단어의 표현(Word Representation)

BOW,TDM,TF-IDF

Fininsight

데이터 분석가 김현진

단어의 표현

단어의 표현이 필요한 이유

자연어 처리를
시작해보겠습니다.

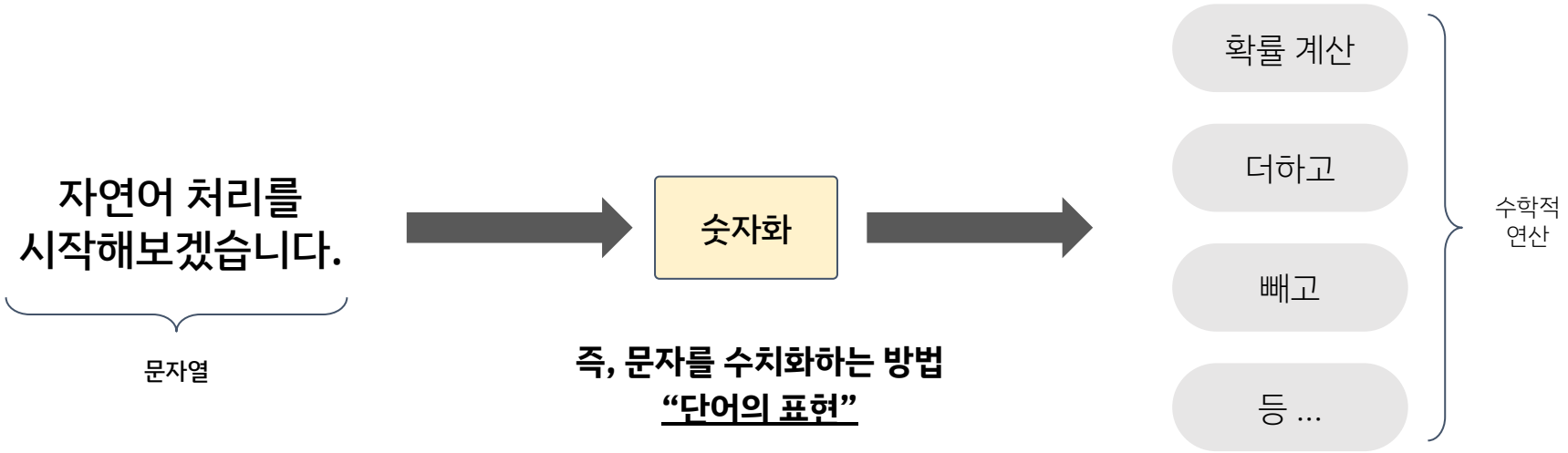
문자열

가능할까요?

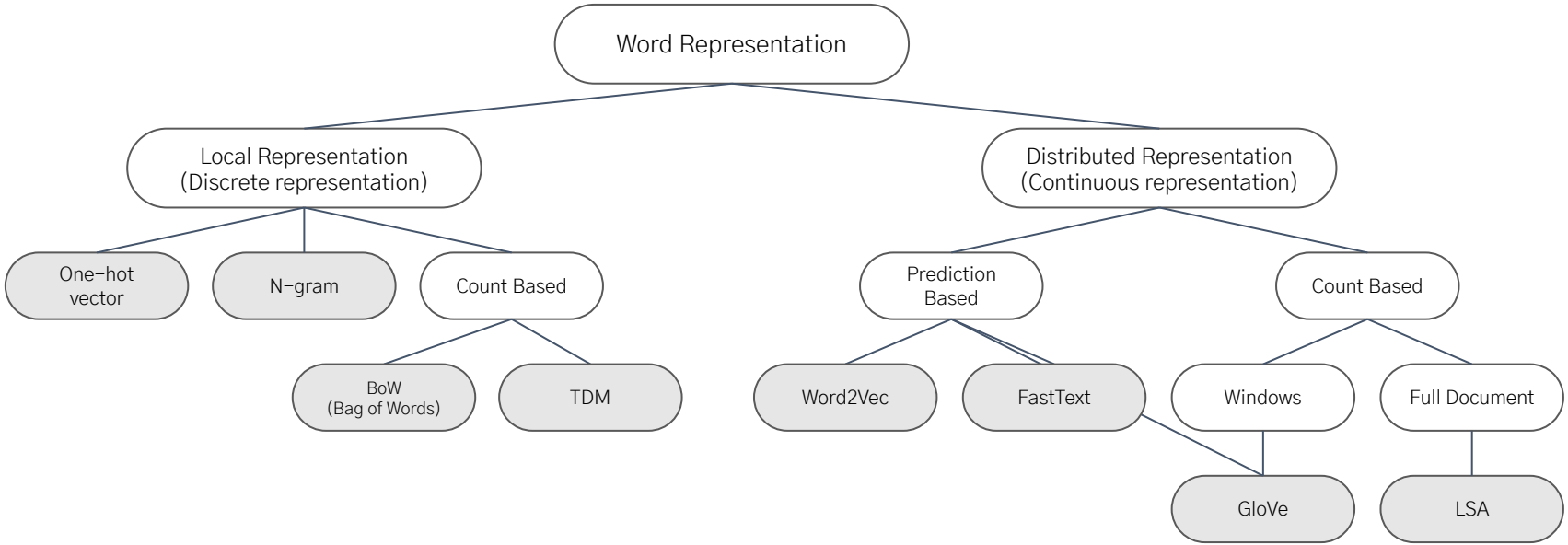
- 확률 계산
- 더하고
- 빼고
- 등 ...

