자연어기반 산업 분류 자동화 모델 구축

산업코드 KU분기

14기 오화진
15기 김지후
15기 김진수

1. Data : Preprocessing
2. Modeling
3. 챗봇 결과

1. Data : Preprocessing
2. Modeling
3. 분류 결과

01 **Project Topics**

공공분야의 자연어 데이터 : 민원 상담데이터 / 전국사업체 조사의 산업소개 데이터)

민원 챗봇

공공민원 데이터를 사용해
민원에 대한 질문과 답을 주고
받으면서 **대화**한다.

산업 분류 자동화

산업에 대한 설명(무엇을, 어떤 방법으로, 생산•제공하였는가)으로 **산업분류 코드**(소,중,대분류)를 예측한다.

02 Task1

민원 챗봇

공공민원 데이터를 사용해
민원에 대한 질문과 답을 주고
받으면서 **대화**한다.

날씨가 좀 풀린거 같아요

따뜻해졌죠

여권 만들려고 하는데
몇시까지 접수하나요

평일 일곱시까지
접수합니다.

Q	A
갑자기 물어봐서 당황했어	갑작스러웠나봐요.
갑자기 불편한 사이가 된 거 같아	관계의 변화가 왔나봅니다.
강렬한 첫인상 남겨야 하는데	처음 3초가 중요해요. 당신의 매력을 어필해보세요.
강아지 키우고 싶어	책임질 수 있을 때 키워 보세요.
강아지 키우고 싶은데 역시 안돼겠지	먼저 생활패턴을 살펴 보세요.
강아지 키울 수 있을까	먼저 생활패턴을 살펴 보세요.
강아지 키울까	책임질 수 있을 때 키워 보세요.
강원도 가서 살까?	아름다운 곳이죠.

Chatbot_data_for_Korean v1.0

https://github.com/songys/Chatbot_data :

일상 챗봇 학습용 문답
11,876개 데이터

AI HUB 한국어대화

공공민원 데이터 :

여권 민원 문답
403개 데이터

question	answer
네 여권발급 관련해서 문의드리려고 하는데요 지금 재발급하는 건데 뭐 비용이나 절차 ...	예 오실 때 본인 신분증과 최근 육개월 안에 촬영한 여권용 사진 두 매 지참해서 오..
인감증명 인감도장이런 건 필요없나요	아 필요없습니다
미성년자 고등학교 일학년 여권 발급 필요한 서류하고 좀 문의 할려구요	신청자 분께서 신분증 소지하시고 방문을 하셔야 되는데요. 방문을 하실 때 미성년자 ..
여권이 중국에 사는 사람인데요 여권이 만료됐는데 십일년으로 만료됐는데 새로 낼라 카...	오실 때 이제 선생님 그 본인 직접 방문하시고 신분증하고 사로 찍은 여권용 사진 두..
만드는데 필요한 서류가 어떤 거죠.	본인이 꼭 가셔야 되구요 그리고 여권용 사진 두 매랑 신분증
아 예 그 여권발급을 신청할라하면은요 준비물이 뭐가 필요합니까?	그렇다면은 신분증 하고 최근 육개월 이내 찍은 여권용 사진 두 장 이 두 가지만 가..
여권 재발급 받을려고 하는데요 그 구비 서류가 어떻게 되는가요	아 그러세요 쌤 그런 경우 준비 서류는요 지금 가지고 계신 그 구여권 가지고 오셔야..

Step1.

결측치 제거

: NULL 값을 삭제

Step2.

한글/영문 남기기

: 정규표현식을 사용해 한글과 영문만 남기기

Step3.

불용어 제거

: 대화 분류에 도움이 되지 않는 불용어 삭제

ex) '아', '음', '그면', '혹', '잠시만요', '어', '여보세요' ...

모델 선정(Transformer, KoGPT2)

