

문서 군집화를 위한 효율적인 k-means 활용:

빠른 학습, 군집 레이블링, 시각화

Hyunjoong Kim

soy.lovit@gmail.com

github.com/lovit

-
- 대량의 문서 집합을 몇 개의 토픽들로 요약하기 위하여 토픽 모델링이 이용될 수 있습니다.
 - 오늘의 주요 뉴스
 - 영화 리뷰들의 요약

-
- Latent Dirichlet Allocation (LDA) 은 대표적인 토픽 모델링 방법입니다.
 - LDA 는 문서 집합에 k 개의 토픽이 있다고 가정합니다.
 - 각 문서는 토픽이 얼마나 포함되어 있는지의 토픽 확률 벡터로 표현합니다.
 - 각 토픽은 어떤 단어들로 구성되어 있는지 단어 확률 벡터로 표현합니다.

-
- k-means 도 토픽 모델링을 위하여 이용될 수 있습니다.
 - 하나의 문서에 하나의 토픽이 존재하는 문서 집합도 많습니다.
 - 뉴스 기사, 맛집 리뷰
 - “k-means 토픽 모델링”은 문서 집합에 1 개의 토픽이 있다고 가정합니다.
 - 각 군집은 토픽입니다.
 - 각 문서는 하나의 토픽(군집)으로 구성됩니다.

k-means using scikit-learn

```
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.preprocessing import normalize
from sklearn.cluster import KMeans
```

```
# vectorizing
```

```
texts = ['예문입니다', 'list of str 형식입니다']
```

```
x = CountVectorizer().fit_transform(texts)
```

```
x = normalize(x)
```

```
# clustering labeling
```

```
kmeans = KMeans(n_clusters=10, max_iter=10)
```

```
labels = kmeans.fit_predict(x)
```

```
centers = kmeans.cluster_center_
```

-
- 그러나 k-means 를 학습한 뒤, 각 군집의 의미 해석은 분석가의 몫입니다.
 - 이 발표에서는 문서 군집화를 위해 학습한 k-means 를 해석하는 방법에 대하여 다룹니다.
 - Centroid vectors 를 이용하여 각 군집에 레이블을 부여합니다.
 - pyLDAvis 를 이용하여 k-means 학습 결과를 시각화 합니다.

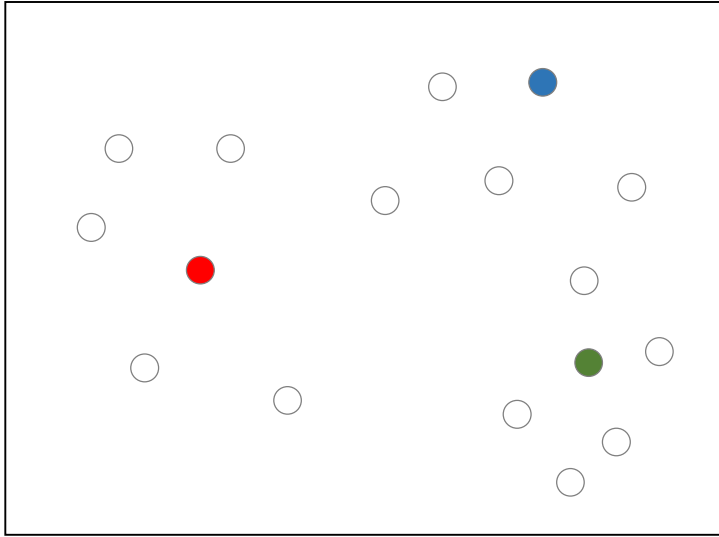
-
- 또한 scikit-learn 의 k-means 는 문서 군집화에 최적이지는 않습니다.
 - 이 발표에서는 문서 군집화를 위하여 효율적으로 k-means 를 학습하는 방법에 대하여 논의합니다.

k-means review

k -means clustering

- 군집화는 비슷한 데이터를 하나의 집합(군집)으로 묶습니다.
- k -means 는 n 개의 데이터를 k 개의 군집으로 나눕니다.

