Natural Language Processing

김기영

kky416@snu.ac.kr

Ph.D. in engineering (Fluid dynamics, Applied mathematics)

Experience in Finance, Digital marketing and Healthcare

<비개발자를 위한 한국어 언어처리 인공지능 무료 강의>

코딩을 몰라도 내 데이터로 인공지능을 실행해 볼 수 있다고?

- AI기술이 대중화되고 있음에도 불구하고 여전히 초심자들의 **기술접근성이 좋지 않습니다**. 저 역시 그러한 경험을 하며 누군가가AI의 기본적인 원리와 활용법을 너무 어렵지도, 너무 쉽지도 않게 알려주었으면 좋겠다고 생각하였습니다.

- 이런 생각을 바탕으로 **비개발자들도 쉽게 인공지능을 이해하고 간단한 기능을 실행해 볼 수 있도록** 지난 2월 서울대 학생 110명을 대상으로 한국어 인공지능 강의를 열었습니다. 이렇게 쉽게 해볼 수 있는지 몰랐다는 좋은 의견과 피드백을 통해 강의를 확장하게 되었습니다.

- 수학은 **사칙연산**만 아셔도 이해하실 수 있게 설명드립니다. **직접 메모장과 엑셀로 된 데이터를 넣어 유사 문장을 골라보고, 문장의 긍부정 평가 인공지능을 만들어보세요**.

1. 주제: 인공지능 언어 처리 기초
2. 일정: 3월 17일 부터 4주간 (주 1회, 매주 수요일) 저녁 8시 ~ 9시반
3. 오픈카톡: https://open.kakao.com/o/g46ZM7Zc (참여코드 aiai)
   * 오픈카톡을 통해 강의링크 전달 및 질의응답을 해드립니다
4. 비용: 무료

김기영

- 서울대학교 기계항공공학부 학사, 박사(유체역학, 응용수학)
- 한국금거래소 디지털에셋 금융 상품 기획
- 디지털마케팅 회사 공팔리터 AI팀 PM (리뷰 분석 솔루션)
- AI기반 금융 & 마케팅 솔루션 개발 업체 Artificial Society 대표

Linkedin: https://www.linkedin.com/in/kiyoung-kim-a6315a170/

Github: https://github.com/kiyoungkim1

커리큘럼

✓ 한국어 언어처리 인공지능 **핵심 원리를 이해**합니다.

✓ 메모장과 엑셀파일로 된 내 데이터로 **인공지능 모델을 활용해** 봅니다.

날짜	내용
3.17	인공지능 동향, Bag of words와 TFIDF, 문서 유사도 분석
3.24	Word2Vec와 Fasttext, Doc2Vec, 유사 단어 찾기, 문서 유사도 분석 심화
3.31	Transformer, Transfer-learning(Bert, Gpt 소개), 감성분석
4.07	Huggingface 라이브러리를 활용한 최신 언어 모델 사용

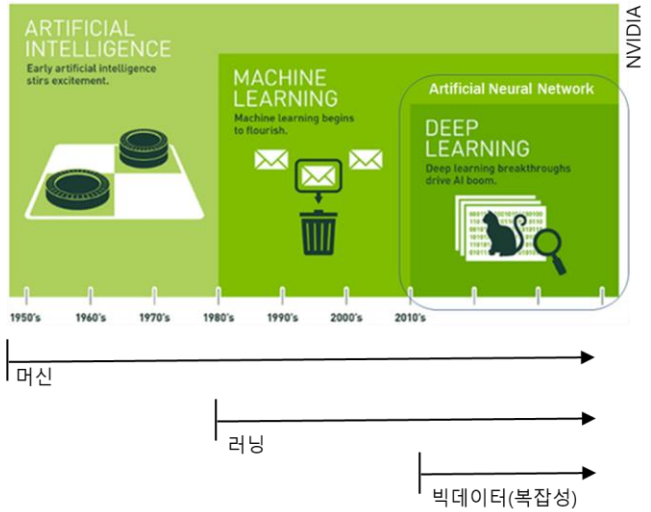
- 시간은 **20:00-21:30**이며 **링크는 시작 15분 전에 오픈카톡방**을 통해 전달
- 인공지능 **지식이 없어도 되며**, 수학은 **사칙연산**만 사용합니다.
- 강의는 **녹화**하여 올려드립니다. 주소는 오픈카톡방을 통해 전달
- **자율 참석**

문의: kky416@gmail.com

Bag of words, TFIDF

Artificial Intelligence

- 인공지능은 새로운 자동화 기술이다.
- 빅데이터와 머신러닝을 통해 그동안 자동화가 불가능할 것으로 생각되었던 영역의 자동화가 이루어지고 있다.