1. Transformer

```
[ ]  model=transformer(num_layers=1,d_model=128,num_heads=4,dff=256,
                       input_vocab_size=VOCAB_SIZE-2, target_vocab_size=VOCAB_SIZE,ts_input=98,ts_target=104,rate=0.1)
                       #num_layers=2,3,4는 작동안함. 아마도 작으면 encoder_input에 영향을 많이 받도록함
                       #num_layers가 크면 encoder보다 decoder_input에 더 큰영향을 받아 첫 word=START_TOKEN으로 같으므로
                       #항상 같은 답변으로 나올 가능성이 많음. 실제로 그러함. embedding size는 작은 것이 우수함
     optimizer = tf.keras.optimizers.Adam()
     model.compile(optimizer=optimizer, loss=loss_function, metrics=[accuracy_function])
     model.fit(data_final,epochs=20)

     Epoch 1/20
     383/383 [==============================] - 387s 997ms/step - loss: 6.5388 - accuracy_function: 0.1850
     Epoch 2/20
     383/383 [==============================] - 371s 968ms/step - loss: 5.0934 - accuracy_function: 0.2617
     Epoch 3/20
     383/383 [==============================] - 370s 966ms/step - loss: 4.0208 - accuracy_function: 0.3534
     Epoch 4/20
     383/383 [==============================] - 368s 961ms/step - loss: 3.1638 - accuracy_function: 0.4369
     Epoch 5/20
     383/383 [==============================] - 369s 964ms/step - loss: 2.4903 - accuracy_function: 0.5226
     Epoch 6/20
     383/383 [==============================] - 369s 964ms/step - loss: 1.9403 - accuracy_function: 0.6067
     Epoch 7/20
     383/383 [==============================] - 367s 959ms/step - loss: 1.4910 - accuracy_function: 0.6782
     Epoch 8/20
     383/383 [==============================] - 369s 965ms/step - loss: 1.1524 - accuracy_function: 0.7385
     Epoch 9/20
     383/383 [==============================] - 373s 973ms/step - loss: 0.9083 - accuracy_function: 0.7837
     Epoch 10/20
```

인코더-디코더 구조를 Attention만으로 구현한 모델

2. KoGPT2

```
[ ]  2 for epoch in range(epochs):
     3   epoch_loss = 0
     4
     5   for batch in tqdm.tqdm_notebook(dataset, total=steps):
     6     with tf.GradientTape() as tape:
     7       result = model(batch, labels=batch)
     8       loss = result[0]
     9       batch_loss = tf.reduce_mean(loss)
    10
    11     grads = tape.gradient(batch_loss, model.trainable_variables)
    12     adam.apply_gradients(zip(grads, model.trainable_variables))
    13     epoch_loss += batch_loss / steps
    14
    15   print('[Epoch : {:>4}] cost = {:>.9}'.format(epoch + 1, epoch_loss))

     /usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:5: TqdmDeprecationWarning: This function will be removed i
     Please use `tqdm.notebook.tqdm` instead of `tqdm.tqdm_notebook`
       """
     100% 370/370 [03:22<00:00, 1.90it/s]
     [Epoch :    1] cost = 2.1270802
     100% 370/370 [02:34<00:00, 2.09it/s]
     [Epoch :    2] cost = 1.69889617
     100% 370/370 [02:33<00:00, 2.44it/s]
     [Epoch :    3] cost = 1.37640476
```