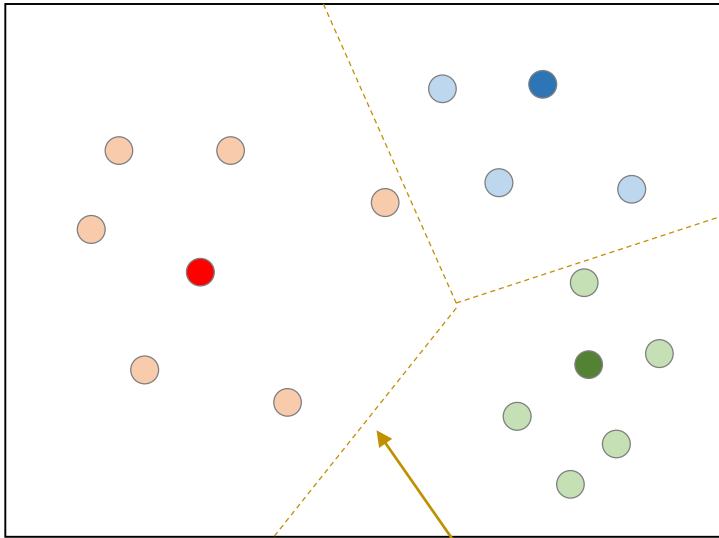
k -means clustering



1. Initialize

$k=3$ 이라 가정하면 3개의 점을 임의로 선택

k -means clustering



1. Initialize

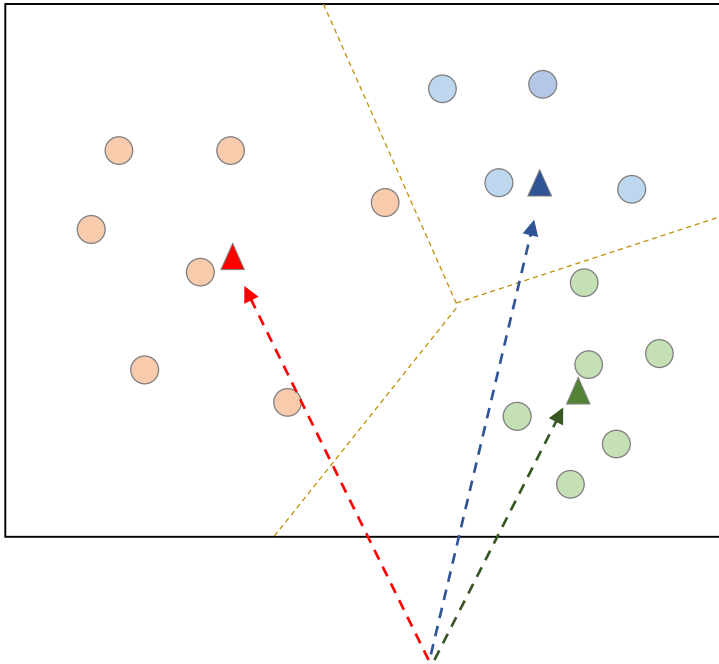
$k=3$ 이라 가정하면 3개의 점을 임의로 선택

2. Assign (epoch=0)

모든 점을 k 개의 centroid 중 가장 가까운 점의 색깔(label)로 할당

k 개의 centroids에 의하여 분할된 공간의 경계면으로, Voronoi partition, Voronoi diagram이라 부름

k -means clustering



데이터에는 존재하지 않는 가상의 centroids

1. Initialize

$k=3$ 이라 가정하면 3개의 점을 임의로 선택

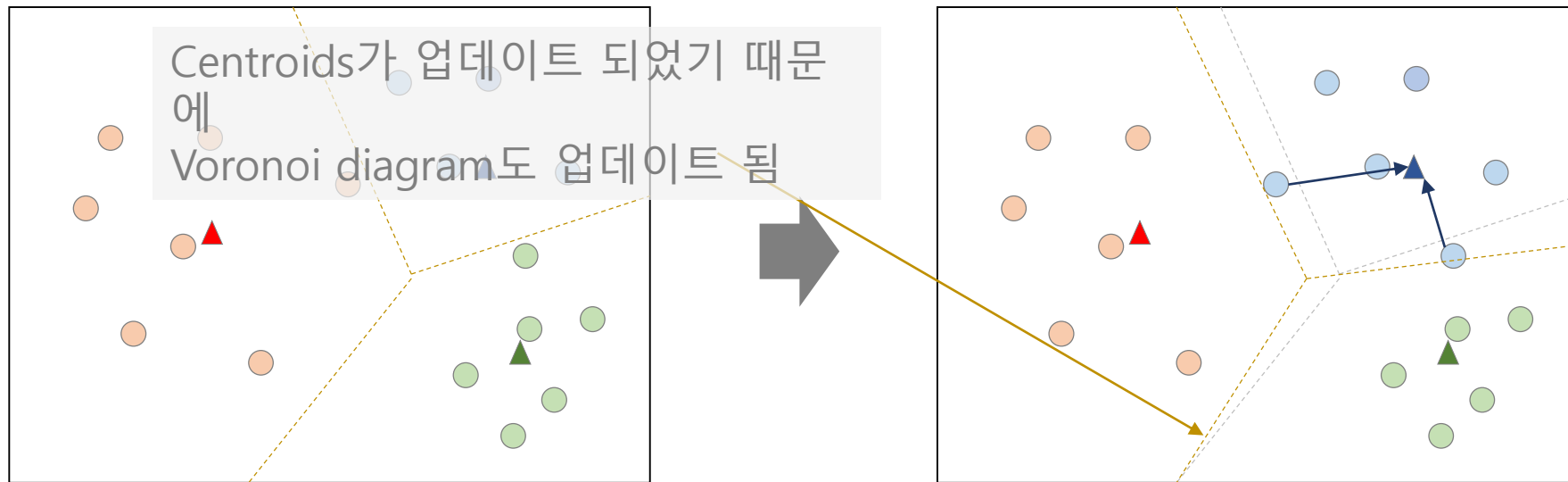
2. Assign (epoch=0)

