Natural Language Processing

- 개발자 그룹 -

2020.02.10

김기영

kky416@snu.ac.kr

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

Experience in Finance, Digital marketing and Healthcare

Introduction

- 언어처리의 경우 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.

  - ✓ 연구을 진행속도가 매우 빠르기에 쉽게 사용할 수 있도록 **오픈소스화** 되지 않은 연구는 학계에서 널리 퍼지기 어려웠으며, 따라서 많은 연구는 공개되어 있다.
  - ✓ 오픈소스 코드는 도서관의 영어책과 같다. 누구나 도서관에 가서 읽을 수 있도록 되어있지만, 소수의 사람만이 책을 **읽고 활용**한다.

- 인공지능의 상용화가 가까워지는 현재에는 기업들에서 연구를 **비공개 하려는 경향**이 보이기도 한다(예: GPT3)

- 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 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 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.

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

- 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, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...]

Tokenizer

1. 뛰어쓰기 단위

- 쉽지만 '맛있다, 맛있어요, 맛있었다 ' 와 같은 단어가 모든 다르게 인식되며, 단어사전이 매우 커짐.

2. 문자 단위

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

3. **Subword 단위**

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

eg) '학교에' → '학교' + '에'

3.2. Wordpiece, BPE: 빈번하게 나오는 문자들을 묶어 가며 단어사전의 수를 줄임.

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

좋은 Tokenizer가 embedding을 잘 시킬 수 있고, embedding이 잘되면 모델의 성능이 좋아진다.

Embedding: Bag of words

- **Bag of words**

```
[7]  corpus = [
         '학교에 가서 수업을 들었다. 학교에 간건 오랜만이다.',
         '학교에 가서 친구 얘기를 들었다.',
         '내일 가서 뭐 먹지?'
     ]
```

```
[8]  from sklearn.feature_extraction.text import CountVectorizer

     vect = CountVectorizer()
     vect.fit(corpus)
     vect.vocabulary_
```

```
{'가서': 0,
 '간건': 1,
 '내일': 2,
 '들었다': 3,
 '먹지': 4,
 '수업을': 5,
 '얘기를': 6,
 '오랜만이다': 7,
 '친구': 8,
 '학교에': 9}
```

```
vect.transform(corpus).toarray()
```

```
array([[1, 1, 0, 1, 0, 1, 0, 1, 0, 2],
       [1, 0, 0, 1, 0, 0, 1, 0, 1, 1],
       [1, 0, 1, 0, 1, 0, 0, 0, 0, 0]])
```

```
[10] vect.transform(['수업을 들었다. 수업은 재미있다.']).toarray()
```

```
array([[0, 0, 0, 1, 0, 1, 0, 0, 0, 0]])
```

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

'the book is on the desk'

'the sky and the see'

'book and pencil'

✓ 빈번하게 나타나는 단어지만 문장의 특징을 나타내지 않는 단어(a, the, 전치사 등)의 가중치를 낮출 순 없을까?

✓ 단어가 나온 문장의 수로 나눠주자!