Transformer 체계에서 40GB 이상의 텍스트로 학습된 한국어 디코더만 사용하는 언어모델

Transformer

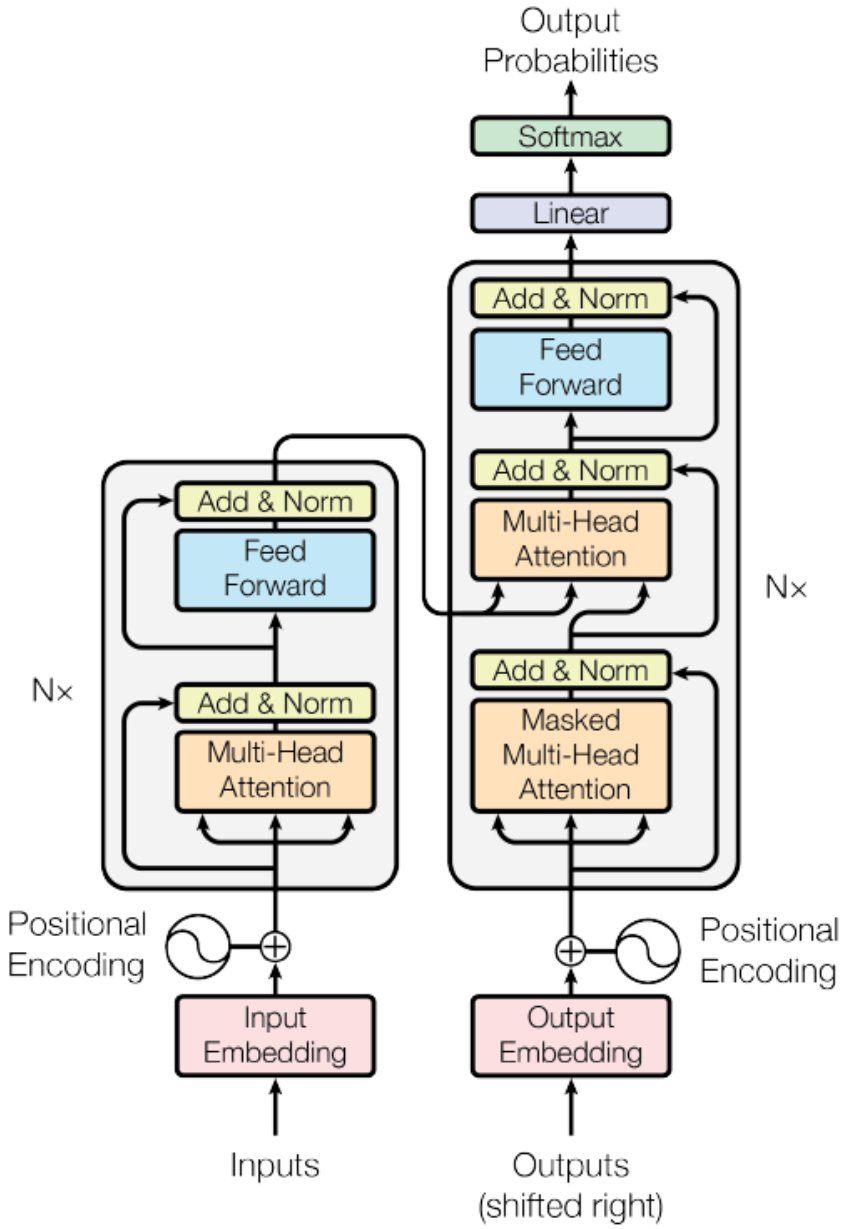
학습결과를 비교해 더 자연스러운 챗봇을 만든 **Transformer**가 최종 모델

주요 하이퍼파라미터

d_model = 128
nhead = 4
dff = 256
numlayers = 1
epochs = 30
runing_rate = 0.1

Activation
Relu

optimizer
Adam



: 일상 대화

```
[ ]  predict('영화 볼래?')

     '영화 추천 부탁드립니다.'

[ ]  predict('고민이 있어')

     '그게 무슨 고민일까요?'
```

: 민원 문답

```
predict('여권 발급 하는 데 며칠 걸리죠')

[[  93  712   44  115 2890 3381   33    0    0    0    0    0    0    0
     0    0    0    0    0    0]]
(1, 104, 7608)
[[7606 2033   14   23 1296 7033 7382 1342  892 1592 3194  305    0    0
     0    0    0    0    0    0    0    0    0    0    0    0    0    0
     0    0    0    0    0    0    0    0    0    0    0    0    0    0
     0    0    0    0    0    0    0    0    0    0    0    0    0    0
     0    0    0    0    0    0    0    0    0    0    0    0    0    0
     0    0    0    0    0    0    0    0    0    0    0    0    0    0
     0    0    0    0    0    0    0    0    0    0    0    0    0    0
     0    0    0    0    0    0]]
[[2033   14   23 1296 7033 7382 1342  892 1592 3194  305 7607  305 2033
   305  305  305  305  305  305  305  305  305   23   23   23  121  121
   121  121  121  121  121   23   23 1364 1364  305  305  305  254 6572
  6572 6572 6382 6382 6382 6382 6382 6382 2033 2033  305  305  305  305
  2033  305  305  305  305  305 2033 2033 2033 2033 2033  305 2033 2033
  2033 1364 1364 1364 1364 2033 1364 1364 1364 1364 1364 1342 1342 5770
  5770  305  305  305  305  305  661 2033 1364 2033 2033 2033 2033 2033
  2033 2033 2033  103  103 1364]]
(1, 104)
Input: 여권 발급 하는 데 며칠 걸리죠
Output: ['신청', '은 ', '그 ', '신청 ', '당일', ' ', '포함해서 ', '평일 ', '사일 ', '소요', '됩니다']
['신청', '은 ', '그 ', '신청 ', '당일', ' ', '포함해서 ', '평일 ', '사일 ', '소요', '됩니다']
```

여권 발급 하는데 며칠 걸리죠

신청은 그 신청 당일 포함해서 평일 사일 소요됩니다

산업 분류 자동화

산업에 대한 설명(무엇을, 어떤 방법으로, 생산•제공하였는가)으로 **산업분류 코드**(소,중,대분류)를 예측한다.