모든 점을 k 개의 centroid 중 가장 가까운 점의 색깔(label)로 할당

3. Update centroid (epoch=0)

같은 색깔(label) 점들의 평균값을 가상의 centroids로 설정

k -means clustering



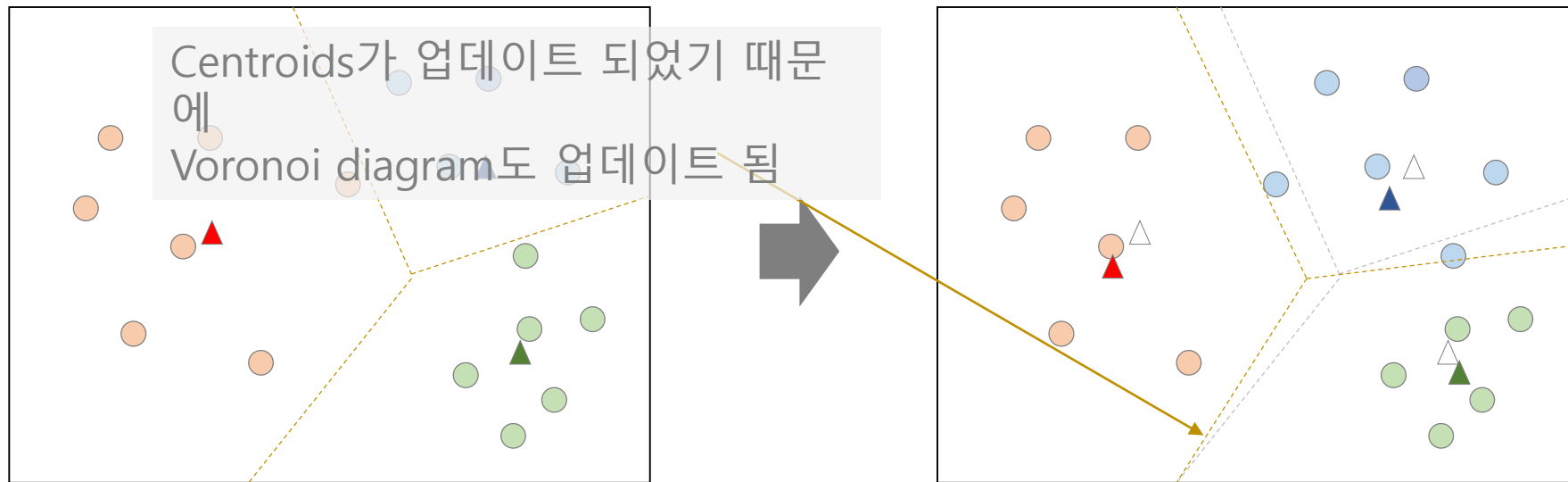
1. Initialize

$k=3$ 이라 가정하면 3개의 점을 임의로 선택

2. Assign (epoch=1)

모든 점을 업데이트 된 centroids 중 가장 가까운 점으로 할당

k -means clustering



1. Initialize

$k=3$ 이라 가정하면 3개의 점을 임의로 선택

2. Assign (epoch=1)

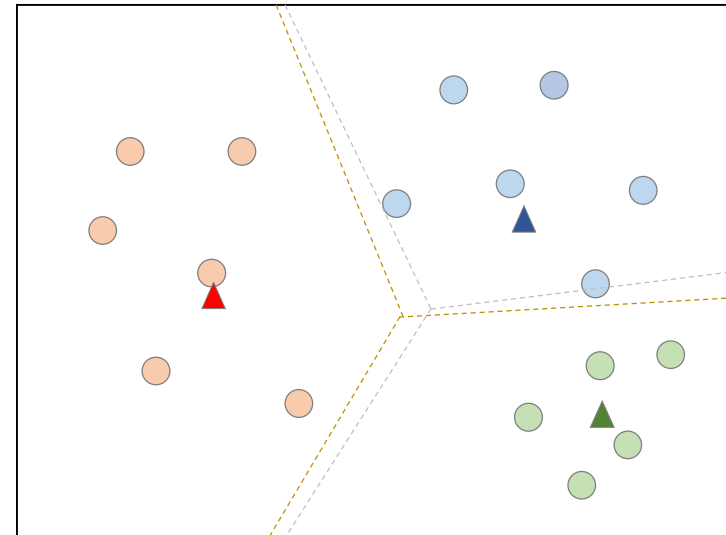
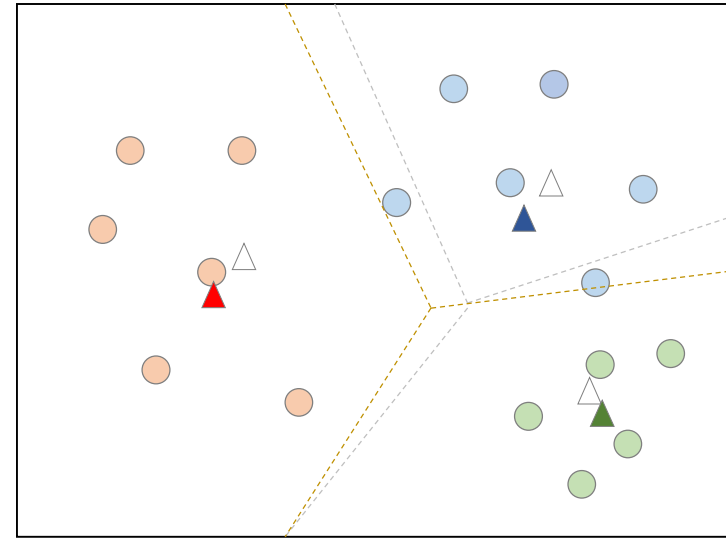
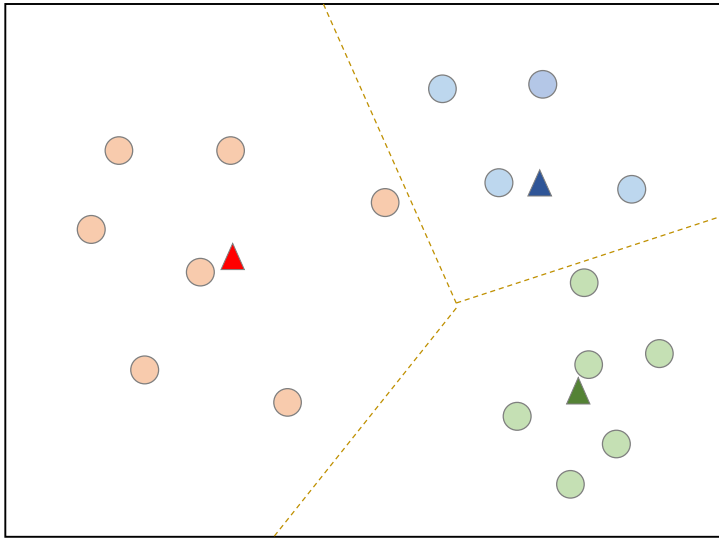
모든 점을 업데이트 된 centroids 중 가장 가까운 점으로 할당

3. Update centroid (epoch=1)

색깔이 바뀐 점이 있기 때문에 Centroid를 다시 업데이트

알고리즘이 종료 될 때까지 2, 3을 반복

k -means clustering



1. Initialize

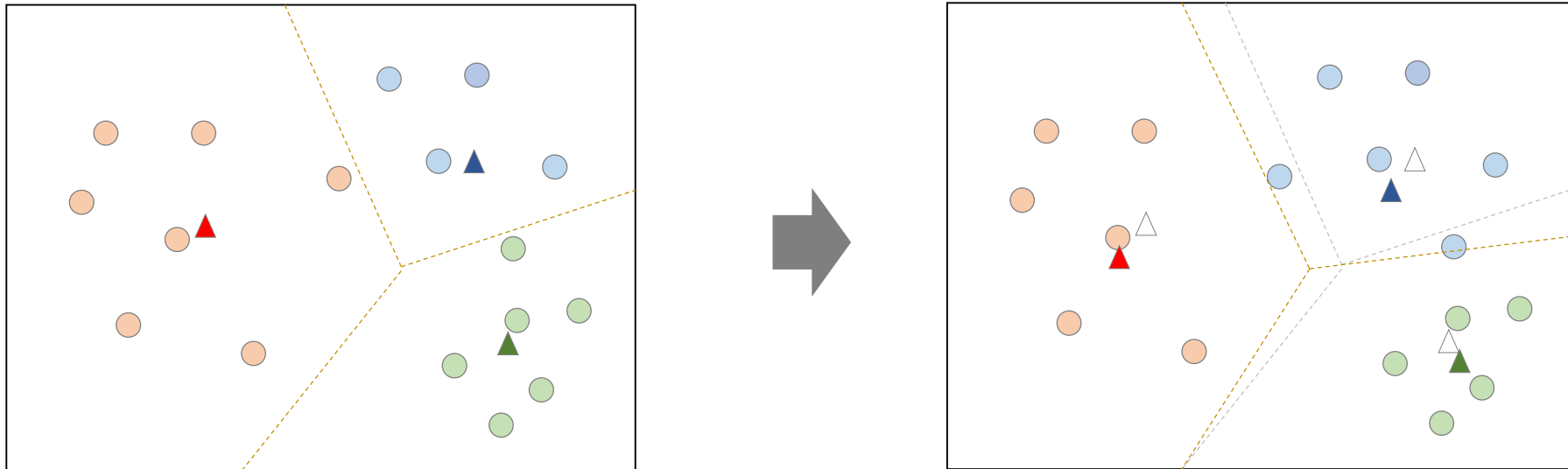
$k=3$ 이라 가정하면 3개의 점을 임의로 선택

2. Assign (epoch=2)

모든 점을 가장 가까운 centroids로 할당하여도 색깔이 변하지 않으므로 알고리즘 종료

Centroids based clustering

- 우리에게 익숙한 k -means 은 local optimal 을 찾는 heuristic 입니다 [1].
 - Lloyd k -means 는 수렴할 때까지 반복적으로 centroids를 업데이트 합니다.



