LSA, LDA

잠재 의미 분석(Latent Semantic Analysis, LSA)

Bag of words 기반의 DTM(document-term matrix), TF-IDF는 단어의 의미를 고려하지 못한다는 단점이 있음

→ LSA: DTM의 잠재된(Latent) 의미를 이끌어내는 방법

특이값 분해(Singular Value Decomposition, SVD)

$$A = U\Sigma V^{\mathrm{T}}$$

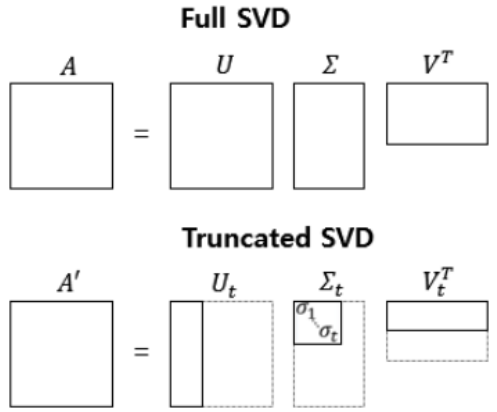
$U : m \times m$ 직교행렬 $(AA^{\mathrm{T}} = U(\Sigma\Sigma^{\mathrm{T}})U^{\mathrm{T}})$
$V : n \times n$ 직교행렬 $(A^{\mathrm{T}}A = V(\Sigma^{\mathrm{T}}\Sigma)V^{\mathrm{T}})$
$\Sigma : m \times n$ 직사각 대각행렬

$$\Sigma = \begin{bmatrix} 12.4 & 0 & 0 \\ 0 & 9.5 & 0 \\ 0 & 0 & 1.3 \end{bmatrix}$$

대각의 숫자는 행렬 A의 특잇값(singular value)

차원 축소, 노이즈 제거, 토픽 모델링 등 분야에서 활용

절단된 SVD(Truncated SVD):
- 대각 행렬 Σ의 대각 원소의 값 중에서 상위값 t개만 남김, U행렬과 V행렬의 t열까지만 남김
- 계산 비용이 낮아지는 것 외에도 상대적으로 중요하지 않은 정보를 삭제하는 효과
- t는 토픽의 수



잠재 의미 분석(Latent Semantic Analysis, LSA)

LSA: DTM이나 **TF-IDF 행렬**에 **truncated SVD** 사용해 차원을 축소

과일이	길고	노란	먹고	바나나	사과	싶은	저는	좋아요
문서1	0	0	0	1	0	1	1	0
문서2	0	0	0	1	1	0	1	0
문서3	0	1	1	0	2	0	0	0
문서4	1	0	0	0	0	0	0	1

$$A = U\Sigma V^{\mathrm{T}}$$

예시 (A는 dtm 행렬)

```
행렬 U :
[[-0.24  0.75  0.   -0.62]
 [-0.51  0.44 -0.    0.74]
 [-0.83 -0.49 -0.   -0.27]
 [-0.   -0.    1.    0.  ]]
```

```
특이값 벡터 :
[2.69 2.05 1.73 0.77]
```

```
직교행렬 VT :
[[-0.   -0.31 -0.31 -0.28 -0.8  -0.09 -0.28 -0.   -0.  ]
 [ 0.    0.24  0.24  0.58 -0.26  0.37  0.58 -0.   -0.  ]
 [ 0.58 -0.    0.    0.   -0.    0.   -0.    0.58  0.58]
 [ 0.   -0.35 -0.35  0.16  0.25 -0.8   0.16 -0.   -0.  ]
 [-0.   -0.78 -0.01 -0.2   0.4   0.4  -0.2   0.    0.  ]
 [-0.29  0.31 -0.78 -0.24  0.23  0.23  0.01  0.14  0.14]
 [-0.29 -0.1   0.26 -0.59 -0.08 -0.08  0.66  0.14  0.14]
 [-0.5  -0.06  0.15  0.24 -0.05 -0.05 -0.19  0.75 -0.25]
 [-0.5  -0.06  0.15  0.24 -0.05 -0.05 -0.19 -0.25  0.75]]
```

- **축소된 U (4 × 2)**는 문서의 개수 × 토픽의 수 t의 크기

U의 각 행은 잠재 의미를 표현하기 위해 수치화 된 각각의 **문서 벡터**

- **축소된 V^T (2 × 9)**는 토픽의 수 t × 단어의 개수의 크기

V^T 의 각 열은 잠재 의미를 표현하기 위해 수치화 된 각각의 **단어 벡터**

이 문서 벡터들과 단어 벡터들을 통해 **다른 문서의 유사도, 다른 단어의 유사도, 단어(쿼리)로부터 문서의 유사도**를 구하는 것들이 가능

잠재 의미 분석(Latent Semantic Analysis, LSA)