사무실에서

중개를 통해서

건축용 목재•골재

↓

대분류 : G
중분류 : 46
소분류 : 461

모델 개발용 자료

AI_id 번호	digit_1 대분류	digit_2 중분류	digit_3 소분류	text_obj 무엇을 가지고 (원재료, 영업장소 등)	text_mthd 어떤 방법으로 (주요 영업, 생산 활동)	text_deal 생산·제공하였는가 (최종 재화, 용역)
1	S	95	953	고객의 요청에 의해	에어컨 자재 등을 가지고 고객 현장에서	냉난방공조기기 등 가전 서비스 및 유지보수
2	Q	86	862	외과의원에서	외래 및 입원실 18병상, 수술실 등을 갖추고	일반외과 수술 및 치료 입원관리
...	...	...	...	...	...	...
100만	G	46	464	중국에서 제조하여 한국을 거쳐 미국으로 수출	중국에서 매트리스 외 가구류를 가지고	메모리폼 침대 외 가구류

산업 분류표

digit_1	digit_2	digit_3	digit_4	digit_5	type_5
A	1	11	111	1110	곡물 및 기타 식량작물 재배업
			112	1121	채소작물 재배업
				1122	화훼작물 재배업
				1123	종자 및 묘목 생산업
			113	1131	과실작물 재배업
				1132	음료용 및 향신용 작물 재배업

text_obj 무엇을 가지고 (원재료, 영업장소 등)	text_mthd 어떤 방법으로 (주요 영업, 생산 활동)	text_deal 생산·제공하였는가 (최종 재화, 용역)
고객의 요청에 의해	에어컨 자재 등을 가지고 고객 현장에서	냉난방공조기기 등 가전 서비스 및 유지보수
외과의원에서	외래 및 입원실 18병상, 수술실 등을 갖추고	일반외과 수술 및 치료 입원관리
…	…	…
중국에서 제조하여 한국을 거쳐 미국으로 수출	중국에서 매트리스 외 가구류를 가지고	메모리폼 침대 외 가구류

→

‘DATA’ 로 통합

불용어 제거

저빈도 단어

텍스트 병합

산업 분류표

digit_1	digit_2	digit_3	digit_4	digit_5	type_5
A	1	11	111	1110	곡물 및 기타 식량작물 재배업
			112	1121	채소작물 재배업
				1122	화훼작물 재배업
				1123	종자 및 묘목 생산업
			113	1131	과실작물 재배업
				1132	음료용 및 향신용 작물 재배업


1
명사 추출 +
keras 활용한 cnn

KoNLPy 의

Mecab vs OKT

```
import konlpy
from konlpy.tag import Okt
```

```
okt=Okt()
```

```
test_1=test[:].copy()
```

```
for i in range(len(test_1['text_obj'])):
  try:
    test_1['text_obj'][i]=okt.nouns(test_1['text_obj'][i])
  except AssertionError:
    pass
```

	AI_id	digit_1	digit_2	digit_3	text_obj	text_mthd	text_deal
0	id_0000001	S	95	952	['카', '센터']	자동차부분정비	타이어오일교환
1	id_0000002	G	47	472	['상점', '내']	일반인을 대상으로	채소.과일판매
2	id_0000003	G	46	467	['절단', '업체', '매']	공업용고무를가지고	합성고무도매
3	id_0000004	G	47	475	['영업', '점']	일반소비자에게	열쇠잠금장치
4	id_0000005	Q	87	872	['어린이집']	보호자의 위탁을 받	취학전아동보육
5	id_0000006	C	29	291	['철']	절삭.용접	카프라배관자재
6	id_0000007	I	56	561	['음식점']	접객시설을 갖추고	참치회(일본식)
7	id_0000008	C	10	107	['쌀', '가지']	가공하여	떡제조

805406	id_0805407	I	56	562	['커피점']		커피판매
805407	id_0805408	C	10	107	['떡', '가게']	접객시설없이제조ㅎ	떡 소매
805408	id_0805409	M	71	713	['사무실']	일반인을 대상으로	인터넷대행광고
805409	id_0805410	N	75	751	['사업']	고객의요청에의해	고용알선
805410	id_0805411	P	85	856	['학원']	학생을 대상으로	피아노교습
805411	id_0805412	G	47	473	['매장']	소매	핸드폰
805412	id_0805413	I	56	561	['음식점']	접객시설을 갖추고	설렁탕

토큰화 완료 데이터 :
총 80만여개의 데이터 명사만 추출

Step1.