수학적 연산

단어의 표현이 필요한 이유



단어의 표현의 분류



Local Representation

- Local Representation = Discrete Representation
- 표현하고자 하는 단어만을 보고 수치화하여 표현

ex) 원숭이 코뿔소 바나나 오렌지

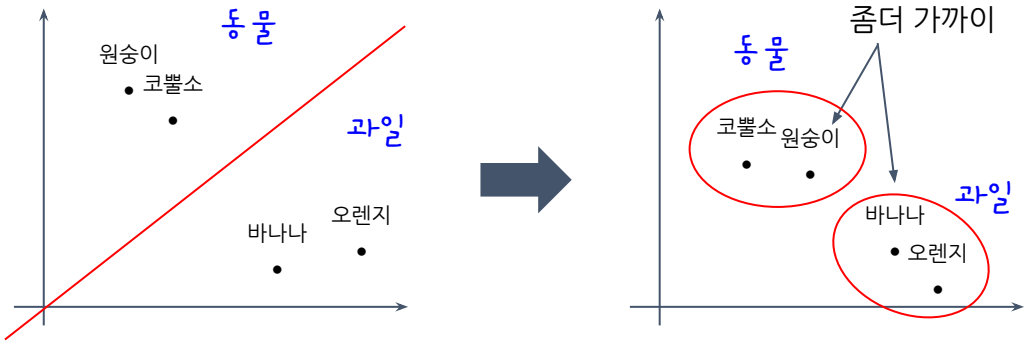
원숭이 코뿔소 바나나 오렌지 = [0, 1, 2, 3]

입력한 단어를 인덱스로 표현

Distributed Representation

- Distributed Representation = Continuous Representation
- 단어를 표현할때, 표현하고자 하는 단어만 보는 것이 아니라 주변단어를 함께 참고하여 수치화

ex) 원숭이 코뿔소 바나나 오렌지



Local Representation

Local Representation

One-hot vector

Ngram

Count based method

- BOW(Bag of Words)
- DTM(Document Term Matrix) & TDM(Term Document Matrix)
- TF-IDF(Term Frequency-Inverse Document Frequency)

One-hot encoding

- 표현하고자하는 단어의 갯수를 벡터차원으로하여 표현하고자 하는 단어의 인덱스에는 ‘1’ 다른 인덱스에는 ‘0’을 부여하여 표현하는 벡터표현방식.

ex) 원숭이 코뿔소 바나나 오렌지 각각을 one-hot encoding으로 표현하면?

원숭이 = [1, 0, 0]
코뿔소 = [0, 1, 0]
바나나 = [0, 0, 1]

> 이렇게 표현하는 방법을 one-hot encoding이라 하고, 각각의 벡터를 one-hot vector

One-hot encoding의 한계

- 차원의 문제

- 벡터가 단어의 의미를 담지 못한다.

