

Natural Language Processing

- 1차 -

2020.03.17

김기영

kky416@gmail.com

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

Experience in Finance, Digital marketing and Healthcare






The General Language Understanding Evaluation (GLUE) benchmark

Rank Name		Model	URL	Score	CoLA	SST-2	MRPC	STS-B	QQP	M
1	ERNIE Team - Baidu	ERNIE	↗	90.9	74.4	97.8	93.9/91.8	93.0/92.6	75.2/90.9	
2	DeBERTa Team - Microsoft	DeBERTa / TuringNLRv4	↗	90.8	71.5	97.5	94.0/92.0	92.9/92.6	76.2/90.8	
3	HFL iFLYTEK	MacALBERT + DKM		90.7	74.8	97.0	94.5/92.6	92.8/92.6	74.7/90.6	
+	4	Alibaba DAMO NLP	StructBERT + TAPT	↗	90.6	75.3	97.3	93.9/91.9	93.2/92.7	74.8/91.0
+	5	PING-AN Omni-Sinitic	ALBERT + DAAF + NAS		90.6	73.5	97.2	94.0/92.0	93.0/92.4	76.1/91.0
6	T5 Team - Google	T5	↗	90.3	71.6	97.5	92.8/90.4	93.1/92.8	75.1/90.6	
7	Microsoft D365 AI & MSR AI & GATECH	HMT-DNN-SMART	↗	89.9	69.5	97.5	93.7/91.6	92.9/92.5	73.9/90.2	
+	8	Huawei Noah's Ark Lab	NEZHA-Large		89.8	71.7	97.3	93.3/91.0	92.4/91.9	75.2/90.7
+	9	Zihang Dai	Funnel-Transformer (Ensemble B10-10-10H1024)	↗	89.7	70.5	97.5	93.4/91.2	92.6/92.3	75.4/90.7
+	10	ELECTRA Team	ELECTRA-Large + Standard Tricks	↗	89.4	71.7	97.1	93.1/90.7	92.9/92.5	75.6/90.8
+	11	Microsoft D365 AI & UMD	FreeLB-RoBERTa (ensemble)	↗	88.4	68.0	96.8	93.1/90.8	92.3/92.1	74.8/90.3
12	Junjie Yang	HIRE-RoBERTa	↗	88.3	68.6	97.1	93.0/90.7	92.4/92.0	74.3/90.2	

GLUE

<https://gluebenchmark.com/leaderboard>

Leaderboard Version: 2.0

	Rank	Name	Model	URL	Score	BoolQ	CB	COPA	MultiRC	ReCoRD	RTE	WiC	WSC	AX-b	AX-g
+	1	Zirui Wang	T5 + Meena, Single Model (Meena Team - Google Brain)		90.4	91.4	95.8/97.6	98.0	88.3/63.0	94.2/93.5	93.0	77.9	96.6	69.1	92.7/91.9
+	2	DeBERTa Team - Microsoft	DeBERTa / TuringNLRv4		90.3	90.4	95.7/97.6	98.4	88.2/63.7	94.5/94.1	93.2	77.5	95.9	66.7	93.3/93.8
	3	SuperGLUE Human Baselines	SuperGLUE Human Baselines		89.8	89.0	95.8/98.9	100.0	81.8/51.9	91.7/91.3	93.6	80.0	100.0	76.6	99.3/99.7
+	4	T5 Team - Google	T5		89.3	91.2	93.9/96.8	94.8	88.1/63.3	94.1/93.4	92.5	76.9	93.8	65.6	92.7/91.9
+	5	Huawei Noah's Ark Lab	NEZHA-Plus		86.7	87.8	94.4/96.0	93.6	84.6/55.1	90.1/89.6	89.1	74.6	93.2	58.0	87.1/74.4
+	6	Alibaba PAI&ICBU	PAI Albert		86.1	88.1	92.4/96.4	91.8	84.6/54.7	89.0/88.3	88.8	74.1	93.2	75.6	98.3/99.2
+	7	Infosys : DAWN : AI Research	RoBERTa-icETS		86.0	88.5	93.2/95.2	91.2	86.4/58.2	89.9/89.3	89.9	72.9	89.0	61.8	88.8/81.5
+	8	Tencent Jarvis Lab	RoBERTa (ensemble)		85.9	88.2	92.5/95.6	90.8	84.4/53.4	91.5/91.0	87.9	74.1	91.8	57.6	89.3/75.6
	9	Zhuiyi Technology	RoBERTa-mtl-adv		85.7	87.1	92.4/95.6	91.2	85.1/54.3	91.7/91.3	88.1	72.1	91.8	58.5	91.0/78.1
	10	Facebook AI	RoBERTa		84.6	87.1	90.5/95.2	90.6	84.4/52.5	90.6/90.0	88.2	69.9	89.0	57.9	91.0/78.1
+	11	Anuar Sharafudinov	AILabs Team, Transformers		82.6	88.1	91.6/94.8	86.8	85.1/54.7	82.8/79.8	88.9	74.1	78.8	100.0	100.0/100.0