인공지능 동향

Bag of words와 TFIDF, 문서 유사도 분석

Embedding: Bag of words

1. 단어 빈도로 나타내자

뛰어쓰기로 나눌 것인가(학교에), 형태소로 나눌 것인가(학교+에) 아니면 다른 방법?

1. 이 단어 사전은 미리 구축해 둬야 한다 (Tokenize)

	학교에	가서	수업을	들었다	온건	오랜만이다	친구	얘기를	내일	뭐	먹지
학교에 가서 수업을 들었다. 학교에 온건 오랜만이다.	2	1	1	1	1	1	0	0	0	0	0
학교에 가서 친구 얘기를 들었다.	1	1	0	1	0	0	1	1	0	0	0
내일 가서 뭐 먹지?	0	0	0	0	0	0	0	0	1	1	1

2. 문장 속 단어 빈도를 표시한다

- 쉽고 직관적이지만 직관적으로 문장을 숫자로 변환할 수 있다
  - ✓ 학교에 가서 친구 얘기를 들었다. → [1, 1, 0, 1, 0, 0, 1, 1, 0, 0, 0]
- 단어 빈도를 기반으로 문서 유사도를 쉽게 파악할 수 있다.
  - ✓ 숫자 분포를 통해 1번 문장은 3번 문장보다 2번 문장과 유사함을 알 수 있다.

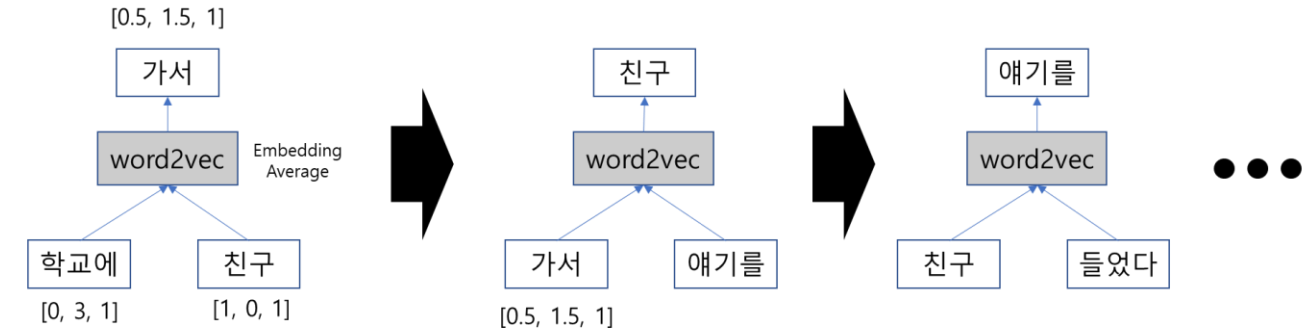
Word2Vec, Fasttext, Doc2Vec

Embedding: Word2Vec (2013)

2. 주변 맥락으로 단어를 표현해 보자

2. 주변 단어의 배열로 관심단어 배열을 만들자!
소수를 통해 더 적은 크기의 배열로도 다양한 단어 표현 가능.
(Bag of words의 경우 크기 3의 배열로는 3단어 밖에 표현 하지 못함)

학습 문장: 학교에 가서 친구 얘기를 들었다.
Bag of words: → [1, 1, 0, 1, 0, 0, 1, 1, 0, 0, 0]



1. 초기 단어를 임의의 배열로 세팅하자

3. 이 과정을 내가 가진 모든 텍스트에 대해 반복하여 학습.
어떤 문장들로 학습했는지(데이터셋)에 따라 각 단어를 나타내는 배열이 다를 것.