Ngram

- n개의 연속된 단어나열

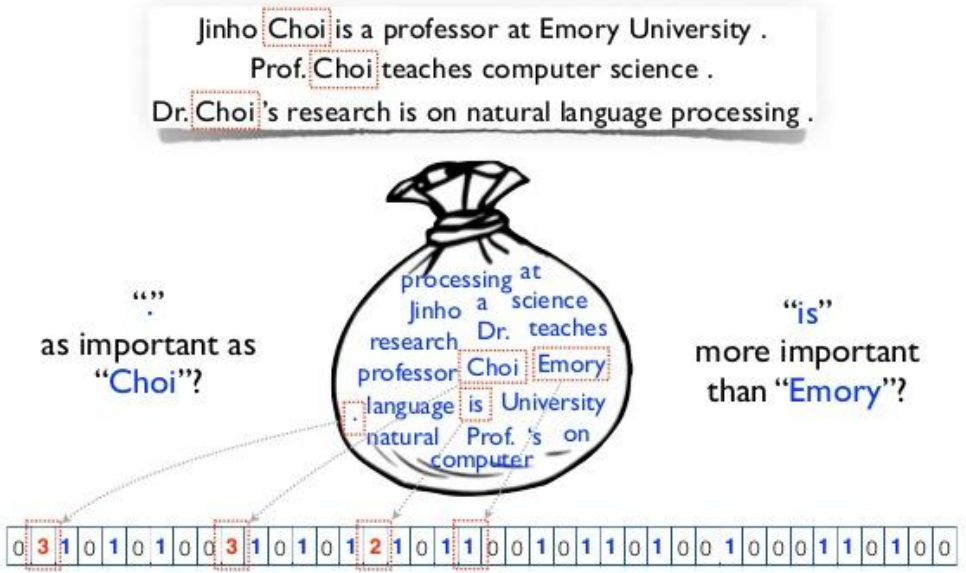
  ex) 나는 오늘 동물원에서 원숭이를 봤어.

  bigram : 나는;오늘, 오늘;동물원에서, 동물원에서;원숭이를, 원숭이를;봤어

  trigram : 나는;오늘;동물원에서, 오늘;동물원에서;원숭이를, 동물원에서; 원숭이를;봤어

Bag of Words

- 문서 내 단어의 출현 순서를 무시한 채, 출현 빈도수만으로 단어를 표현하는 방법



BoW 생성 방법

문서1: 오늘 동물원에서 코끼리 원숭이를 보고 코끼리 원숭이에게 먹이를 줬어
문서2:오늘 동물원에서 원숭이에게 사과를 줬어

Step1. 각 토큰에 고유 인덱스 부여

오늘	0
동물원에서	1
코끼리	2
원숭이를	3
보고	4
원숭이에게	5
먹이를	6
줬어	7
사과를	8

Step2. 각 인덱스 위치에 토큰 등장 횟수를 기록

	오늘	동물원에서	코끼리	원숭이를	보고	원숭이에게	먹이를	줬어	사과를
문서1	1	1	2	1	1	2	1	2	0

	오늘	동물원에서	코끼리	원숭이를	보고	원숭이에게	먹이를	줬어	사과를
문서2	1	1	0	0	0	1	0	1	1

실습 1 - BOW 직접구현하기

```
docs = ['오늘 동물원에서 코끼리 원숭이를 보고 코끼리 원숭이에게 먹이를 줬어',
        '오늘 동물원에서 원숭이에게 사과를 줬어']
```

다음 문서를 공백으로 토큰화하고 BOW 를 직접 구현해보세요

실습 2 - 단어의 순서를 고려하지 않은 BOW

```
docs = ['나는 양념 치킨을 좋아해 하지만 후라이드 치킨을 싫어해',
        '나는 후라이드 치킨을 좋아해 하지만 양념 치킨을 싫어해']
```

다음 문서를 공백으로 토큰화하고 BOW 를 직접 구현해보세요 .
그리고, 위의 두 문장을 BOW로 표현했을때 단점이 무엇인지 생각해보세요.