SuperGLUE

<https://super.gluebenchmark.com/leaderboard>

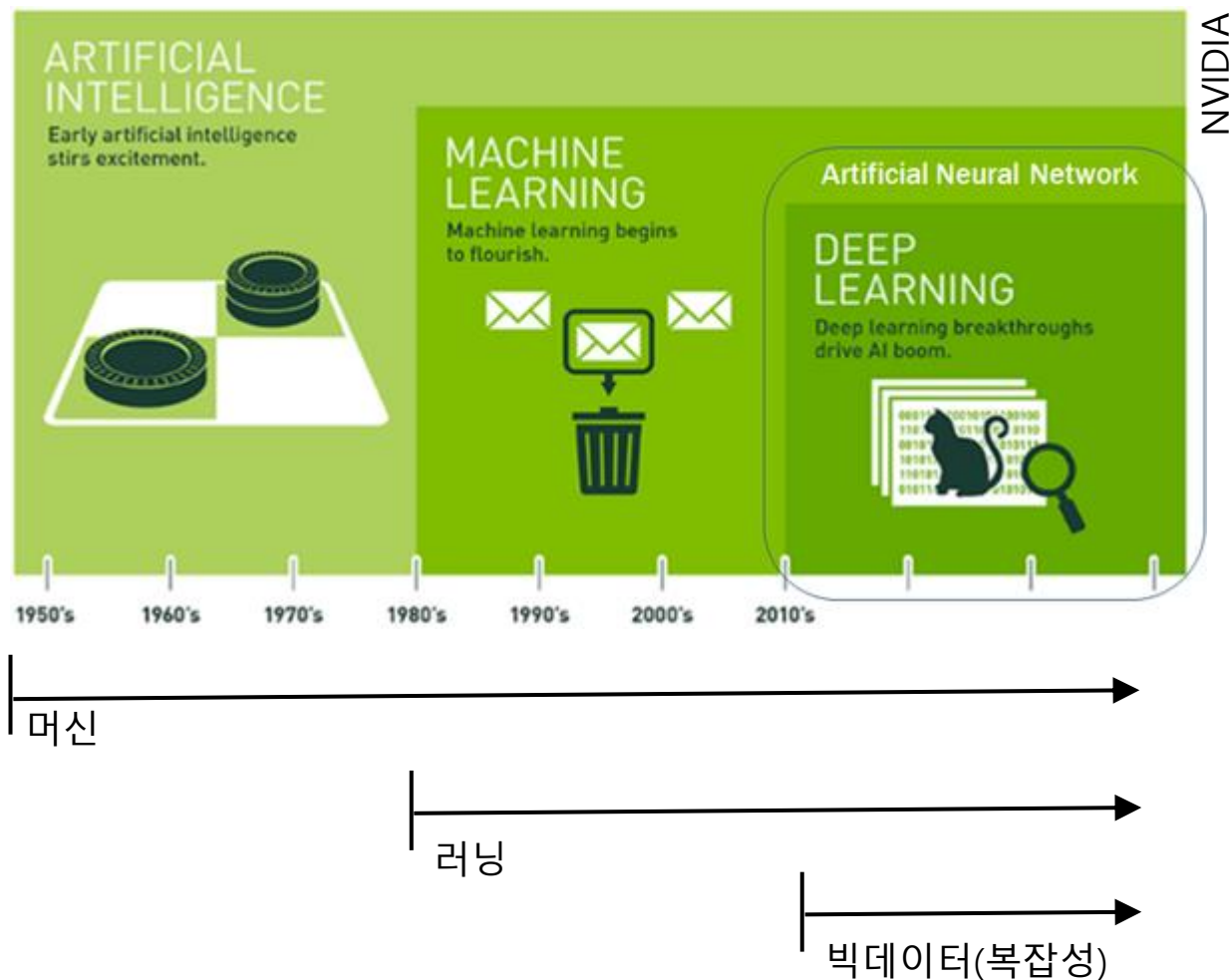
Introduction

- 자연어처리는 2018년 BERT가 나오며 **10% 이상의 성능향상**을 이루었고 지금은 **그로부터 또 10% 이상의 성능향상**을 이루었다.
- 이 과정에서 발생한 대부분의 연구 과정과 결과는 인터넷 상에 **공개**되어 있다. 연구 진행속도가 매우 빠르기에 쉽게 사용할 수 있도록 **오픈소스화** 되지 않은 연구는 학계에서 널리 퍼지기 어렵다.
- 오픈소스 코드는 도서관의 영어책과 같다. 누구나 도서관에 가서 읽을 수 있도록 되어있지만, 소수의 사람만이 책을 **읽고 활용**한다.
- 99%의 정확도로 task를 수행할 수 있다면 **학계**에서는 좋은 모델로 인정받을 수 있다. 하지만 100번에 1번 틀리는 서비스는 좋은 **서비스**가 될 수 없다.

본 PPT에서는 모델의 개념을 단순화 및 일반화하여 설명하므로 실제 사용 모델과는 약간의 차이가 있을 수 있습니다.

Artificial Intelligence

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



Natural Language Processing

- 자연어(컴퓨터 프로그래밍 언어가 아닌 인간이 쓰는 언어)를 다루는 분야로 문서요약, 감성분석, 질의응답, 문장생성을 아우른다.
- 그런데 어떻게 컴퓨터가 언어를 이해하지?