k -means

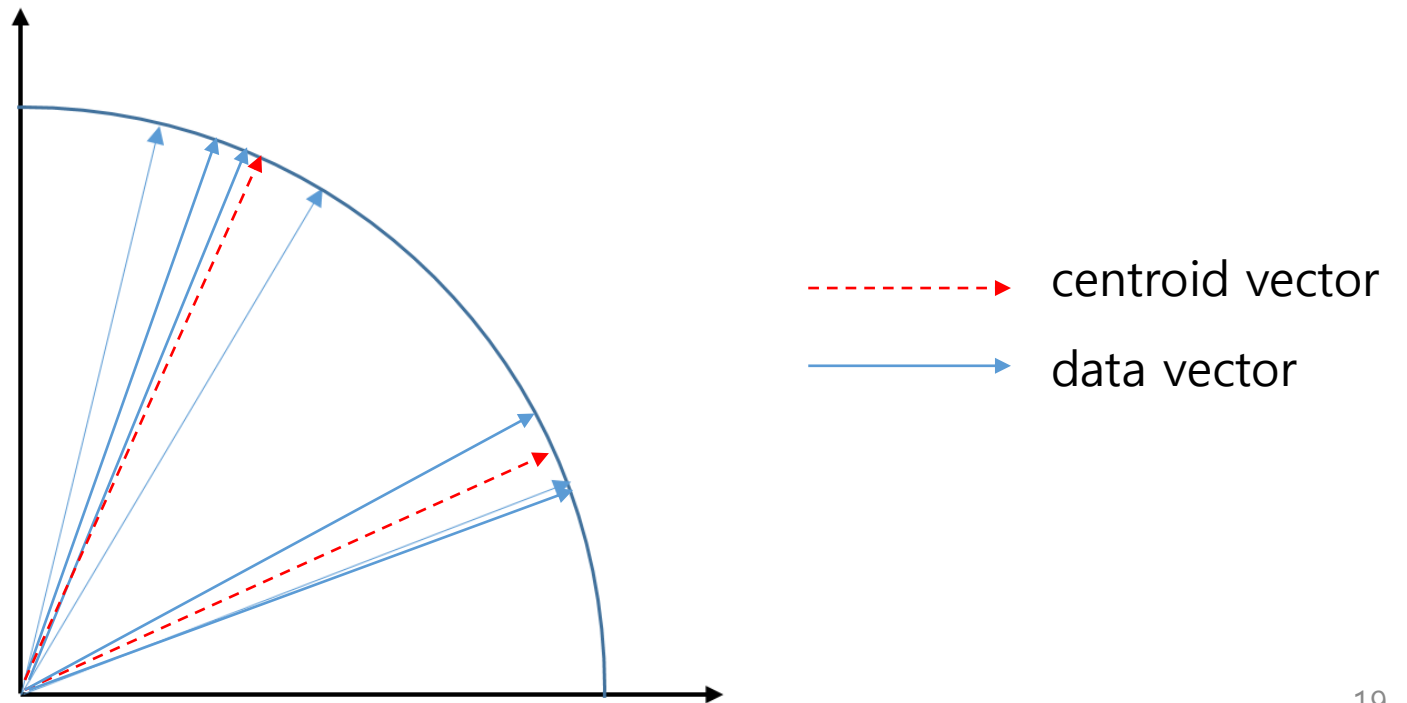
- (Lloyd) k -means 대량의 문서 집합의 군집화에 가장 현실적인 방법입니다.
 - 계산복잡도가 작습니다. $O(n * i * k)$
 - pairwise distance 를 요구하지 않기 때문에 대량의 데이터에 적합하며,
 - row 단위로 학습하기 때문에 mini-batch / 분산환경의 구현이 쉽습니다.

Spherical k -means

- 문서 간 거리는 Euclidean distance 를 이용해서는 안됩니다.
 - 문서 간 유사도는 두 문서의 공통된 단어 유무가 제일 중요한 정보입니다 [1].
 - Euclidean distance 는 이를 고려하지 않습니다.
 - Cosine distance 는 두 문서에 공통으로 등장한 단어의 비율을 고려합니다.
 - Spherical k -means 는 cosine distance 을 이용하는 k -means 입니다.

Spherical k -means

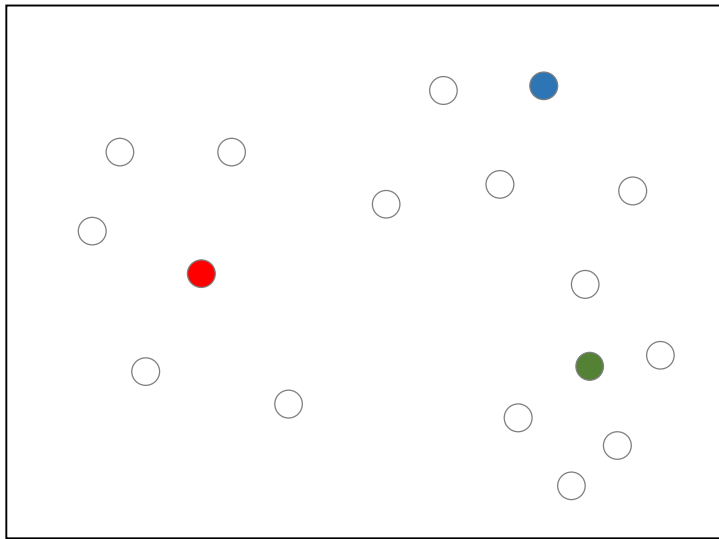
- Cosine distance 를 이용하는 k -means 는 각도 (단어 분포) 가 비슷한 문서를 하나의 군집으로 묶습니다.



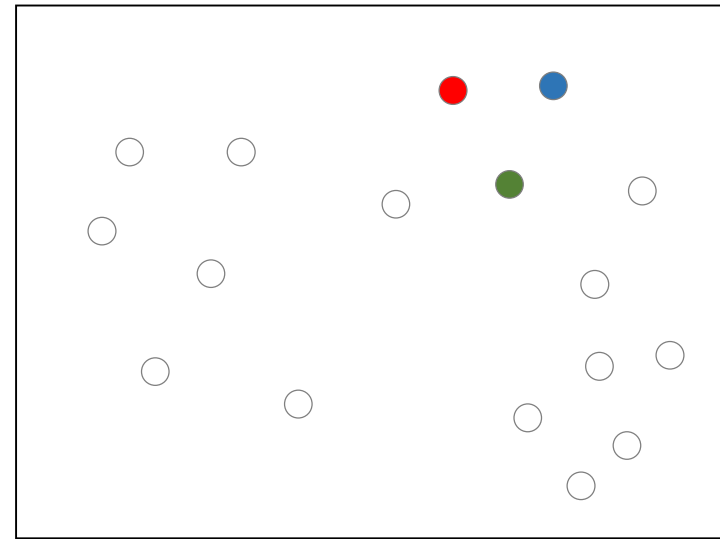
문서 군집화를 위한 *k*-means initialization

Fast initializer for document clustering

- k -means 는 initial points 만 잘 뽑아도 수렴이 빠르다고 알려져 있으며,
 - 사실, 최악의 initialization 만 아니면 수렴은 원래 빠릅니다.
 - 특정 지역에 initial points 가 모여있지만 않아도 됩니다.



