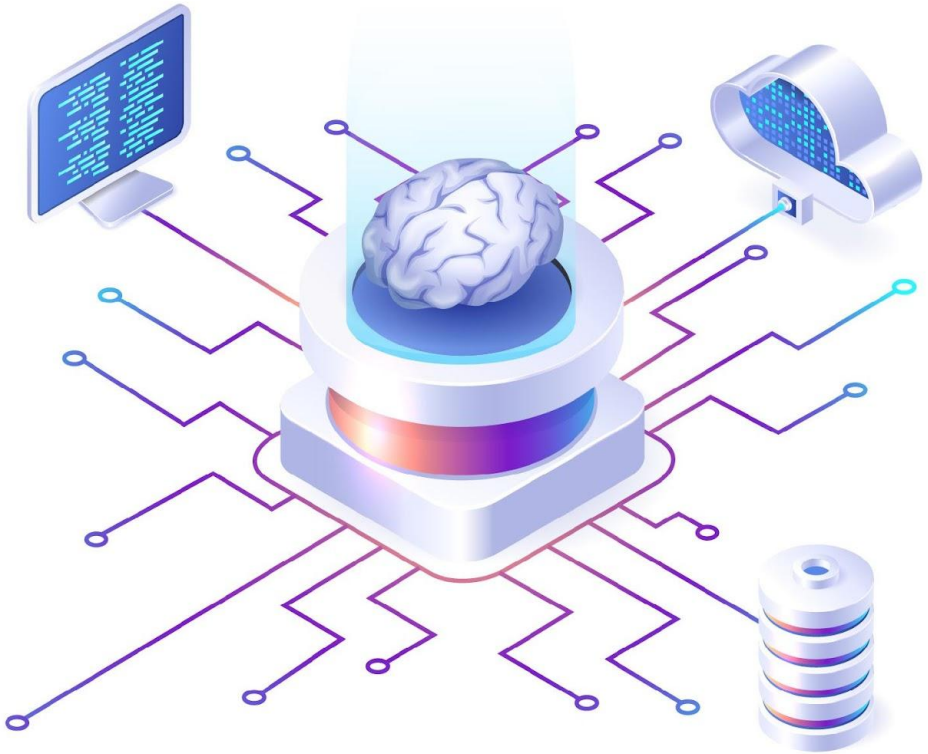
Insight campus SeSAC

표현(Representation)

실무형 인공지능 자연어 처리



4

TF-IDF (단어빈도-역문서빈도)

Term Frequency-Inverse Document Frequency

TF-IDF (Term Frequency-Inverse Document Frequency)

- 단어 빈도 - 역문서 빈도
- TDM 내 각 단어의 중요성을 가중치로 표현
- TDM을 사용하는 것보다 더 정확하게 문서비교가 가능

$$\text{tfidf}(t, d, D) = \text{tf}(t, d) \cdot \text{idf}(t, D)$$

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

Insight campus SeSAC

TF-IDF

$$\text{tfidf}(t, d, D) = \text{tf}(t, d) \cdot \text{idf}(t, D)$$

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 문서에는 많이 등장하지 않고 타 문서에만 많이 등장하는 단어
낮	낮	낮	특정 문서에 많이 등장하지 않고 타 문서에만 많이 등장하는 단어