Word2Vec, Fasttext

유사 단어 찾기, 문서 유사도 분석 심화

Embedding: Word2Vec (2013)

Word2Vec 학습

```
[ ]  # word2vec training

     import os
     from gensim.models import Word2Vec

     def word2vec(texts):
         inputs = [tt.split(' ') for tt in texts]
         print('number of text = ', len(inputs))

         print('word2vec training...')
         model = Word2Vec(inputs, size=50, window=3, min_count=3, negative=5, workers=os.cpu_count(), iter=10, sg=1)
         model.init_sims

         model.save('word2vec')

     word2vec(texts)

     number of text =  136748
     word2vec training...
     word2vec is trained

[ ]  w2v = Word2Vec.load('word2vec')
```

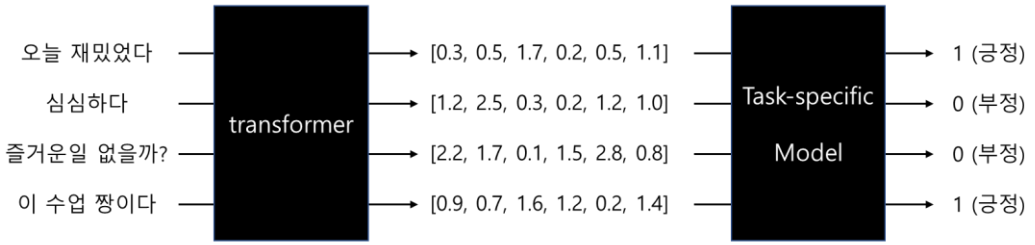
Word2Vec 결과

```
[20] # 단어 벡터
     w2v.wv['감동']

     array([-0.2877369 , -0.3411937 ,  0.98615265,  0.40388942, -0.2993487 ,
            -0.8082169 , -0.05968314, -0.14713229, -0.7971726 , -0.2910246 ,
             0.20702799,  0.1501431 ,  0.62676755,  0.38210574, -0.12299415,
             0.5009918 , -0.2925843 , -1.0950974 ,  0.01489488, -0.16576152,
            -0.05468882,  0.17707469, -0.72506976,  0.2971289 ,  0.10010708,
             1.0921265 , -0.94679564, -0.01515222,  0.03146487,  0.23118128,
             0.1426021 , -0.19204514, -0.27978763, -0.26251298, -0.68439114,
            -0.40017757,  0.9187491 ,  0.42633244,  0.85304224,  0.3333111 ,
             0.07864343,  0.2232901 ,  0.28953448,  0.3314806 , -0.99922657,
             0.13744463,  0.31774592,  0.87070864, -0.29360154,  0.22819856],
           dtype=float32)

[21] # 유사 단어
     w2v.wv.most_similar('이제까지')

     [('10년동안', 0.9651472568511963),
      ('여태껏', 0.9527884125709534),
      ('재난영화중', 0.9520287513732910),
      ('여태껏', 0.9492594003677368),
      ('수천편의', 0.9491361379623413),
      ('9년간', 0.9450587630271912),
      ('이제껏', 0.9443521499633789),
      ('정신병', 0.9437797665596008),
      ('원망스럽다', 0.9426261782646179),
      ('2006년', 0.9412530064582825)]
```

Transformer, Transfer-learning

Transfer leaning

3. 문장의 맥락을 파악해보자



언어를 이해하는 부분
Pre-training

언어를 이해하는 **일반적**인(general)한 영역이다.
대량의 텍스트로 학습 할 수록 성능이 좋아지며,
학습 비용이 비싸다.

태스크에 관한 부분
Fine-tuning

언어를 이해하는 임베딩이 이미 이루어졌으므로,
임베딩 된 배열을 통해 **관심있는 task를 수행**함.
학습 비용이 **저렴**하며, 개인용 컴퓨터로 가능하다.

✓ **Transfer learning:** 미리 대량의 텍스트로 언어 모델을 한번 학습 해두고(pre-training),
이 결과값에 관심있는 task에 적합한 모델을 붙여 다양한 문제를 푸는 모델을 만들어 보자 (fine-tuning)

Transfer-learning

감성분석

+ 코드 + 텍스트 | 드라이브로 복사

```
1    영화가 재밌다.
1    이 영화 추천해요.
0    지루한 영화였습니다.
...
```

```
[ ] !pip3 install -q transformers
    !git clone https://github.com/kiyoungkim1/ReadyToUseAI

    from ReadyToUseAI.src.nlp import make_sample_dataset, bert_sequence_classification
    make_sample_dataset.nsmc(mode='test', text_only=False)  # mode: which datasets? 'train' or 'test'
```