 - 컴퓨터는 언어를 있는 그대로 이해하는 것이 아니라 **숫자**로 계산한다.
 - 우리의 일상언어를 숫자로 표현하는 것을 **임베딩**(embedding)이라 한다.
 - 사과(1), 포도(2), 학교(3)과 같이 언어에 무작위하게 숫자를 붙이는 방법에서 유사한 단어에 유사한 숫자가 부여도 되도록, 유사한 문장을 유사한 숫자로 표현되도록 하는 방향으로 발전하고 있다.
 - 언어처리의 가장 중요한 개념이 바로 이 임베딩이다. 서로 다른 모델은 서로 다른 임베딩 방법을 가지고 있다.

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 뒀야 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 얘 기를 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.

1. 단어 빈도로 나타내자

1. 이 단어 사전은 미리 구축해 뒀야 한다 (Tokenize)

[illegible]

2. 문장 속 단어 빈도를 표시한다

- 단어의 빈도만 고려할 뿐 순서를 고려하지 않는다.
 - ✓ 학교에 가서 친구 얘기를 들었다. → [1, 1, 0, 1, 0, 0, 1, 1, 0, 0, 0] → [학교에, 가서, 들었다, 친구, 얘기를]
- 세상의 모든 단어에 숫자를 붙여야 된다. 너무 많아..
 - ✓ 단어 개수가 늘어나면 배열의 크기가 10만개~100만개로 엄청나게 커진다. 그러나 실제로 대부분의 값은 0이다.