TF-IDF 가중치 계산

Variants of term frequency (tf) weight

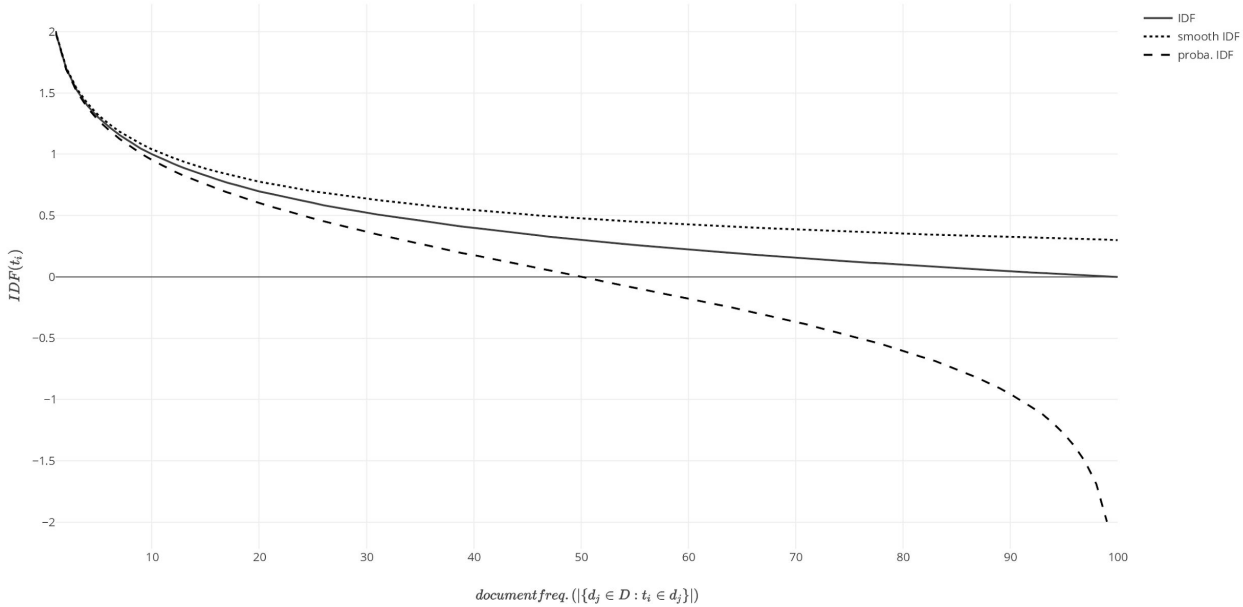
weighting scheme	tf weight
binary	$0, 1$
raw count	$f_{t,d}$
term frequency	$f_{t,d} \Big/ \sum_{t' \in d} f_{t',d}$
log normalization	$\log(1 + f_{t,d})$
double normalization 0.5	$0.5 + 0.5 \cdot \frac{f_{t,d}}{\max_{\{t' \in d\}} f_{t',d}}$
double normalization K	$K + (1 - K)\frac{f_{t,d}}{\max_{\{t'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

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

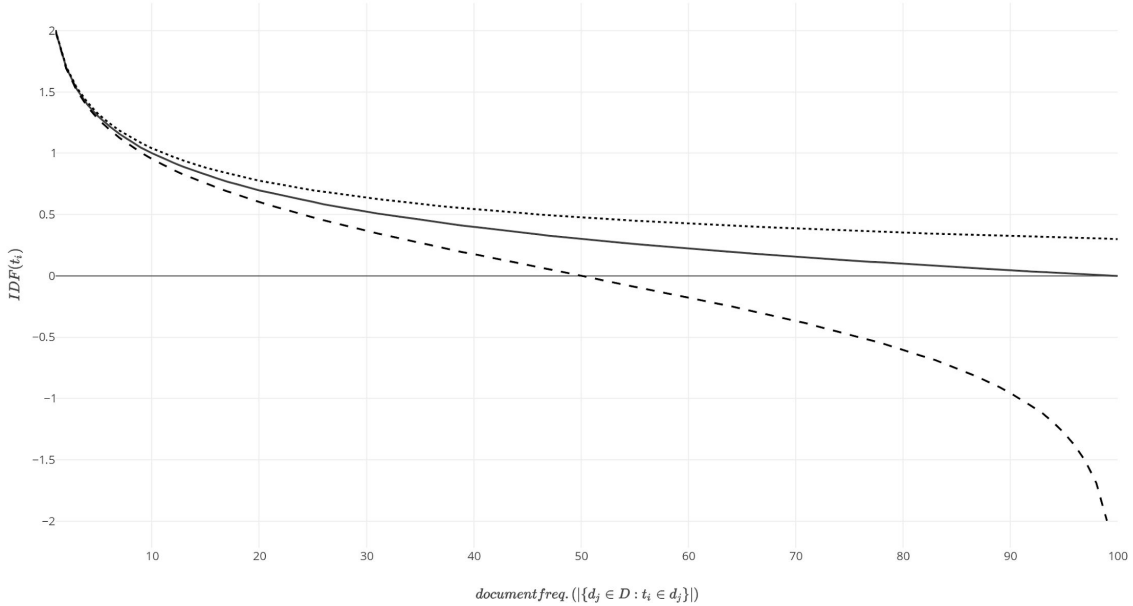
IDF에 로그를 사용하는 이유



총문서수	10,000	
	단어A	단어B
Freq	8	9
DF	0.0008	0.0009
IDF	7.1309	7.0131
변화율	1.65%	
	단어C	단어D
Freq	8000	8001
DF	0.8	0.8001
IDF	0.2231	0.2230
변화율	0.06%	