Ideal initialization



Worst initialization

Fast initializer for document clustering

- ***k-means++*** [1] 은 가장 널리 알려진 initialization method 입니다.
 - 지금의 initial point c_{t-1} 에서 먼 점을 우선적으로 선택합니다.

1. Select a point c_0 randomly
2. Select next point c_t with prob. $\frac{d(c_{t-1}, c_t)^2}{\sum_t d(c_{t-1}, c_t)^2}$
3. Repeat step 2 until choosing k points

Fast initializer for document clustering

- k -means++^[1] 은 가장 널리 알려진 initialization method 입니다.
- scikit-learn 에 구현되어 있습니다.

[sklearn.cluster](#).KMeans ¶

```
class sklearn.cluster.KMeans(n_clusters=8, init='k-means++', n_init=10, max_iter=300, tol=0.0001,  
precompute_distances='auto', verbose=0, random_state=None, copy_x=True, n_jobs=1, algorithm='auto')
```

[\[source\]](#)

Fast initializer for document clustering

- k -means++^[1] 은 고차원 데이터에는 의미가 없습니다.
 - 고차원 데이터에서는 가까운 점의 거리는 $d(x_i, x_j) \cong 0$ 이지만,
 - 조금만 멀어져도 대부분의 점들 간 거리의 값이 비슷합니다.
 - k -means++ 은 저차원 데이터에 적합합니다.
- 특히 sparse data + Cosine 에서는 대부분의 거리가 1 입니다.
 - 비싼 random sampling

[1] Arthur, D., & Vassilvitskii, S. (2007, January). k -means++: The advantages of careful seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms* (pp. 1027-1035). Society for Industrial and Applied Mathematics.

Fast initializer for document clustering

- 하루의 뉴스 데이터 (30,091 개)의 pairwise distance 예시
 - 고차원의 데이터에서는 비슷하다는 것 외의 거리는 의미가 없습니다.

Cosine distance	Percentage
0.00 ~ 0.10	0.31%
0.10 ~ 0.20	0.31%
0.20 ~ 0.30	0.32%
0.30 ~ 0.40	0.19%
0.40 ~ 0.50	0.30%
0.50 ~ 0.60	0.32%
0.60 ~ 0.70	0.54%
0.70 ~ 0.80	2.11%
0.80 ~ 0.90	10.22%
0.90 ~ 1.00	85.39%

Fast initializer for document clustering

- 고차원 벡터에서는 가깝다라는 의미는 있지만, 멀다라는 의미가 없습니다.

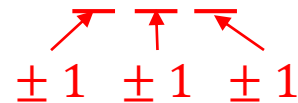
Assume that $|x-y|^2 = 1$ and x, y are integer vector

1차원 $X = [1], y = [0] \text{ or } [2]$

2차원 $X = [1, 0], y = [1, 1], [1, -1], [0, 0], [2, 0]$

3차원 $X = [1, 0, 0], y = [1, 0, 0]$

$\pm 1 \quad \pm 1 \quad \pm 1$



Fast initializer for document clustering

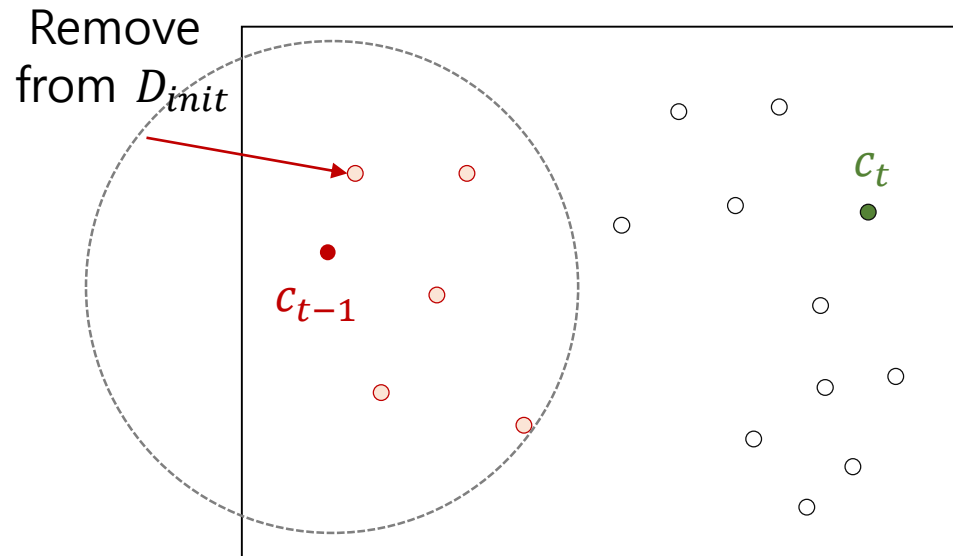
- Initial points 는 거의 비슷한 점만 아니면 충분히 괜찮으며, k 개의 points 를 선택하기 위하여 $n * k$ 번의 계산은 불필요합니다.
- Term frequency representation 의 특징을 이용하면 널리 퍼진 initial points 를 빠르게 찾을 수 있습니다.

Fast initializer for document clustering

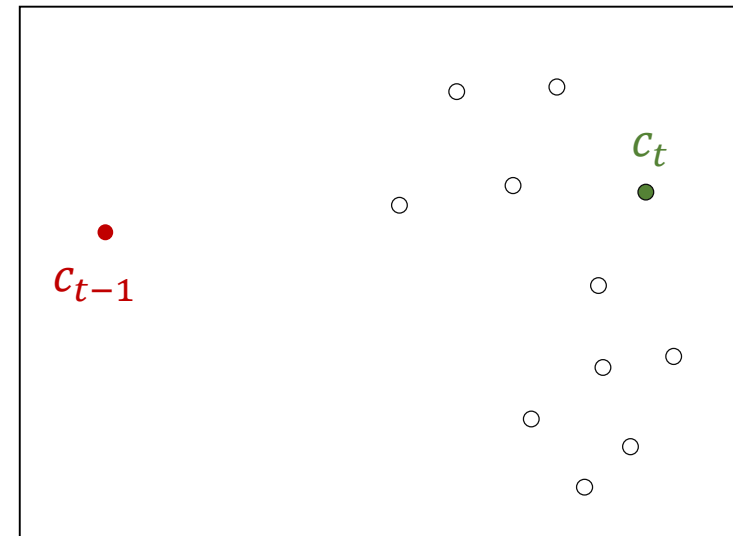
1. $\alpha * k$ 개의 후보를 random sampling $D_{init} \subset D$
2. D_{init} 에서 한 개의 점 c_i 을 임의로 선택.
3. D_{init} 에서 c_i 와 Cosine similarity 가 t_{init} 보다 큰 점을 D_{init} 에서 삭제
4. D_{init} 이 공집합이 아니면 k 개의 점을 뽑을 때까지 step 2 – 3 반복
5. D_{init} 이 공집합이며, 현재까지 선택한 점의 개수, k_0 가 $k_0 < k$ 이면
 $k - k_0$ 개의 점을 $D - D_{init}$ 에서 임의로 추출