[1, 1, 0, 1, 0, 0, 1, 1, 0,]]

오늘 저녁은 맛있었다

1. 띄어쓰기 단위 → [오늘, 저녁은, 맛있었다]

- 명료하고 적용하기 쉽다.
- '맛있다, 맛있어요, 맛있었다'가 모두 다르게 인식되며, 단어사전이 매우 커짐.

2. 문자 단위 → [오, 늘, 저, 녀, 은, 맛, 있, 었, 다]

- 각 token이 의미를 담지 못한다. eg) 학교에 → '학', '교', '에'

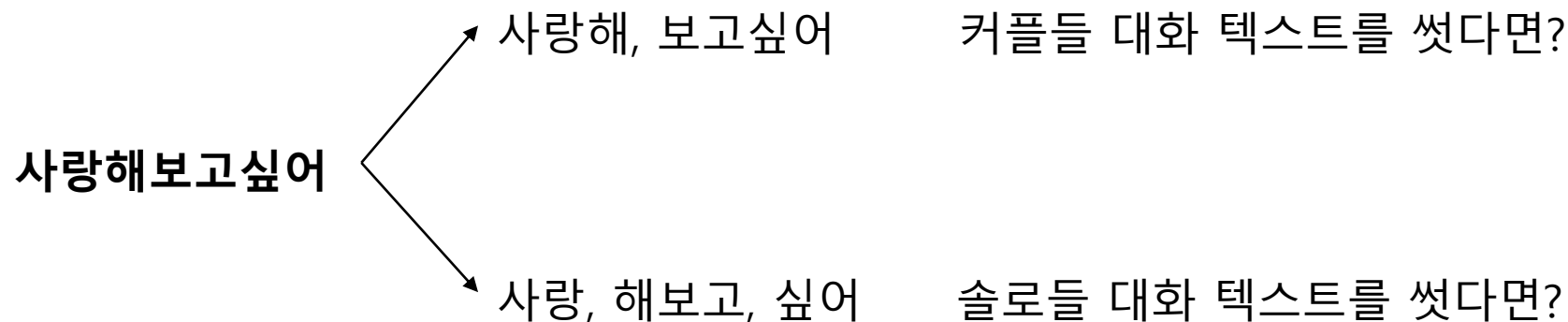
3. Subword 단위 → [오늘, 저녁, 은, 맛있, 었, 다]

3.1. 형태소로 나누기: 매우 효율적이지만 언어 지식이 필요함(컴퓨터가 알아서 할 수 없음)

eg) '학교에' → '학교' + '에'

3.2. BPE(byte-pair encoding): 전체 문서를 문자 단위로 쪼갠 뒤 빈번하게 나오는 문자들을 묶어 단어사전의 수를 줄임.

eg) '학', '교'로 따로 쓰이는 것보다 '학교' 붙여 쓴 게 많으면 후자를 단어사전에 채택



- ✓ Subword tokenizer도 학습해야 하며, **학습한 텍스트 데이터셋이 따라 결과가 다르다.**
 - 띄어쓰기나 문자단위 tokenizer는 학습할 필요가 없음.
 - tokenizer 학습은 인공지능 모델의 학습과의 별도로 이루어짐.
- ✓ 모든 언어처리 과정은 tokenizer로 부터 시작한다.
- ✓ Tokenizer가 좋아야 embedding을 잘 시킬 수 있고, embedding이 잘되면 모델의 성능이 좋아진다.

Tokenizer

Vocab

Vocab Len	lower_case	strip_accent
42000	True	False

- 한글, 영어, 숫자와 일부 특수문자를 제외한 문자는 학습에 방해가 된다고 판단하여 삭제하였습니다(예시: 한자, 이모지 등)
- [Huggingface tokenizers](#) 의 wordpiece 모델을 사용해 40000개의 subword를 생성하였습니다.
- 여기에 2000개의 unused token과 넣어 학습하였으며, unused token는 도메인 별 특화 용어를 담기 위해 사용됩니다.