토픽 모델링 예시 축소 차원에서 근접 단어들을 토픽으로 묶음

```
vectorizer = TfidfVectorizer(stop_words='english', max_features= 1000, # 상위 1,000개의 단어를
존
max_df = 0.5, smooth_idf=True)

X = vectorizer.fit_transform(news_df['clean_doc'])
```

```
TF-IDF 행렬의 크기 : (11314, 1000)
```

```
svd_model = TruncatedSVD(n_components=20, algorithm='randomized', n_iter=100, random_state=12
svd_model.fit(X)
np.shape(svd_model.components_)
```

```
(20, 1000)
```

```
terms = vectorizer.get_feature_names() # 단어 집합. 1,000개의 단어가 저장됨.

def get_topics(components, feature_names, n=5):
    for idx, topic in enumerate(components):
        print("Topic %d:" % (idx+1), [(feature_names[i], topic[i].round(5)) for i in topic.argsor
t()[:-n - 1:-1]])
get_topics(svd_model.components_,terms)
```

잠재 의미 분석(Latent Semantic Analysis, LSA)

토픽 모델링 예시

1. **문서-토픽 행렬 (U):**
   - 각 행은 원본 문서를 나타내고, 각 열은 하나의 토픽을 나타냅니다.
   - 이 행렬의 값은 해당 문서가 각 토픽과 얼마나 관련이 있는지를 나타내는 가중치입니다. 값이 클수록 문서가 그 토픽과 더 관련이 깊다는 것을 의미합니다.
2. **토픽의 중요도를 나타내는 대각 행렬 (Σ):**
   - 이 대각선에 위치한 값들은 각 토픽의 중요도 또는 "특이값"을 나타냅니다.
   - 값이 큰 토픽은 데이터 세트 전체에 걸쳐 더 많은 정보를 담고 있다고 해석할 수 있습니다. 이를 통해 가장 중요한 토픽을 식별할 수 있습니다.
3. **토픽-단어 행렬 (V^T):**
   - 각 행은 하나의 토픽을 나타내고, 각 열은 원본 데이터 세트의 단어를 나타냅니다.
   - 이 행렬의 값은 특정 단어가 토픽에 속하는 정도를 나타내는 가중치입니다. 값이 클수록 해당 단어는 그 토픽을 잘 대표한다는 의미입니다.

잠재 의미 분석(Latent Semantic Analysis, LSA)

토픽 모델링 예시

```
Topic 1: [('like', 0.2138), ('know', 0.20031), ('people', 0.19334), ('think', 0.17802), ('good',
0.15105)]
Topic 2: [('thanks', 0.32918), ('windows', 0.29093), ('card', 0.18016), ('drive', 0.1739), ('mai
l', 0.15131)]
Topic 3: [('game', 0.37159), ('team', 0.32533), ('year', 0.28205), ('games', 0.25416), ('season',
0.18464)]
Topic 4: [('drive', 0.52823), ('scsi', 0.20043), ('disk', 0.15518), ('hard', 0.15511), ('card',
0.14049)]
Topic 5: [('windows', 0.40544), ('file', 0.25619), ('window', 0.1806), ('files', 0.16196), ('prog
ram', 0.14009)]
```

잠재 의미 분석(Latent Semantic Analysis, LSA)

LSA는 쉽고 빠르게 구현이 가능할 뿐만 아니라 단어의 잠재적인 의미를 이끌어낼 수 있어 문서의 유사도 계산 등에서 좋은 성능을 보임

하지만 SVD의 특성상 이미 계산된 LSA에 새로운 데이터를 추가하여 계산하려고 하면 처음부터 다시 계산해야 함
즉, **새로운 정보에 대해 업데이트가 어려움**

→ Word2Vec 등 단어의 의미를 벡터화할 수 있는 또 다른 방법론인 인공 신경망 기반의 방법론 사용

잠재 디리클레 할당(Latent Dirichlet Allocation, LDA)

잠재 디리클레 할당(Latent Dirichlet Allocation, LDA)은 **토픽 모델링**의 대표적인 알고리즘

LDA는 문서들은 토픽들의 혼합으로 구성되어져 있으며, 토픽들은 **확률 분포**에 기반하여 단어들을 생성한다고 가정
LDA는 **각 문서의 토픽 분포**와 **각 토픽 내의 단어 분포를** 추정

데이터가 주어지면, LDA는 문서가 생성되던 과정을 역추적

LDA는 DTM 또는 TF-IDF 행렬을 입력으로 함. 즉, LDA는 단어의 순서는 신경쓰지 않음

문서1 : 저는 사과랑 바나나를 먹어요
문서2 : 우리는 귀여운 강아지가 좋아요
문서3 : 저의 깜찍하고 귀여운 강아지가 바나나를 먹어요

<각 문서의 토픽 분포>
문서1 : 토픽 A 100%
문서2 : 토픽 B 100%
문서3 : 토픽 B 60%, 토픽 A 40%

<각 토픽의 단어 분포>
토픽A : **사과 20%, 바나나 40%, 먹어요 40%**, 귀여운 0%, 강아지 0%, 깜찍하고 0%, 좋아요 0%
토픽B : 사과 0%, 바나나 0%, 먹어요 0%, **귀여운 33%, 강아지 33%, 깜찍하고 16%, 좋아요 16%**