Fast initializer for document clustering

- D_{init} 에서 한 번 선택된 점 주위의 다른 점들을 모두 제거하면 중복되거나, 근처의 점이 선택될 위험이 적습니다.



c_{t-1} 에서 먼 점 중 하나는 c_t



$c_{t-1} \cong c_{t+1}$ 인 점은 선택되지 않음

Spherical k -means with fast initializer

```
from soyclustering import SphericalKMeans
from soyclustering import proportion_keywords

# spherical k-means

spherical_kmeans = SphericalKMeans(n_clusters=1000, max_iter=10, verbose=1, init='similar_cut')
labels = spherical_kmeans.fit_predict(x)

# clustering labeling

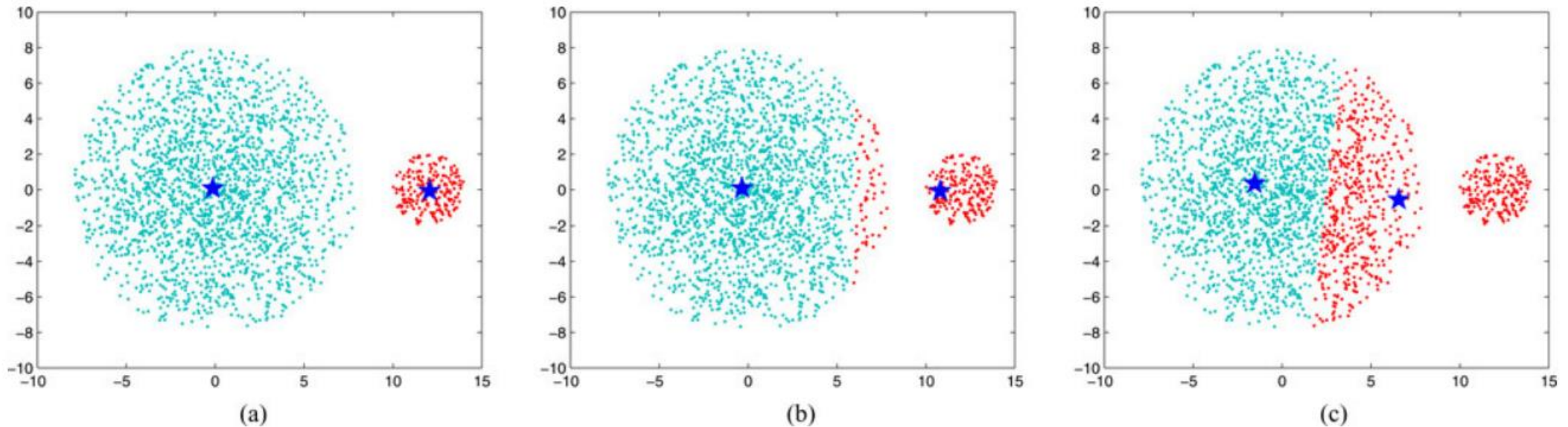
vocabs = ['this', 'is', 'vocab', 'list']
centers = kmeans.cluster_centers_
keywords = proportion_keywords(centers, labels, vocabs)
```

<https://github.com/lovit/clustering4docs>

Defining k

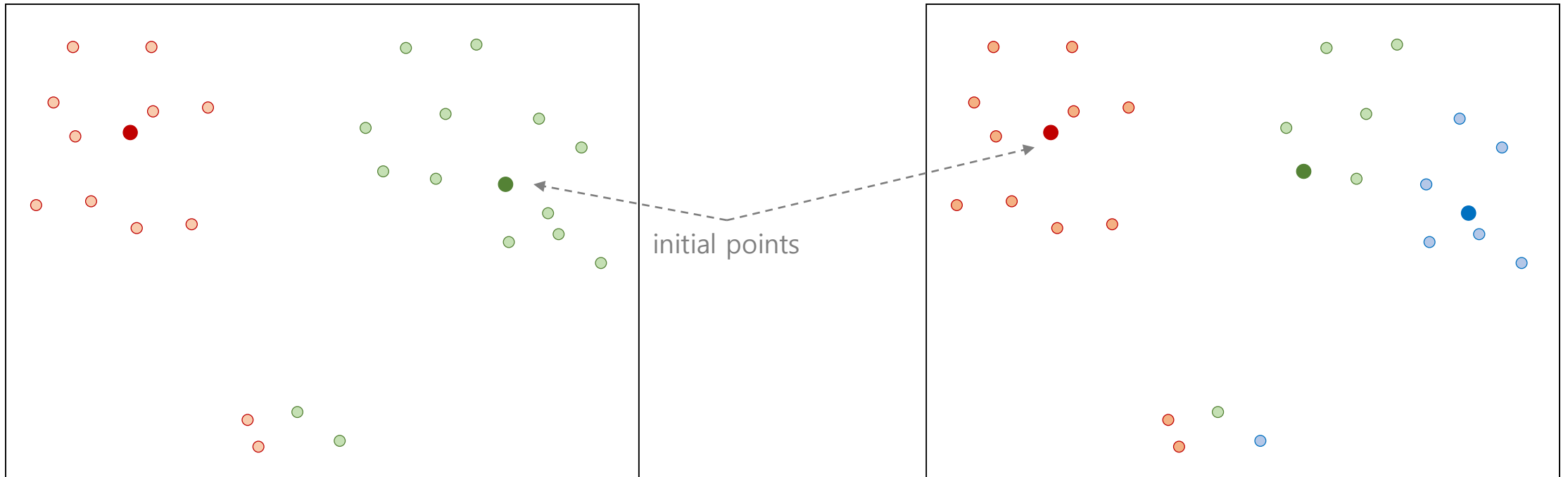
Defining k

- Imbalanced class data 일 때, class distribution 이 잘 반영되지 않고, 모든 군집의 크기가 균일해지는 현상 (uniform effect) 이 발생합니다 [1]