실습 3 - sklearn 으로 구현

```
docs = ['오늘 동물원에서 코끼리 원숭이를 보고 코끼리 원숭이에게 먹이를 줬어',
        '오늘 동물원에서 원숭이에게 사과를 줬어']
```

다음 문서를 sklearn 의 CountVectorizer를 사용해 BOW를 생성해 보고,
실습 1에서 직접구현한것과 그 결과를 비교해 보세요.

한계

- 단어의 **순서를 고려 하지 않는다.**

- **BoW 는 Spare** 하다. 즉, 벡터 공간의 낭비, 연산 비효율성 초래한다.

- **단어 빈도수가 중요도를 바로 의미 하지 않는다.**
  단어가 자주 등장한다고 중요한 단어는 아니겠죠?

- 전처리가 매우 중요하다. 같은 의미의 다른 단어 표현이 있을 경우 다른것으로 인식될 수 있음.
  (뉴스와 같이 정제된 어휘를 사용하는 매체는 좋으나, 소셜에서는 활용하기 어려움)

DTM과 TDM
(Document Term Matrix, Term Document Matrix)

- Bag of Words의 방법 중 하나
- 문서에 등장하는 각 단어의 등장빈도를 행렬로 표현

	단어1	단어2	단어3	...	단어n
문서1	0	0	0	0	0
문서2	1	1	0	0	0
문서3	1	0	0	0	0
...	0	0	3	1	1
문서n	0	0	0	1	0

Document Term Matrix(DTM)

	문서1	문서2	문서3	...	문서n
단어1	0	1	1	0	0
단어2	0	1	0	0	0
단어3	0	0	0	3	0
...	0	0	0	1	1
단어n	0	0	0	1	0

Term Document Matrix(TDM)

DTM (Document-Term Matrix)

문서1: 동물원 코끼리
문서2: 동물원 원숭이 바나나
문서3: 엄마 코끼리 아기 코끼리
문서4: 원숭이 바나나 코끼리 바나나

	동물원	코끼리	원숭이	바나나	엄마	아기
문서1	1	1	0	0	0	0
문서2	1	0	1	1	0	0
문서3	0	2	0	0	1	1
문서4	0	1	1	2	0	0

실습 1 - TDM 직접구현하기

```
docs = ['동물원 코끼리',
        '동물원 원숭이 바나나',
        '엄마 코끼리 아기 코끼리',
        '원숭이 바나나 코끼리 바나나']
```

다음 문서를 공백으로 토큰화하고 TDM을 직접 구현해보세요

실습 2 - sklearn 으로 구현

```
docs = ['동물원 코끼리',
        '동물원 원숭이 바나나',
        '엄마 코끼리 아기 코끼리',
        '원숭이 바나나 코끼리 바나나']
```

다음 문서를 sklearn 의 CountVectorizer를 사용해 TDM를 생성해 보고,
실습 1에서 직접구현한것과 그 결과를 비교해 보세요.

TDM (Term-Document Matrix) 의 한계

- 단어의 **순서를 고려 하지 않는다.**

- **TDM 는 Spare** 하다. 즉, 벡터 공간의 낭비, 연산 비효율성 초래한다.

- **단어 빈도수가 중요도를 바로 의미 하지 않는다.**

  ex) the, a 와 같이 빈번하게 등장하는 관사를 생각해보자.
  이러한 관사는 모든 문서에 빈번하게 등장하므로 TDM에서 중요한 단어로 판단 될 수 있다.
  => 이를 보완 하기 위하여 TF - IDF 를 사용한다.

TF-IDF

(Term Frequency-Inverse Document Frequency)

- (단어빈도-역문서빈도) 라고 생각할 수 있다.
- 즉, 문서집합에서 단어와 문서의 관련성을 평가하는 방법으로 TF(단어빈도)와 IDF(역문서빈도)를 곱하여 계산한다.
- 이렇게 하면 단어의 상대적 중요도를 계산할 수 있다.