TFIDF ~ 해당 문장의 단어수 / 단어가 나오는 문장 수

Bag of words에 해당

여러 문장에 나올 수록 단어의 가중치를 낮춤

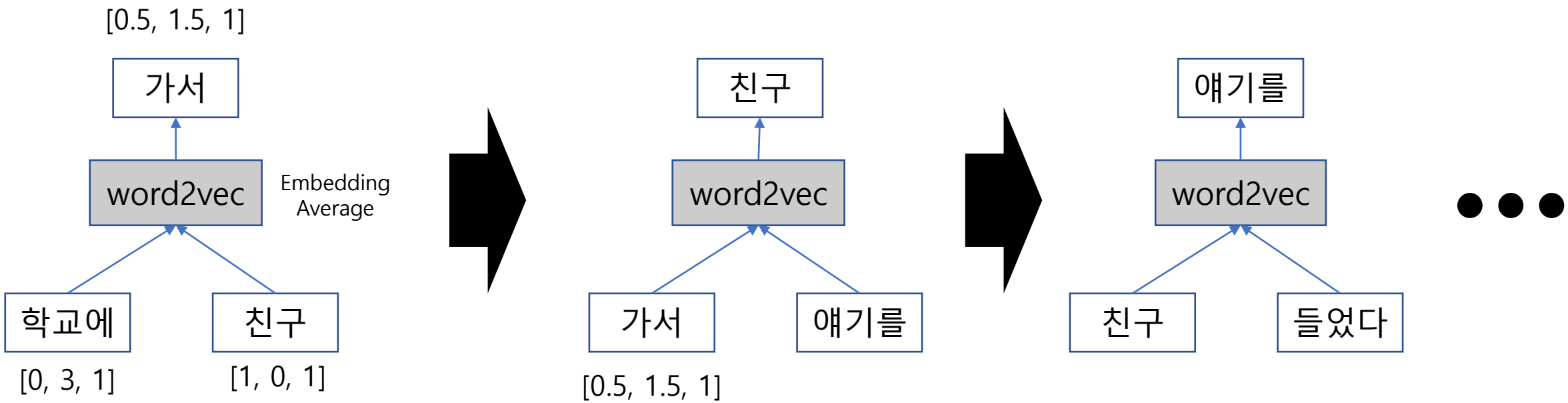
```
from sklearn.feature_extraction.text import TfidfVectorizer

tfidv = TfidfVectorizer().fit(corpus)
tfidv.transform(corpus).toarray()
```

```
array([[0.23642005, 0.40029393, 0.        , 0.30443385, 0.        ,
        0.40029393, 0.        , 0.40029393, 0.        , 0.60886771],
       [0.31544415, 0.        , 0.        , 0.40619178, 0.        ,
        0.        , 0.53409337, 0.        , 0.53409337, 0.40619178],
       [0.38537163, 0.        , 0.65249088, 0.        , 0.65249088,
        0.        , 0.        , 0.        , 0.        , 0.        ]])
```

2. 주변 맥락으로 단어를 표현해 보자

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

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)

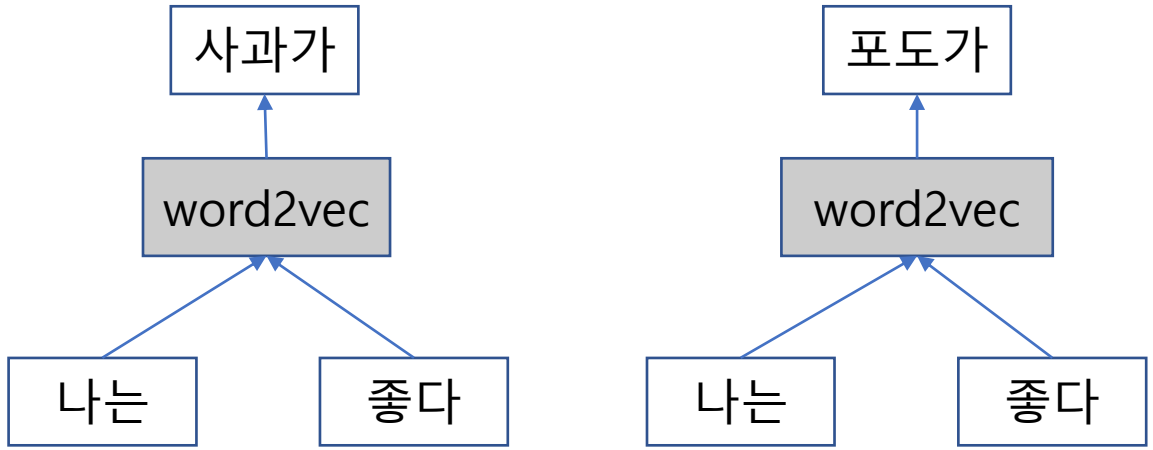


1. 초기 단어를 임의의 배열로 세팅하자

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

2. 주변 맥락으로 단어를 표현해 보자

- 주변단어가 비슷한것 단어들은 배열이 유사해 질 것.즉, 유사단어끼리 유사한 배열을 가질 수 있다.