명사들 요소 확인

의미와 빈도를 기준으로
불용어 자체 정의
Ex. '및', '위주', '해', '등'

Step2.

불용어 제거

Step3.

결측치 처리

```
AI_id            0
digit_1          0
digit_2          0
digit_3          0
text_obj     16677
text_mthd    43619
text_deal    67652
dtype: int64
```

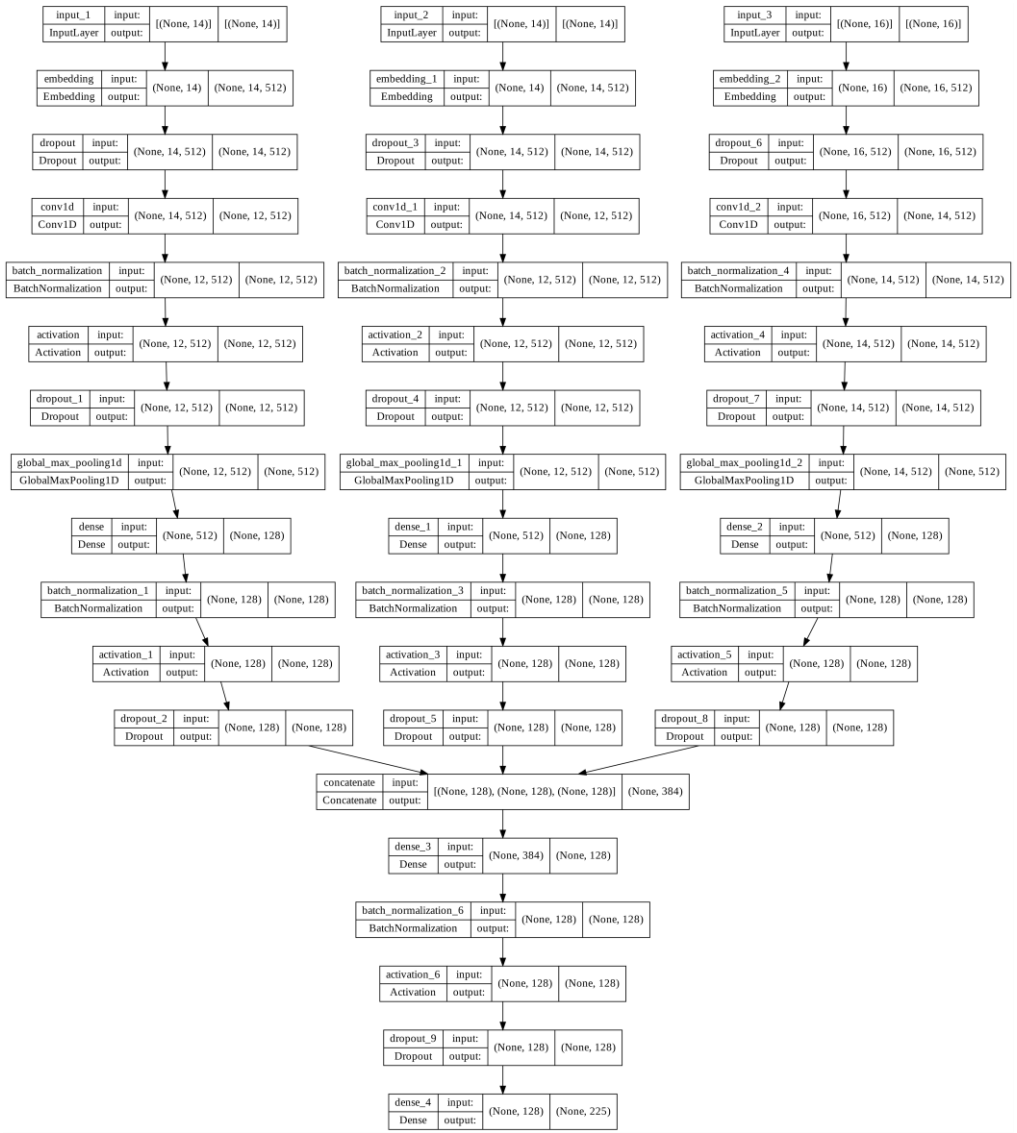
```
tokenizer=Tokenizer(oov_token = 'OOV')

tokenizer.fit_on_texts(list(noun_train['all_noun']))
word_vocab = tokenizer.word_index

vocab_size = len(word_vocab)+2
```

```
단어 집합(vocabulary)의 크기 : 30817
등장 빈도가 1번 이하인 희귀 단어의 수: 12377
단어 집합에서 희귀 단어의 비율: 40.162897102248756
전체 등장 빈도에서 희귀 단어 등장 빈도 비율: 0.2135263241884076
예상 단어집합 크기 : 18440
```