[Training]

- 첨부된 샘플의 경우 약 40min 소요 (Tesla T4 GPU)
- min_sentence_length보다 긴 문장만 사용합니다.
- MAX_LEN은 모델이 인식하는 token의 길이로, 전체길이가 약 MAX_LEN의 2배보다 긴 문장은 뒷부분이 삭제됩니다 (예를들어 MAX_LEN = 128이면, 대략 길이가 256이상인 문장은 뒷부분이 무시됨).
- batch_size는 한번에 몇개의 sample을 계산하는지를 나타내며, 제한된 메모리에서 MAX_LEN을 줄이면 batch_size를 키울 수 있고, MAX_LEN를 키우면 batch_size를 줄여야 합니다.
- epochs는 데이터셋을 몇번 반복해서 학습할지 여부이며, dataset_split은 전체 데이터 중 몇 %를 검증용 데이터셋으로 사용할지 여부입니다.

```
[ ] CLS = bert_sequence_classification.Classification(model_name='kykim/bert-kor-base', min_sentence_length=10, MAX_LEN=128, batch_size=32, use_bert_tokenizer=True)
    CLS.dataset(data_path='dataset.xlsx')
    CLS.load_model(mode='train')
    CLS.train(epochs=3, dataset_split=0.1)
```

[Inference]

- sentences에 원하는 문장을 아래 형식과 같이 넣으면 해당하는 카테고리를 반환합니다.
- saved_model_path는 학습된 모델이 저장된 '폴더명'입니다.

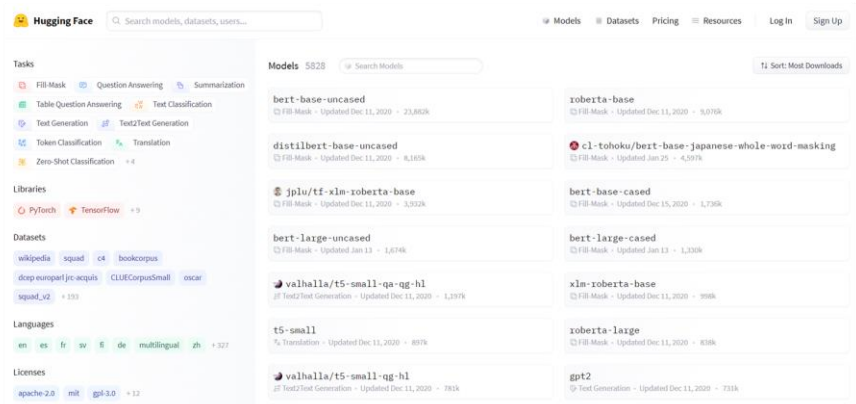
```
[ ] sentences = ['영화 재밌어요', '영화 재미없어요', '그냥 시간떼우기용', '완전 추천작']
    saved_model_path='model/saved/3'

    CLS = bert_sequence_classification.Classification(model_name='kykim/bert-kor-base', min_sentence_length=10, MAX_LEN=128, batch_size=64, use_bert_tokenizer=True)
    CLS.load_model(mode='inference', saved_model_path=saved_model_path)
    logit = CLS.inference(sentences=sentences)
    print(logit)  # 네이버 영화평의 경우 0은 부정 카테고리, 1은 긍정 카테고리로 설정되어 있음
```

Huggingface 라이브러리

Huggingface library

- 개인 또는 기관(google, facebook 등)이 pre-training한 모델을 업로드하고, 유저는 그것을 받아 fine-tuning

  ✓ **Official Pre-trained Models**: https://huggingface.co/transformers/pretrained_models.html



✓ **Pre-trained Models**
https://huggingface.co/models

Pre-training 모델

최신 언어 모델 활용

Huggingface library

- 많은 최근 연구 결과를 직접 사용해 볼 수 있음 → **실습**

```
from transformers import T5TokenizerFast, T5ForConditionalGeneration

tokenizer = T5TokenizerFast.from_pretrained('t5-small')
model = T5ForConditionalGeneration.from_pretrained('t5-small')

text = "translate to german: I like this class very much"
input_ids = tokenizer.encode(text, return_tensors='pt')

outputs = model.generate(input_ids)
tokenizer.decode(outputs[0], skip_special_tokens=True)
```

```
'Ich mag diese Klasse sehr'
```

T5로 번역