문맥상 '사과가', '포도가' 이 두개가 비슷한 것을 학습 가능

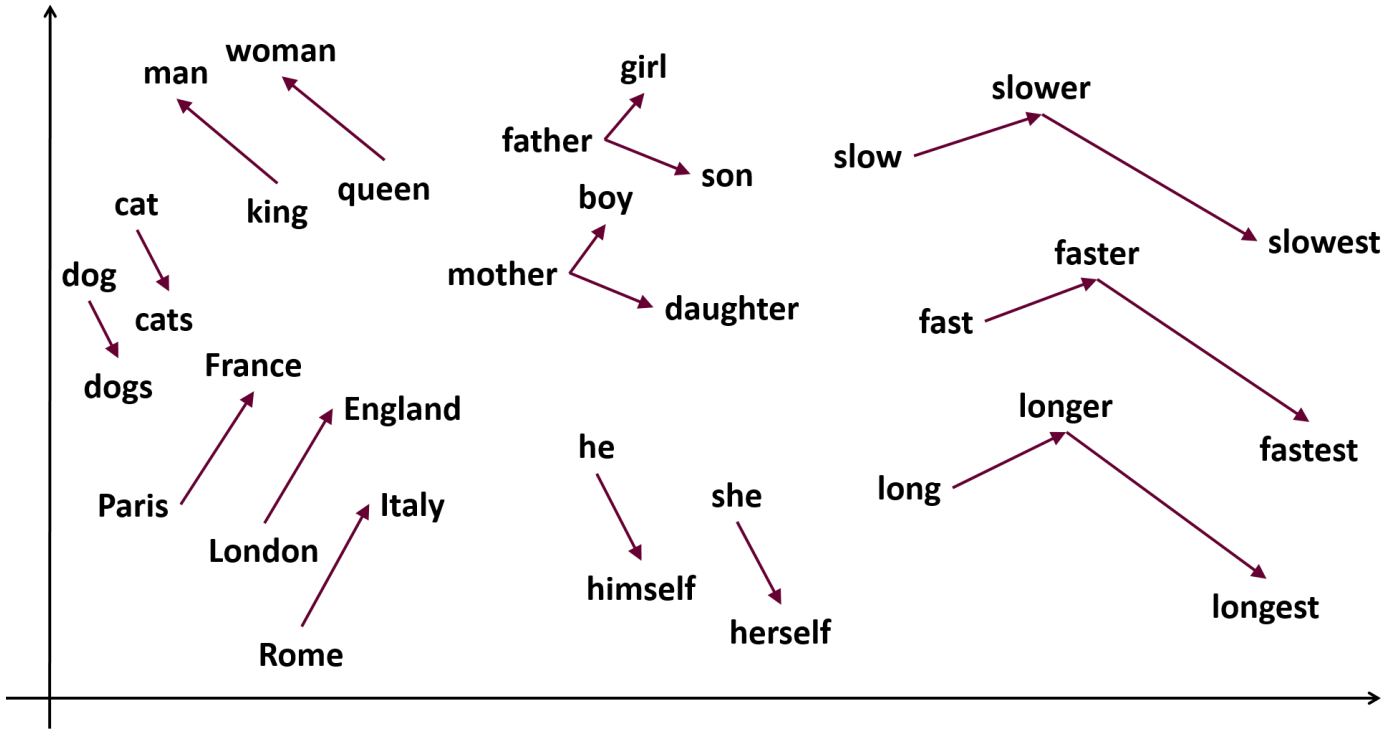
Word2vec을 이용한 단어 유사도 평가 예시

감동	고등학교	이제까지	명작
웃음	초등학교	10년동안	명작.
그자체	중학교	3년간	명작이다.
소름	이제까지	여태것	불후의
재미와	근	재난영화중	명작중의
영상	2학년때	머리털나고	걸작

26MB의 네이버 영화 텍스트 데이터로 1분간 학습한 결과

2. 주변 맥락으로 단어를 표현해 보자

- 주변단어가 비슷한것 단어들은 배열이 유사해 질 것.즉, 유사단어끼리 유사한 배열을 가질 수 있다.
- 배열은 단어의 주변 맥락을 표현한다. 이를 더하고 뺌으로써 단어 사이의 관계를 파악할 수 있다.



https://medium.com/analytics-vidhya/implementing-word2vec-in-tensorflow-44f93cf2665f

King – man + woman = **queen**

[1.5, 0, 0] – [0, 1, 0] + [1.5, 1, 0] = **[3, 0, 0]**

- 매우 빠른 속도와 합리적인 성능으로 word2vec은 2021년 현재에도 속도가 중요한 상용 시스템에서 많이 쓰이는 중이다.