modeling


input_1 | input: | [(None, 14)] | [(None, 14)]
InputLayer | output:
embedding | input: | (None, 14) | (None, 14, 512)
Embedding | output:
dropout | input: | (None, 14, 512) | (None, 14, 512)
Dropout | output:
conv1d | input: | (None, 14, 512) | (None, 12, 512)
Conv1D | output:
batch_normalization | input: | (None, 12, 512) | (None, 12, 512)
BatchNormalization | output:
activation | input: | (None, 12, 512) | (None, 12, 512)
Activation | output:
dropout_1 | input: | (None, 12, 512) | (None, 12, 512)
Dropout | output:
global_max_pooling1d | input: | (None, 12, 512) | (None, 512)
GlobalMaxPooling1D | output:
dense | input: | (None, 512) | (None, 128)
Dense | output:
batch_normalization_1 | input: | (None, 128) | (None, 128)
BatchNormalization | output:
activation_1 | input: | (None, 128) | (None, 128)
Activation | output:
dropout_2 | input: | (None, 128) | (None, 128)
Dropout | output:
input_2 | input: | [(None, 14)] | [(None, 14)]
InputLayer | output:
embedding_1 | input: | (None, 14) | (None, 14, 512)
Embedding | output:
dropout_3 | input: | (None, 14, 512) | (None, 14, 512)
Dropout | output:
conv1d_1 | input: | (None, 14, 512) | (None, 12, 512)
Conv1D | output:
batch_normalization_2 | input: | (None, 12, 512) | (None, 12, 512)
BatchNormalization | output:
activation_2 | input: | (None, 12, 512) | (None, 12, 512)
Activation | output:
dropout_4 | input: | (None, 12, 512) | (None, 12, 512)
Dropout | output:
global_max_pooling1d_1 | input: | (None, 12, 512) | (None, 512)
GlobalMaxPooling1D | output:
dense_1 | input: | (None, 512) | (None, 128)
Dense | output:
batch_normalization_3 | input: | (None, 128) | (None, 128)
BatchNormalization | output:
activation_3 | input: | (None, 128) | (None, 128)
Activation | output:
dropout_5 | input: | (None, 128) | (None, 128)
Dropout | output:
input_3 | input: | [(None, 16)] | [(None, 16)]
InputLayer | output:
embedding_2 | input: | (None, 16) | (None, 16, 512)
Embedding | output:
dropout_6 | input: | (None, 16, 512) | (None, 16, 512)
Dropout | output:
conv1d_2 | input: | (None, 16, 512) | (None, 14, 512)
Conv1D | output:
batch_normalization_4 | input: | (None, 14, 512) | (None, 14, 512)
BatchNormalization | output:
activation_4 | input: | (None, 14, 512) | (None, 14, 512)
Activation | output:
dropout_7 | input: | (None, 14, 512) | (None, 14, 512)
Dropout | output:
global_max_pooling1d_2 | input: | (None, 14, 512) | (None, 512)
GlobalMaxPooling1D | output:
dense_2 | input: | (None, 512) | (None, 128)
Dense | output:
batch_normalization_5 | input: | (None, 128) | (None, 128)
BatchNormalization | output:
activation_5 | input: | (None, 128) | (None, 128)
Activation | output:
dropout_8 | input: | (None, 128) | (None, 128)
Dropout | output:
concatenate | input: | [(None, 128), (None, 128), (None, 128)] | (None, 384)
Concatenate | output:
dense_3 | input: | (None, 384) | (None, 128)
Dense | output:
batch_normalization_6 | input: | (None, 128) | (None, 128)
BatchNormalization | output:
activation_6 | input: | (None, 128) | (None, 128)
Activation | output:
dropout_9 | input: | (None, 128) | (None, 128)
Dropout | output:
dense_4 | input: | (None, 128) | (None, 225)
Dense | output:

```
Epoch 1/5
196/196 [==============================] - ETA: 0s - loss: 0.8801 - accuracy: 0.7592
Epoch 1: val_accuracy did not improve from 0.70053
196/196 [==============================] - 29s 148ms/step - loss: 0.8801 - accuracy: 0.7592
Epoch 2/5
196/196 [==============================] - ETA: 0s - loss: 0.8784 - accuracy: 0.7599
Epoch 2: val_accuracy did not improve from 0.70053
196/196 [==============================] - 27s 138ms/step - loss: 0.8784 - accuracy: 0.7599
Epoch 3/5
196/196 [==============================] - ETA: 0s - loss: 0.8767 - accuracy: 0.7600
Epoch 3: val_accuracy did not improve from 0.70053
196/196 [==============================] - 27s 138ms/step - loss: 0.8767 - accuracy: 0.7600
Epoch 4/5
196/196 [==============================] - ETA: 0s - loss: 0.8765 - accuracy: 0.7602
Epoch 4: val_accuracy did not improve from 0.70053
196/196 [==============================] - 27s 138ms/step - loss: 0.8765 - accuracy: 0.7602
Epoch 5/5
196/196 [==============================] - ETA: 0s - loss: 0.8751 - accuracy: 0.7606
Epoch 5: val_accuracy did not improve from 0.70053
196/196 [==============================] - 27s 138ms/step - loss: 0.8751 - accuracy: 0.7606
```