<TF-IDF 계산 수식>

$$tfidf(t,d,D) = tf(t,d) * idf(t,D)$$

TF

- TF(Term Frequency) : 단어빈도
- 한 문서 내에서 단어가 얼마나 많이 등장 했는가
- 단어가 문서내에 많이 등장할 수록 높은 TF값을 가진다.

  때문에 자주 등장할 수록 중요한 단어라는 의미를 가지게 된다.

일반적인 TF 구하는 방법

$$TF(t,d) = \frac{f_{t,d}}{\sum f_{t',d}} = \frac{\text{특정단어등장빈도}}{\text{문서내전체등장단어빈도}}$$

TF 가중치 계산하는 여러방법

Variants of term frequency (tf) weight

weighting scheme	tf weight
binary	$0, 1$
raw count	$f_{t,d}$
term frequency	$f_{t,d} \Big/ \sum_{t' \in d} f_{t',d}$
log normalization	$\log(1 + f_{t,d})$
double normalization 0.5	$0.5 + 0.5 \cdot \frac{f_{t,d}}{\max_{\{t' \in d\}} f_{t',d}}$
double normalization K	$K + (1 - K)\frac{f_{t,d}}{\max_{\{t' \in d\}} f_{t',d}}$

출처 : https://en.wikipedia.org/wiki/Tf%E2%80%93idf

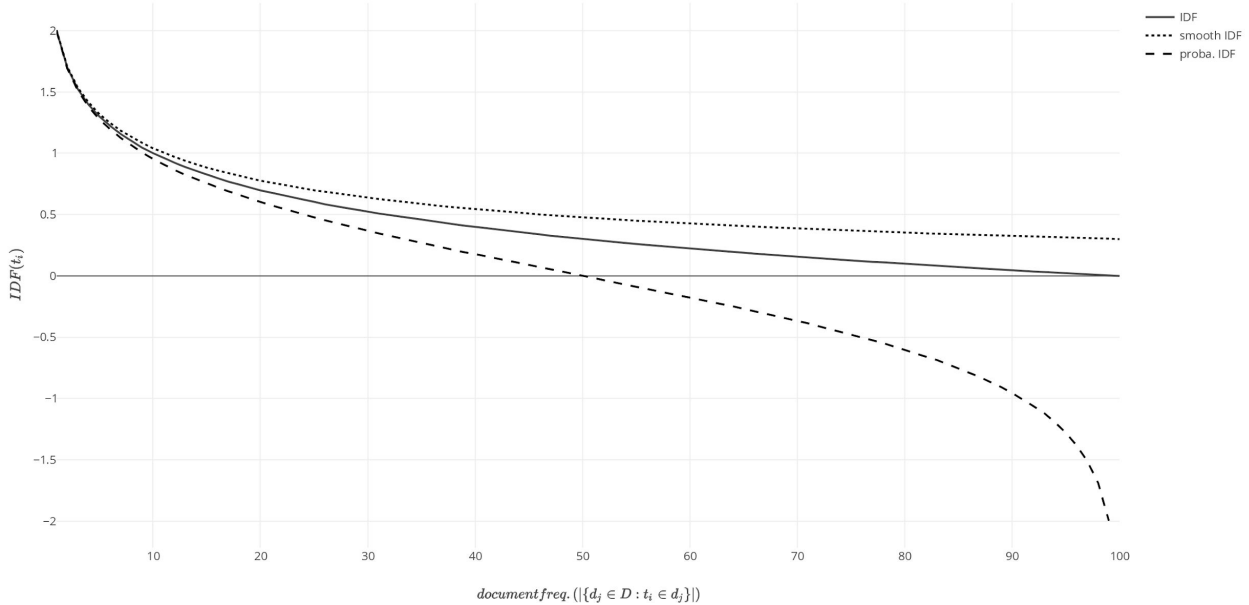
IDF

- IDF(Inverse Document Frequency) : 역문서 빈도
- 단어가 얼마나 많은 문서에 등장했는지를 나타내는 DF의 역수
- 단어가 여러문서에 많이 등장 할 수록 IDF는 작아진다.