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

IDF에 로그를 사용하는 이유



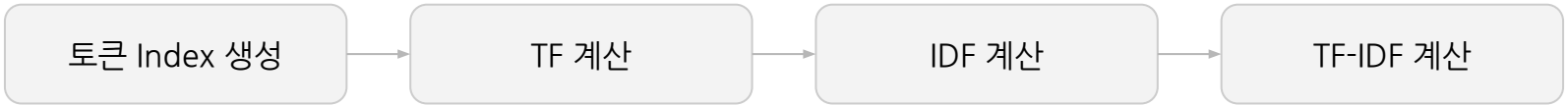
$$\text{idf} = \log\left(\frac{N}{n_t}\right) = -\log\left(\frac{n_t}{N}\right)$$

$$\text{smooth idf} = \log\left(\frac{N}{1+n_t}\right) + 1$$

$$\text{probabilistic idf} = \log\left(\frac{N-n_t}{n_t}\right)$$

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

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

예제 1 : 토큰 Index 생성

문서1 : d1 = "The cat sat on my face I hate a cat"
문서2 : d2 = "The dog sat on my bed I love a dog"

	Index
The	0
cat	1
sat	2
on	3
my	4
face	5
I	6
hate	7
a	8
dog	9
bed	10
lov	11

Insight campus SeSAC

예제 : TF 계산

문서1 : d1 = "The cat sat on my face I hate a cat"
문서2 : d2 = "The dog sat on my bed I love a dog"

$$f_{t,d} \Big/ \sum_{t' \in d} f_{t',d}$$

$f_{t,d}$ = 문서내 토큰 빈도
SUM($f_{t,d}$)= 문서내 전체 토큰빈도

문서1

	문서내 토큰 빈도	문서내 전체 토큰빈도	TF
The	1	10	0.1
cat	2	10	0.2
sat	1	10	0.1
on	1	10	0.1
my	1	10	0.1
face	1	10	0.1
I	1	10	0.1
hate	1	10	0.1
a	1	10	0.1
dog	0	10	0
bed	0	10	0
lov	0	10	0

문서2

	문서내 토큰 빈도	문서내 전체 토큰빈도	TF
The	1	10	0.1
cat	0	10	0
sat	1	10	0.1
on	1	10	0.1
my	1	10	0.1
face	0	10	0
I	1	10	0.1
hate	0	10	0
a	1	10	0.1
dog	2	10	0.2
bed	1	10	0.1
love	1	10	0.1

예제 : IDF 계산

문서1 : d1 = "The cat sat on my face I hate a cat"
문서2 : d2 = "The dog sat on my bed I love a dog"

$$\log \frac{N}{n_t} = -\log \frac{n_t}{N}$$

N = 문서수

n_t = 토큰이 등장한 문서수

	문서수	토큰이 등장한 문서수	IDF
The	2	2	0
cat	2	1	0.301
sat	2	2	0
on	2	2	0
my	2	2	0
face	2	1	0.301
I	2	2	0
hate	2	1	0.301
a	2	2	0
dog	2	1	0.301
bed	2	1	0.301
love	2	1	0.301

Insight campus SeSAC

예제 : TF-IDF 계산

문서1 : d1 = "The cat sat on my face I hate a cat"
문서2 : d2 = "The dog sat on my bed I love a dog"

문서1

	TF	IDF	TF-IDF
The	0.1	0	0
cat	0.2	0.301	0.060
sat	0.1	0	0
on	0.1	0	0
my	0.1	0	0
face	0.1	0.301	0.301
I	0.1	0	0
hate	0.1	0.301	0.301
a	0.1	0	0
dog	0	0.301	0.301
bed	0	0.301	0.301
lov	0	0.301	0.301

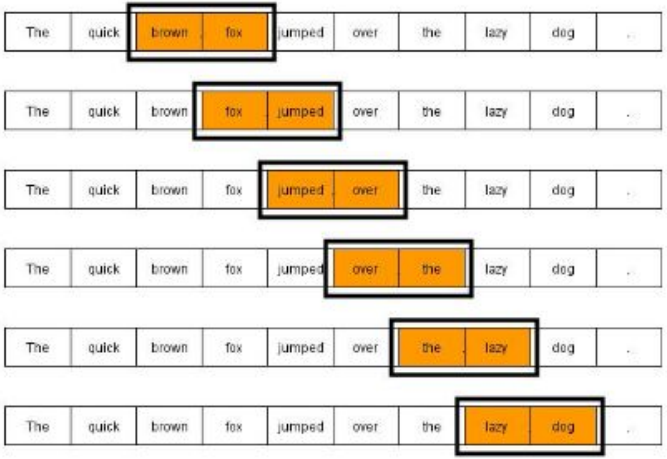
문서2

	TF	IDF	TF-IDF
The	0.1	0	0
cat	0	0.301	0
sat	0.1	0	0
on	0.1	0	0
my	0.1	0	0
face	0	0.301	0.301
I	0.1	0	0
hate	0	0.301	0.301
a	0.1	0	0
dog	0.2	0.301	0.601
bed	0.1	0.301	0.301
love	0.1	0.301	0.301

5 n-Gram

n-Gram 이란?