Validation acc가 0.76으로 만족스럽지 못함

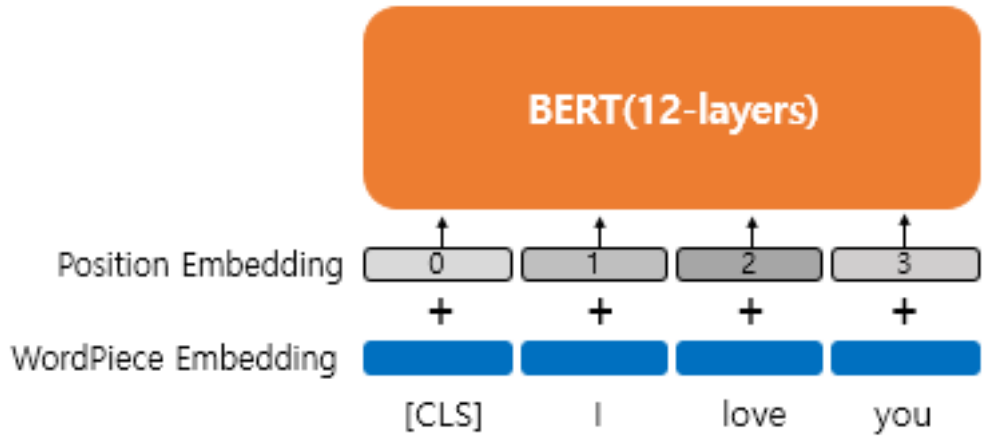
2 NLP 분류모델 활용

Wordpiece Tokenizer

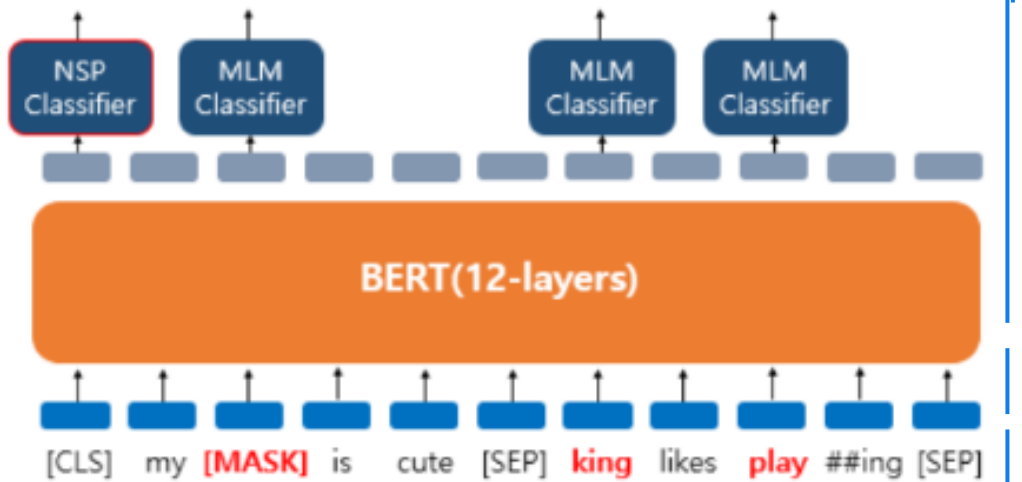
```
result = tokenizer.tokenize('Here is the sentence I want embeddings for.')
print(result)
```

```
['here', 'is', 'the', 'sentence', 'i', 'want', 'em', '##bed', '##ding', '##s', 'for', '.']
```