일반적인 IDF 구하는 방법

$$IDF(t, D) = log\frac{N}{n_t} = -log\frac{n_t}{N} = log\frac{\text{총문서수}}{\text{단어가등장한문서수}}$$

IDF에 로그를 사용하는 이유



출처 : https://en.wikipedia.org/wiki/Tf%E2%80%93idf

IDF 가중치 계산하는 여러가지 방법

Variants of inverse document frequency (idf) weight

weighting scheme	idf weight ($n_t = \left\|\{d \in D : t \in d\}\right\|$)
unary	1
inverse document frequency	$\log \frac{N}{n_t} = -\log \frac{n_t}{N}$
inverse document frequency smooth	$\log\left(\frac{N}{1+n_t}\right)$
inverse document frequency max	$\log\left(\frac{\max_{\{t' \in d\}} n_{t'}}{1+n_t}\right)$
probabilistic inverse document frequency	$\log \frac{N-n_t}{n_t}$

출처 : https://en.wikipedia.org/wiki/Tf%E2%80%93idf

TF-IDF

- TDM 내 각 단어의 중요성을 가중치로 표현한 방법
- TDM을 사용하는 것보다 더 정확하게 문서비교가 가능하다.

$$tfidf(t, d, D) = tf(t, d) * idf(t, D)$$

tf(d,t)	특정 문서 d에서의 특정 단어 t의 등장 횟수
df(t)	특정 단어 t가 등장한 문서의 수
idf(d, t)	df(t)의 역수

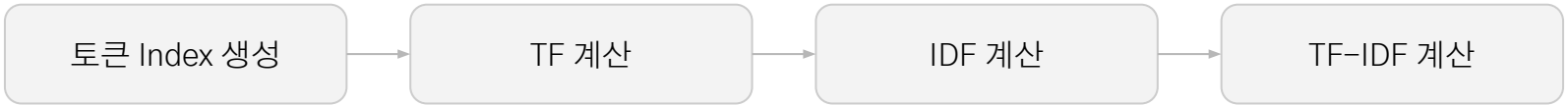
TF-IDF

$$tfidf(t, d, D) = tf(t, d) * idf(t, D)$$

tf(d,t)	특정 문서 d에서의 특정 단어 t의 등장 횟수
df(t)	특정 단어 t가 등장한 문서의 수
idf(d, t)	df(t)의 역수

TF	IDF	TF-IDF	설명
높	높	높	특정 문서에 많이 등장하고 타 문서에 많이 등장하지 않는 단어 (중요 키워드)
높	낮	-	특정 문서에도 많이 등장하고 타 문서에도 많이 등장하는 단어
낮	높	-	특정 문서에 많이 등장하지 않고 타 문서에만 많이 등장하지 않는 단어
낮	낮	낮	특정 문서에 많이 등장하지 않고 타 문서에만 많이 등장하는 단어

TF-IDF 계산절차



각 토큰(단어)을 그대로 사용할 수 없기 때문에 토큰에 Index를 부여

각 토큰(단어)의 등장 빈도를 계산

각 토큰(단어)의 문서 등장 빈도를 계산하여 역수를 취함

계산한 TF와 IDF를 곱하여 TF-IDF를 계산

TF-IDF 예제

문서1 : d1 = 오늘 동물원에서 원숭이와 코끼리를 봤어
문서2 : d2 = 동물원에서 원숭이에게 바나나를 줬어 바나나를

	Index
오늘	0
동물원에서	1
원숭이와	2
코끼리를	3
봤어	4
원숭이에게	5
바나나를	6
줬어	7

다음과 같이 문서1, 문서2가 주어지고 문서들을 공백으로 토큰화하고, 토큰별 인덱스를 부여하면 왼쪽과 같습니다.

이때 각 문서1, 문서2의 TF-IDF 가중치는 각각 얼마이고, 문서에서 중요단어는 무엇일까요?

