381250.meta	KorPatBERT Model 파일
	model.ckpt-381250.index	
	model.ckpt-381250.data-00000-of-00001	

[붙임] 1. KorPatBERT 사용허가 협약서 샘플

- 특허분야 한국어 언어모델 KorPatBERT 사용허가 협약서 -

제1조. 본 사용 허가 협약에 동의하여 서명된 협약서를 한국특허정보원(이하 제공자)에 제출한 경우에만 본 한국어 KorPatBERT 언어모델(이하 언어모델)을 사용할 자격을 갖습니다.

제2조. 본 언어모델은 제공자의 서면 승인 없이 제3자에게 제공, 양도, 사용 허가 또는 기술 공개를 할 수 없습니다.

제3조. 협약자는 본 언어모델에 포함된 라이선스를 준수하고, 상업용으로 활용하고자 할 경우에는 활용정보를 제공자에게 제공해야 합니다.

* 활용정보 접수 메일 : 한국특허정보원 IP디지털혁신센터, ai_support@kipi.or.kr

제4조. 본 언어모델을 수정하거나 개선한 경우 그 수정본이나 개선 버전을 외부에 공개하기 전에 제공자에게 정보를 공유하고 협의 후 진행하여야 합니다.

제5조. 본 언어모델을 활용한 결과를 외부 발표할 때에는 본 언어모델 활용 사실을 밝혀야 합니다.

- 이 연구는 한국특허정보원의 한국어 사전학습 언어모델(KorPatBERT)을 활용함

제6조. 본 사용허가협약의 효력은 제공자의 명의가 변경되더라도 존속됩니다.

제7조. 본 언어모델의 라이선스를 위반 시 제공자가 요청할 경우 즉시 사용을 중단하여야 합니다.

제8조. 본 언어모델이 협약자로 인해 문제가 발생하였을 경우 협약자는 전적으로 민, 형사상의 모든 법적 책임을 부담해야 합니다.

제공자 : 한국특허정보원

협약자 : 소 속 : 직 위 :

전화번호 : 이 름 : (인)

이 메 일 : 협약일 : 년 월 일

[붙임] 2. KorPatBERT 라이센스 샘플

- 특허분야 한국어 언어모델 KorPatBERT 라이선스 -

1. 한국특허정보원 한국어 BERT 언어모델(이하 KorPatBERT)은 연구용으로 활용 시, Apache2.0 라이선스를 따르고 상업용으로 활용 시, 정보 제공 및 사전협의 의무를 가진다.

2. KorPatBERT 모델을 상업적으로 활용하고자 할 경우, 한국특허정보원에 아래 정보를 제공하고 사전협의를 통해 세부적인 사항을 검토·조율하여야 한다.

[상업적 활용 문의]

- *활용 주체* *(예시) AI서비스기업*
- *활용 서비스* *(예시) 문서 자동독해 서비스*
- *예상 서비스 기간* *(예시) 2021년 1월 1일 ~ 2021년 12월 31일*
- *담당자 및 연락처* *(예시) 홍길동 / 042-000-0000*

3. KorPatBERT 구축 노하우를 활용하여 지능형 IP정보서비스 개발 또는 특허분야 外 특정 도메인에 특화된 한국어 BERT 언어모델 개발을 원할 경우 기술이전, 기술제휴 등에 대해 상호 협의한 이후 문서화함을 원칙으로 한다.

[기술협력 문의]

- *협력 주체* *(예시) AI서비스기업*
- *협력 분야* *(예시) 문서 자동독해 서비스 고도화*
- *협력 기간* *(예시) 2022년 1월 1일 ~ 2022년 12월 31일*
- *담당자 및 연락처* *(예시) 홍길동 / 042-000-0000*

문의 메일	한국특허정보원 IP디지털혁신센터, ai_support@kipi.or.kr

---------- 아 래 ----------

Apache License

Version 2.0, January 2004

http://www.apache.org/licenses/

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

- 이 하 생 략-

특허분야 사전학습 언어모델(KorPatBERT) 사용자 매뉴얼

발행일 : 2021년 9월 1일

발행처 : 한국특허정보원

연락처 : 한국특허정보원 IP디지털혁신센터

(042-603-8075, ai_support@kipi.or.kr)