잠재 디리클레 할당(Latent Dirichlet Allocation, LDA)

LDA 실행 순서

1. 사용자는 **토픽의 개수 k**를 설정
LDA는 토픽의 개수 k를 입력 받으면, k개의 토픽이 M개의 전체 문서에 걸쳐 분포되어 있다고 가정

2. 모든 단어를 k개 중 **하나의 토픽에 할당**
모든 문서의 모든 단어에 대해서 k개 중 하나의 토픽을 랜덤으로 할당
→ 각 문서는 토픽을 가지며, 토픽은 단어 분포를 가지는 상태

3. 이제 모든 문서의 모든 단어에 대해서 아래 사항을 **반복 진행(iterative)**

✓ 어떤 문서의 각 단어 w는 자신은 잘못된 토픽에 할당되어져 있지만, 다른 단어들은 전부 올바른 토픽에 할당되어져 있는 상태라고 가정

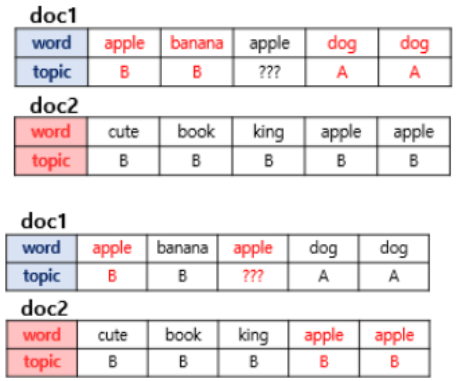
doc1

word	apple	banana	apple	dog	dog
topic	B	B	???	A	A

doc2

word	cute	book	king	apple	apple
topic	B	B	B	B	B

✓ 이에 따라 단어 w는 아래의 **두 가지 기준**에 따라서 토픽 재할당
- p(topic t | document d) : 문서 d의 단어들 중 토픽 t에 해당하는 단어들의 비율
- p(word w | topic t) : 각 토픽들 t에서 해당 단어 w의 분포



잠재 디리클레 할당(Latent Dirichlet Allocation, LDA)

```
import gensim
NUM_TOPICS = 20 # 20개의 토픽, k=20
ldamodel = gensim.models.ldamodel.LdaModel(corpus, num_topics = NUM_TOPICS, id2word=dictionary, p
asses=15)
topics = ldamodel.print_topics(num_words=4)
for topic in topics:
    print(topic)
```

Copy

passes는 알고리즘의 동작 횟수

```
(0, '0.015*"drive" + 0.014*"thanks" + 0.012*"card" + 0.012*"system"')
(1, '0.009*"back" + 0.009*"like" + 0.009*"time" + 0.008*"went"')
(2, '0.012*"colorado" + 0.010*"david" + 0.006*"decenso" + 0.005*"tyre"')
(3, '0.020*"number" + 0.018*"wire" + 0.013*"bits" + 0.013*"filename"')
(4, '0.038*"space" + 0.013*"nasa" + 0.011*"research" + 0.010*"medical"')
(5, '0.014*"price" + 0.010*"sale" + 0.009*"good" + 0.008*"shipping"')
(6, '0.012*"available" + 0.009*"file" + 0.009*"information" + 0.008*"version"')
(7, '0.021*"would" + 0.013*"think" + 0.012*"people" + 0.011*"like"')
(8, '0.035*"window" + 0.021*"display" + 0.017*"widget" + 0.013*"application"')
```

잠재 디리클레 할당(Latent Dirichlet Allocation, LDA)

	문서 번호	가장 비중이 높은 토픽	가장 높은 토픽의 비중	각 토픽의 비중
0	0	9.0	0.5071	[(7, 0.30498007), (9, 0.5071116), (11, 0.13196...
1	1	7.0	0.7634	[(7, 0.76344866), (13, 0.02931299), (14, 0.128...
2	2	7.0	0.5224	[(7, 0.522409), (9, 0.36602637), (16, 0.097610...
3	3	10.0	0.5854	[(1, 0.16931751), (5, 0.04911898), (6, 0.04034...
4	4	7.0	0.4215	[(7, 0.42152897), (12, 0.21915697), (17, 0.327...
5	5	7.0	0.3865	[(7, 0.38652688), (8, 0.1446223), (9, 0.244169...
6	6	7.0	0.3182	[(0, 0.30865452), (5, 0.2190474), (6, 0.059171...
7	7	1.0	0.3606	[(1, 0.36058533), (7, 0.34877154), (9, 0.16878...
8	8	14.0	0.5236	[(0, 0.21124822), (7, 0.18776271), (12, 0.0538...
9	9	7.0	0.4333	[(0, 0.0694689), (5, 0.37277842), (7, 0.433252...

LSA : DTM을 차원 축소하여 축소 차원에서 근접 단어들을 토픽으로 묶는다.
LDA : 단어가 특정 토픽에 존재할 확률과 문서에 특정 토픽이 존재할 확률을 결합확률로 추정하여 토픽을 추출한다.

참고자료

- 딥 러닝을 이용한 자연어 처리 입문(https://wikidocs.net/book/2155)