예제 : TF 계산

문서1 : d1 = 오늘 동물원에서 원숭이와 코끼리를 봤어
문서2 : d2 = 동물원에서 원숭이에게 바나나를 줬어 바나나를

$$TF(t,d) = \frac{f_{t,d}}{\sum f_{t',d}} = \frac{\text{특정단어등장빈도}}{\text{문서내전체등장단어빈도}}$$

문서1

	문서내 토큰 빈도	문서내 전체 토큰빈도	TF
오늘	1	5	0.2
동물원에서	1	5	0.2
원숭이와	1	5	0.2
코끼리를	1	5	0.2
봤어	1	5	0.2
원숭이에게	0	5	0
바나나를	0	5	0
줬어	0	5	0

문서2

	문서내 토큰 빈도	문서내 전체 토큰빈도	TF
오늘	0	5	0
동물원에서	1	5	0.2
원숭이와	0	5	0
코끼리를	0	5	0
봤어	0	5	0
원숭이에게	1	5	0.2
바나나를	2	5	0.4
줬어	1	5	0.2

예제 : IDF 계산

문서1 : d1 = 오늘 동물원에서 원숭이와 코끼리를 봤어
문서2 : d2 = 동물원에서 원숭이에게 바나나를 줬어 바나나를

$$IDF(t, D) = log\frac{N}{n_t} = -log\frac{n_t}{N} = log\frac{\text{총문서수}}{\text{단어가등장한문서수}}$$

	문서수	토큰이 등장한 문서수	IDF
오늘	2	1	0.301
동물원에서	2	2	0
원숭이와	2	1	0.301
코끼리를	2	1	0.301
봤어	2	1	0.301
원숭이에게	2	1	0.301
바나나를	2	1	0.301
줬어	2	1	0.301

예제 : TF-IDF 계산

문서1 : d1 = 오늘 동물원에서 원숭이와 코끼리를 봤어
문서2 : d2 = 동물원에서 원숭이에게 바나나를 줬어 바나나를

문서1

	TF	IDF	TF-IDF
오늘	0.2	0.301	0.0602
동물원에서	0.2	0	0
원숭이와	0.2	0.301	0.0602
코끼리를	0.2	0.301	0.0602
봤어	0.2	0.301	0.0602
원숭이에게	0	0.301	0
바나나를	0	0.301	0
줬어	0	0.301	0

문서2

	TF	IDF	TF-IDF
오늘	0	0.301	0
동물원에서	0.2	0	0
원숭이와	0	0.301	0
코끼리를	0	0.301	0
봤어	0	0.301	0
원숭이에게	0.2	0.301	0.0602
바나나를	0.4	0.301	0.1204
줬어	0.2	0.301	0.0602

예제 : TF-IDF 계산

문서1 : d1 = 오늘 동물원에서 원숭이와 코끼리를 봤어
문서2 : d2 = 동물원에서 원숭이에게 바나나를 줬어 바나나를

문서1

	TF	IDF	TF-IDF
오늘	0.2	0.301	0.0602
동물원에서	0.2	0	0
원숭이와	0.2	0.301	0.0602
코끼리를	0.2	0.301	0.0602
봤어	0.2	0.301	0.0602
원숭이에게	0	0.301	0
바나나를	0	0.301	0
줬어	0	0.301	0

문서2

	TF	IDF	TF-IDF
오늘	0	0.301	0
동물원에서	0.2	0	0
원숭이와	0	0.301	0
코끼리를	0	0.301	0
봤어	0	0.301	0
원숭이에게	0.2	0.301	0.0602
바나나를	0.4	0.301	0.1204
줬어	0.2	0.301	0.0602

문서1의 중요 단어는 '오늘', '원숭이와', '코끼리를', '봤어'
문서2의 중요 단어는 '바나나를'

실습 1 - TF-IDF 직접구현하기

```
docs = ['오늘 동물원에서 원숭이와 코끼리를 봤어',
'동물원에서 원숭이에게 바나나를 줬어 바나나를']
```

다음 문서를 공백으로 토큰화하고 TF-IDF를 직접 구현해보세요

실습 2 - sklearn 으로 구현

```
docs = ['오늘 동물원에서 원숭이와 코끼리를 봤어',
'동물원에서 원숭이에게 바나나를 줬어 바나나를']
```

다음 문서를 sklearn 의 TfidfVectorizer를 사용해 TF-IDF를 생성해 보고,
실습 1에서 직접구현한것과 그 결과를 비교해 보세요.