<https://github.com/kiyoungkim1/LMkor>

kykim	Create README.md	70ece2f
.gitattributes	345.0B	↓
README.md	442.0B	↓
config.json	725.0B	↓
pytorch_model.bin	453.7MB	↓
tf_model.h5	451.5MB	↓
tokenizer_config.json	80.0B	↓
vocab.txt	336.2KB	↓

<https://huggingface.co/kykim/bert-kor-base/tree/main>

25307 인텔
25308 땀을
25309 아라
25310 ##니를
25311 확률이
25312 자연의
25313 다니기
25314 ##두기
25315 콜라보
25316 다른데
25317 북미
25318 그들
25319 큐브
25320 입이
25321 향산화
25322 봤다
25323 ##일지
25324 사용하다
25325 말아야
25326 잡혀
25327 입원
25328 ##분해
25329 쓴다
25330 유대

Embedding: Bag of words 실습

```
[ ] !git clone https://github.com/kiyoungkim1/ReadyToUseAI

from ReadyToUseAI.src.nlp import make_sample_dataset, count, tfidf
make_sample_dataset.simple()
```

[6] # 단어 빈도수를 계산 후 결과를 엑셀파일로 저장할 경우 실행

```
df, _ = count.apply(dataset_path='dataset.txt', save_path='count.xlsx')
df
```

Result is saved at count.xlsx

	가서	학교에	들었다	간건	내일	먹지	수업을	얘기를	오랜만이다	친구
total	3	3	2	1	1	1	1	1	1	1
학교에 가서 수업을 들었다. 학교에 간건 오랜만이다.	1	2	1	1	0	0	1	0	1	0
학교에 가서 친구 얘기를 들었다.	1	1	1	0	0	0	0	1	0	1
내일 가서 뭐 먹지?	1	0	0	0	1	1	0	0	0	0

[7] # 단어 빈도수를 계산 후 아래의 text와 유사한 문장 고르기

```
text = '학교 가서'
df = count.find_similar_sentence(text, dataset_path='dataset.txt', save_path='count_similarity.xlsx')
df
```

Result is saved at count_similarity.xlsx

	similarity	가서	학교에	들었다	간건	내일	먹지	수업을	얘기를	오랜만이다	친구
내일 가서 뭐 먹지?	0.577350	1	0	0	0	1	1	0	0	0	0
total	0.557086	3	3	2	1	1	1	1	1	1	1
학교에 가서 친구 얘기를 들었다.	0.447214	1	1	1	0	0	0	0	1	0	1
학교에 가서 수업을 들었다. 학교에 간건 오랜만이다.	0.333333	1	2	1	1	0	0	1	0	1	0

<https://github.com/kiyoungkim1/ReadyToUseAI>

→ notebooks/count_and_tfidf

Embedding: TFIDF

- **TFIDF** (Term Frequency-Inverse Document Frequency)

오늘은 기분이 좋다 → 오늘, 은, 기분, 이, 좋, 다

내일은 수업이 없다 → 내일, 은, 수업, 이, 없, 다

기분이 최고 → 기분, 이, 최고

the book is on the desk

the sky and the see

book and pencil

- ✓ 빈번하게 나타나는 단어지만 문장의 특징을 나타내지 않는 단어(a, the, 조사 등)의 가중치를 낮출 순 없을까?
- ✓ 단어가 나온 문장의 수로 나눠주자!
- ✓ 다른 문장에는 자주 안 나오는데 특정 문장에서 빈번하게 등장하는 단어 → **키워드(중요한 단어)**