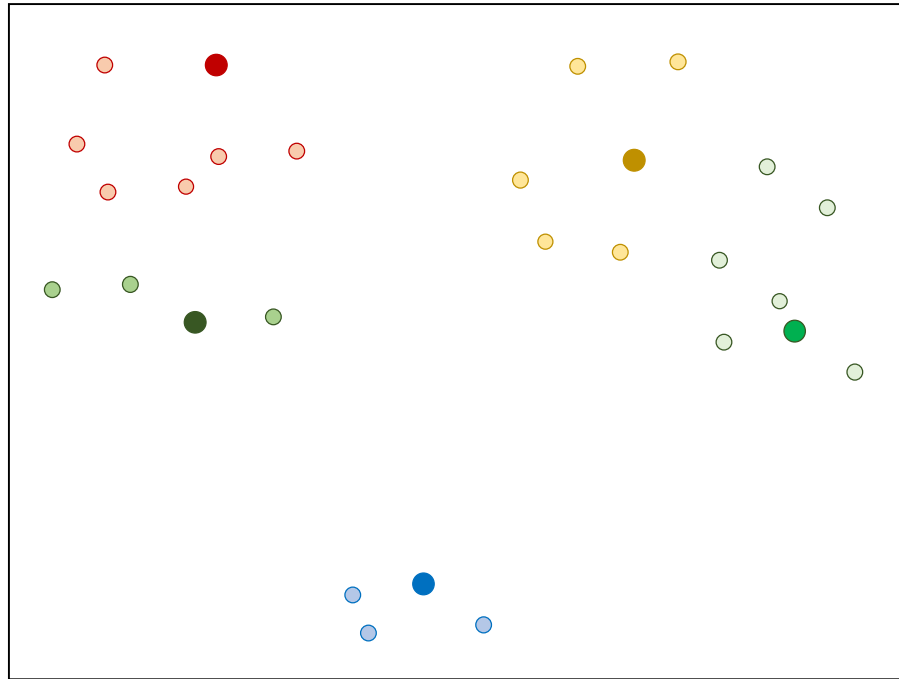
Defining k

- 작은 군집은 독립된 군집으로 선택되기가 어렵습니다.
 - Initial points 가 선택되지 않으면 큰 군집들로 나뉘어져 흡수됩니다.
 - 한 번 나뉘어진 작은 군집이 독립된 군집이 될 가능성은 거의 없습니다.



Defining k

- 큰 군집이 여러 개의 중복된 군집이 되더라도, k 를 크게 설정하여 작은 군집들이 독립적으로 존재하도록 해야 합니다.



Defining k

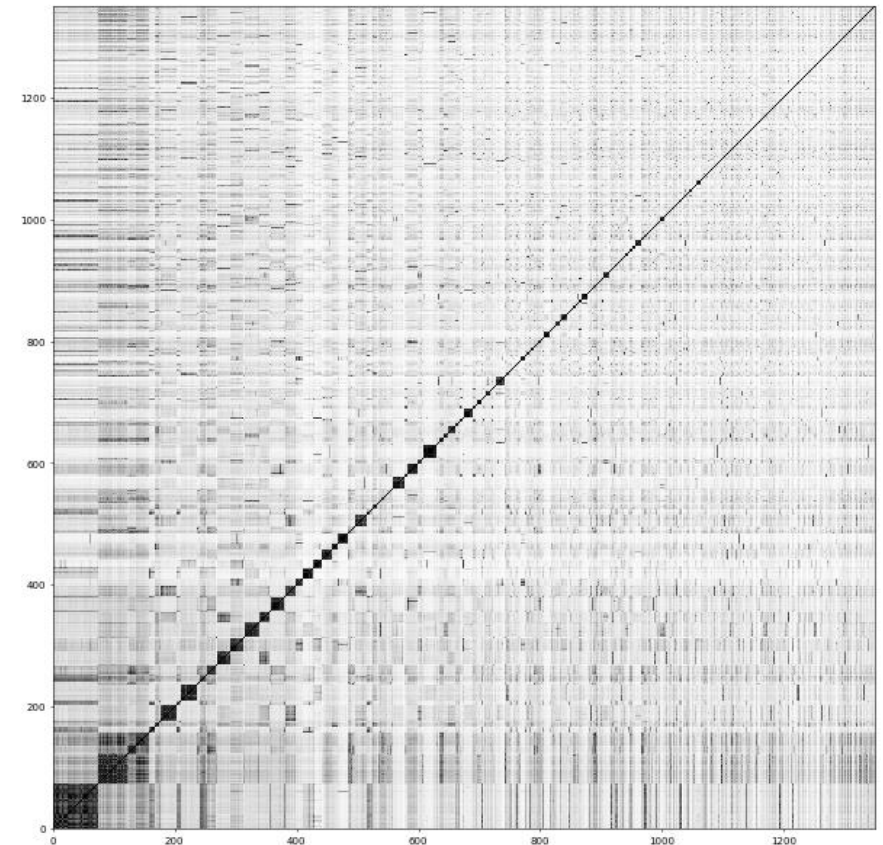
- 현실적인 미봉책은 예상하는 군집의 개수보다 크게 k 설정한 뒤, 후처리로 비슷한 군집을 병합합니다.
 - 학습 후, 하나의 군집이 여러개로 나뉘어 졌는지를 확인하기는 쉽습니다.
 - 하나의 군집에서 잘못된 점을 찾는 것이 더 어려우며, 그 점들의 후처리도 어렵습니다.

Defining k

- 현실적인 미봉책은 예상하는 군집의 개수보다 크게 k 설정한 뒤, 후처리로 비슷한 군집을 병합합니다.
 - 실제 군집의 개수보다 k 가 크면 major 군집들이 여러 개로 나뉘어집니다.
 - k 가 작으면 minor 군집이 찢어질 가능성이 높습니다.
 - 이 때 centroids 는 major 편입니다.