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

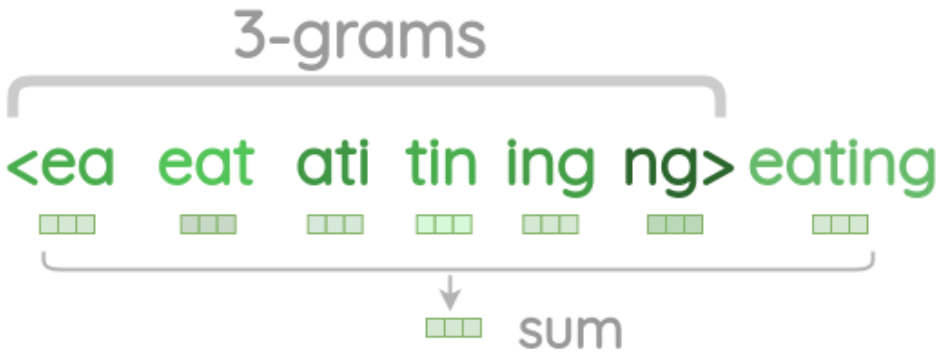
```
[20] # 단어 벡터
     w2v.wv['감동']
```

```
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 ,  0.62876755,  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),
 ('여태것', 0.9527884125709534),
 ('재난영화중', 0.952028751373291),
 ('여태껏', 0.9492594003677368),
 ('수천편의', 0.9491361379623413),
 ('3년간', 0.9450587630271912),
 ('이제껏', 0.9443521499633789),
 ('정신병', 0.9437797665596008),
 ('원망스럽다', 0.9426261782646179),
 ('2006년', 0.9412530064582825)]
```

Embedding: Word2Vec (2013)



https://simonezz.tistory.com/54

- 단어를 문자단위로 쪼개서, 각 조각마다 word2vec을 수행.
- 중간에 오타가 있어도 합리적인 결과를 얻을 수 있음.
- 한글의 경우 자모로 쪼개서 할 경우 성능이 더 향상됨.

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

```
[25] wav.wv.most_similar('고능학교')
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-25-c60380e797c6> in <module>()
----> 1 wav.wv.most_similar('고능학교')

NameError: name 'wav' is not defined
```

SEARCH STACK OVERFLOW

```
[24] fasttext.wv.most_similar('고능학교')
```

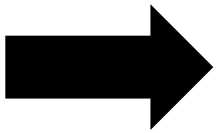
```
[('학교', 0.9648439884185791),
 ('중학교', 0.9563544988632202),
 ('고등학교', 0.9352776408195496),
 ('초등학교', 0.9316931962966919),
 ('국민학교', 0.9257780909538269),
 ('대학교', 0.9132347106933594),
 ('2학년', 0.8956236839294434),
 ('다닐', 0.8853365778923035),
 ('친구집에', 0.8824955821037292),
 ('1학년', 0.8821584582328796)]
```

오타가 있을 때 Word2Vec과 Fasttext

2. 주변 맥락으로 단어를 표현해 보자

- 동음이의어를 구분할 수 없다.
  - ✓ 우리는 **말을** 보며 **말을** 나눴다 → **말을** 이라는 단어의 배열에는 horse와 mouth의 맥락이 혼재된다.

- 단어 단위로 학습하다보니 문장 단위의 맥락을 이해하지 못한다.
  - ✓ 어제 식당에서 친구를 만났다. **거기서** 함께 밥을 먹었다 → **거기서**가 무엇을 가리키는지 알지 못한다.

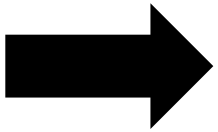


문장 단위의 문맥을 고려할 필요성이 생긴다.

2. 주변 맥락으로 단어를 표현해 보자

- 동음이의어를 구분할 수 없다.
  - ✓ 우리는 **말을** 보며 **말을** 나눴다 → **말을** 이라는 단어의 배열에는 horse와 mouth의 맥락이 혼재된다.

- 단어 단위로 학습하다보니 문장 단위의 맥락을 이해하지 못한다.
  - ✓ 어제 식당에서 친구를 만났다. **거기서** 함께 밥을 먹었다 → **거기서**가 무엇을 가리키는지 알지 못한다.



문장 단위의 문맥을 고려할 필요성이 생긴다.

2/17 20시: “Transformer” and “Transfer learning”