TFIDF ~ $\log(\frac{\text{해당 문장의 단어 수}}{\text{단어가 나오는 문장 수}})$

Bag of words에 해당 여러 문장에 나올 수록
단어의 가중치를 낮춤

Embedding: TFIDF 실습

```
[ ] !git clone https://github.com/kiyoungkim1/ReadyToUseAI

from ReadyToUseAI.src.nlp import make_sample_dataset, count, tfidf
make_sample_dataset.simple()
```

[8] # TFIDF를 계산 후 결과를 엑셀파일로 저장할 경우 실행

```
df, _ = tfidf.apply(dataset_path='dataset.txt', save_path='tfidf.xlsx')
df
```

Result is saved at tfidf.xlsx

	학교에	가서	들었다	내일	먹지	얘기를	친구	간건	수업을	오랜만이다
total	1.015059	0.937236	0.710626	0.652491	0.652491	0.534093	0.534093	0.400294	0.400294	0.400294
학교에 가서 수업을 들었다. 학교에 간건 오랜만이다.	0.608868	0.236420	0.304434	0.000000	0.000000	0.000000	0.000000	0.400294	0.400294	0.400294
학교에 가서 친구 얘기를 들었다.	0.406192	0.315444	0.406192	0.000000	0.000000	0.534093	0.534093	0.000000	0.000000	0.000000
내일 가서 뭐 먹지?	0.000000	0.385372	0.000000	0.652491	0.652491	0.000000	0.000000	0.000000	0.000000	0.000000

[9] # 단어 빈도수를 계산 후 아래의 text와 유사한 문장 고르기

```
text = '학교 가서'
df = tfidf.find_similar_sentence(text, dataset_path='dataset.txt', save_path='tfidf_similarity.xlsx')
df
```

Result is saved at tfidf_similarity.xlsx

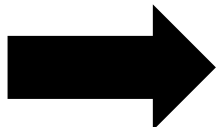
	similarity	학교에	가서	들었다	내일	먹지	얘기를	친구	간건	수업을	오랜만이다
학교에 가서 수업을 들었다. 학교에 간건 오랜만이다.	0.608868	0.608868	0.236420	0.304434	0.000000	0.000000	0.000000	0.000000	0.400294	0.400294	0.400294
total	0.488571	1.015059	0.937236	0.710626	0.652491	0.652491	0.534093	0.534093	0.400294	0.400294	0.400294
학교에 가서 친구 얘기를 들었다.	0.406192	0.406192	0.315444	0.406192	0.000000	0.000000	0.534093	0.534093	0.000000	0.000000	0.000000
내일 가서 뭐 먹지?	0.000000	0.000000	0.385372	0.000000	0.652491	0.652491	0.000000	0.000000	0.000000	0.000000	0.000000

<https://github.com/kiyoungkim1/ReadyToUseAI>

→ notebooks/count_and_tfidf

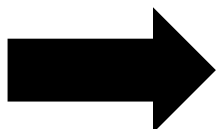
Next class

- **Term frequency** (원래 bag of words): 숫자 배열은 문장 내 단어의 **빈도수**
- **TFIDF** (업그레이드 된 bag of words): 숫자 배열은 문장 내 단어의 **중요도**



언어 처리를 할 때 가장 먼저 빠르게 결과를 볼 수 있는 방법

- 단어의 빈도만 고려할 뿐 순서를 고려하지 않는다.
 - ✓ 학교에 가서 친구 얘기를 들었다. → [1, 1, 0, 1, 0, 0, 1, 1, 0, 0, 0] → [학교에, 가서, 들었다, 친구, 얘기를]
- 세상의 모든 단어에 숫자를 붙여야 된다. 너무 많아..
 - ✓ 단어 개수가 늘어나면 배열의 크기가 10만개~100만개로 엄청나게 커진다. 그러나 실제로 대부분의 값은 0이다.
[1, 1, 0, 1, 0, 0, 1, 1, 0,]



단점을 극복하기 위해 순서를 고려하여 주변 맥락으로 단어를 표현해 보자

3/24 20시: "Word2Vec" and "Fasttext"