- 복수개(n개) 단어를 보는냐에 따라 unigram, bigram, trigram 등 으로 구분
- 제한적으로 문맥을 표현할 수 있음



n-Gram 이란?

an adorable little boy is spreading smile

- unigrams : an, adorable, little, boy, is, spreading, smiles
- bigrams : an adorable, adorable little, little boy, boy is, is spreading, spreading smiles
- trigrams : an adorable little, adorable little boy, little boy is, boy is spreading, is spreading smiles
- 4-grams : an adorable little boy, adorable little boy is, little boy is spreading, boy is spreading smiles

n-Gram 한계

- n의 크기는 trade-off 문제
  - 1보다는 2를 선택하는 것이 대부분 언어 모델 성능을 높일 수 있음
  - n을 너무 크게 선택하면 n-gram 이 unique 할 확률이 높아 등장수가 낮을 확률이 높음. (OOV, Out of Vocabulary 문제가 발생할 수 있음)
  - n을 너무 작게 하면 카운트는 잘되지만 정확도가 떨어질 수 있음. n은 최대 5를 넘지 않도록 권장

-	Unigram	Bigram	Trigram
Perplexity	962	170	109

스탠포드에 3,800만개 단어 토큰을 n-Gram으로 학습한 결과

- n-Gram 카운트가 0인 경우
  - n-Gram 이 모든 단어를 커버 할 수 없기 때문에 Out of Vocabulary 문제가 발생할수 있음

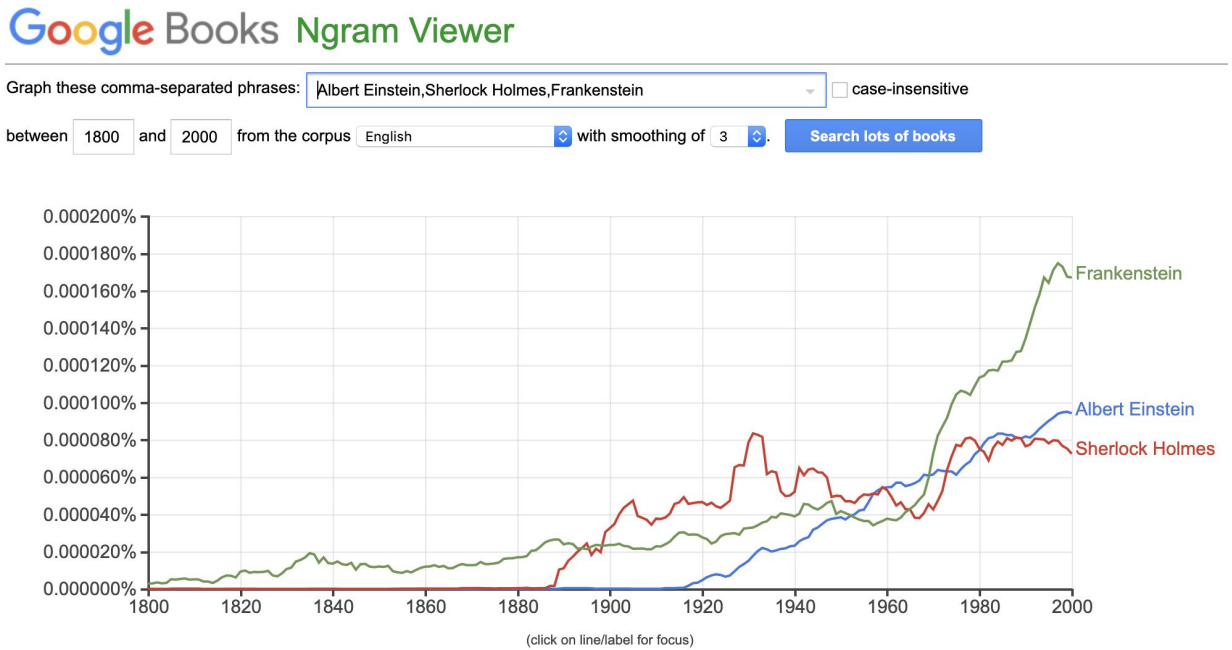
적용 분야(Domain)에 맞는 코퍼스의 수집

- 분야(Domain)에 따라 단어들의 확률 분포는 다름
(금융 분야는 금융 관련 용어가 많이 등장하고, 마케팅은 관련 용어가 많이 등장할 것임)

- 분야에 적합한 코퍼스를 사용하면 언어 모델의 성능이 높아질 수 있음
(훈련에 사용되는 코퍼스에 따라 언어 모델의 성능이 달라짐 이는 언어 모델의 약점으로 분류되기도함)

Insight campus SeSAC

Google Books Ngram Viewer



https://books.google.com

감사합니다.

Insight campus SeSAC

daniel@fins.ai