Positional Embedding



Mask



Pretrained BERT

- 두 개의 사전 학습된 모델 사용

BERT multilingual

Wikipedia를 이용해
104개 언어에 대해 사전 학습

Korean BERT (KoBERT)

한국어 BERT

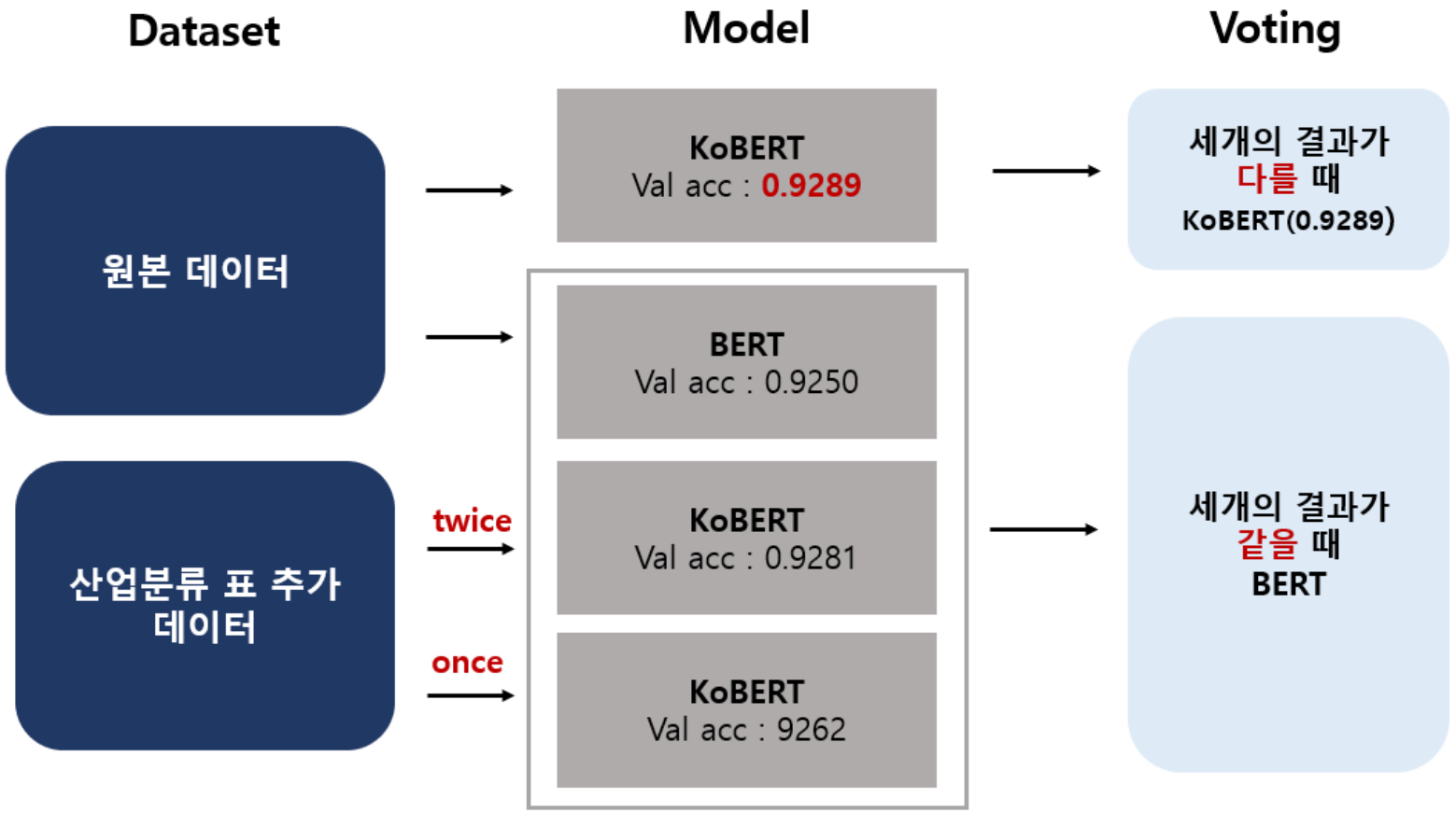
1. Original data input
2. 산업분류표를 Input으로 사용

Training

Korean BERT (KoBERT)

1. Original data input
2. 산업분류표 Added (once)
3. 산업분류표 Added (twice)

Model Ensemble




#1
연속적인
챗봇
데이터
활용
#2
Task1:
민원 챗봇
#3
분류모델
직접
앙상블

감사합니다