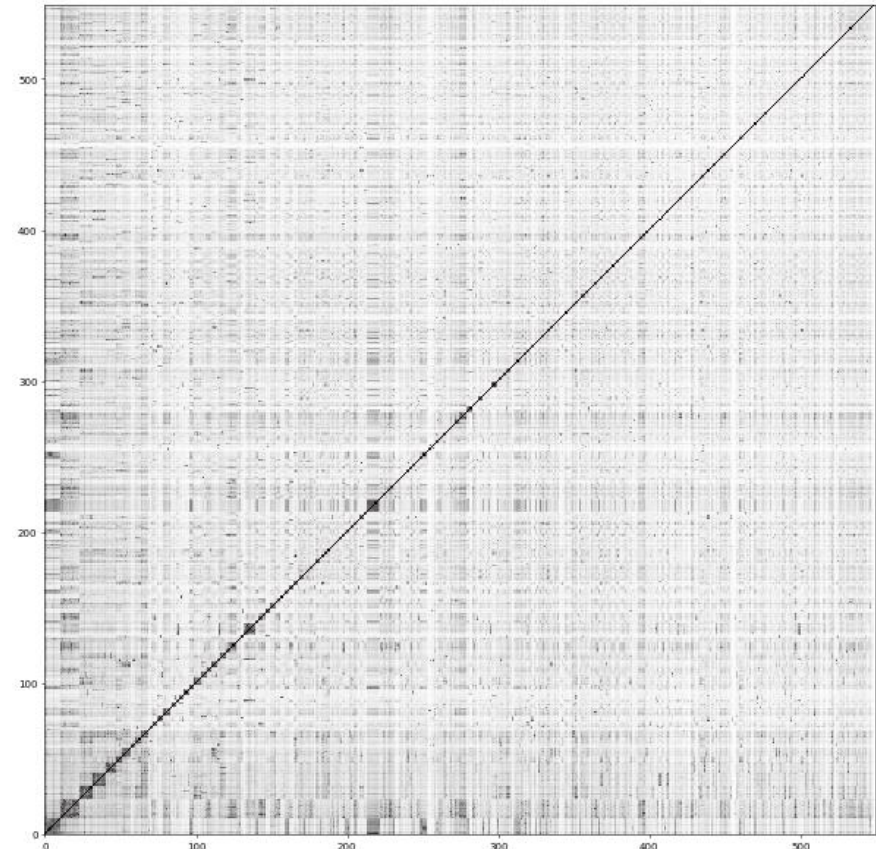
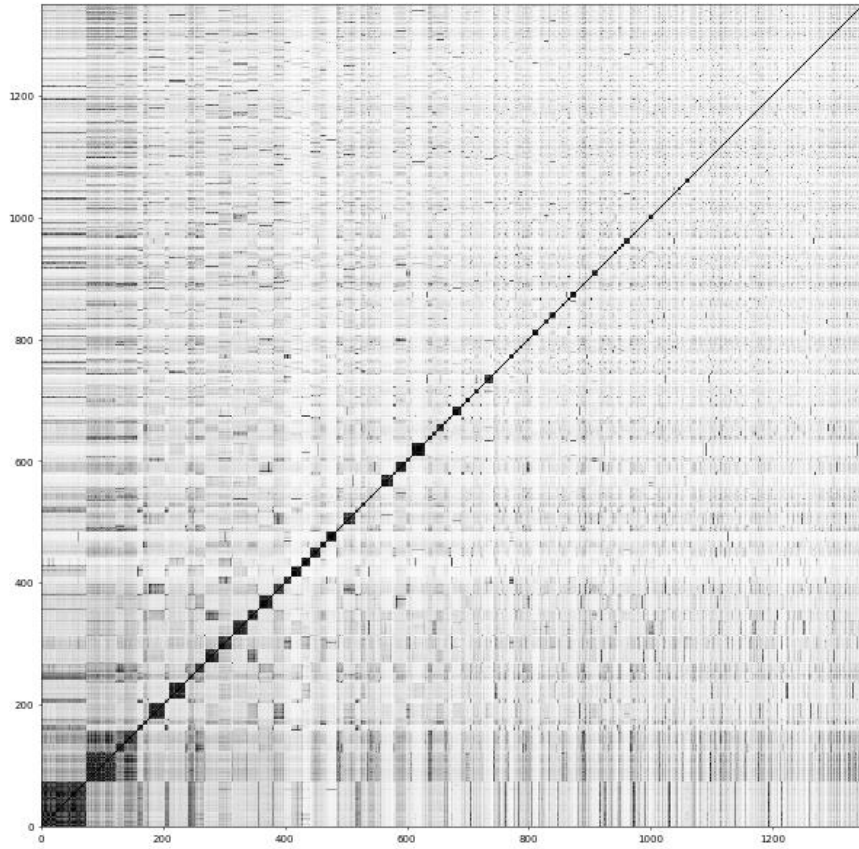
Defining k

- Centroid vectors 의 pairwise distance matrix 는 군집화 결과의 직관적인 이해를 도와줍니다.
- Diagonal 만 진할수록, 각 cluster 는 separation 이 잘 이뤄진 것입니다.
- 진한 square 를 하나의 군집으로 묶을 수 있습니다.



Defining k

- term frequency vector 간의 cosine similarity $\geq t$ (eg 0.4) 를 하나의 군집으로 묶음으로서 손쉽게 후처리를 할 수 있습니다.



Defining k

```
from soyclustering import visualize_pairwise_distance
from soyclustering import merge_close_clusters

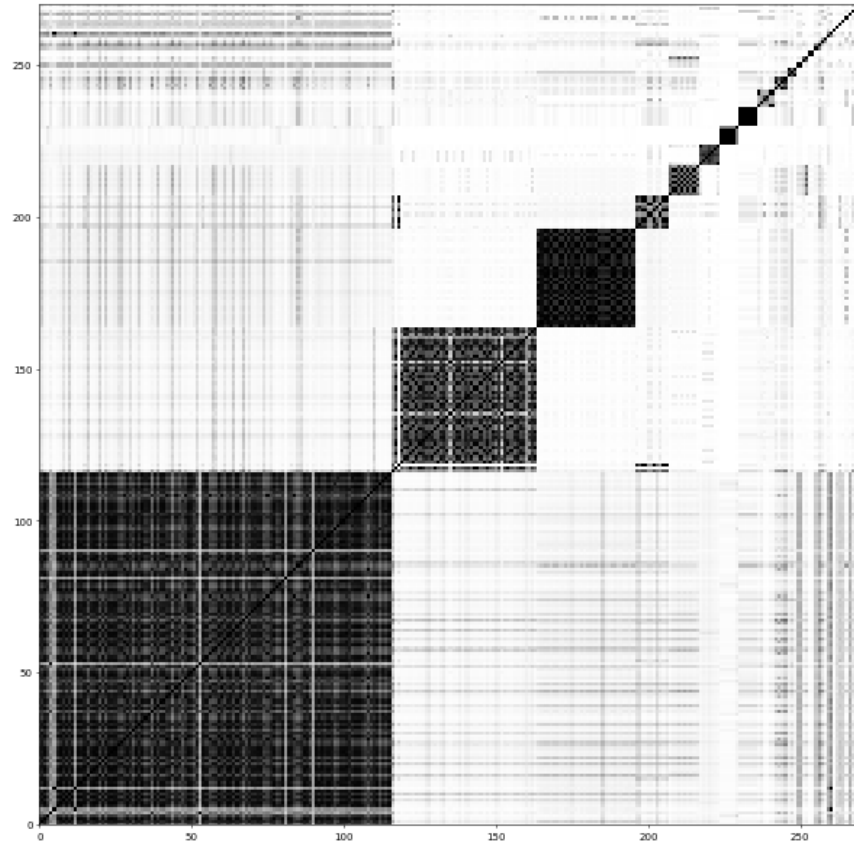
# visualize pairwise distance matrix
fig = visualize_pairwise_distance(centers, max_dist=.7, sort=True)

# postprocessing: merging similar clusters
group_centers, groups = merge_close_clusters(centers, labels, max_dist=.5)

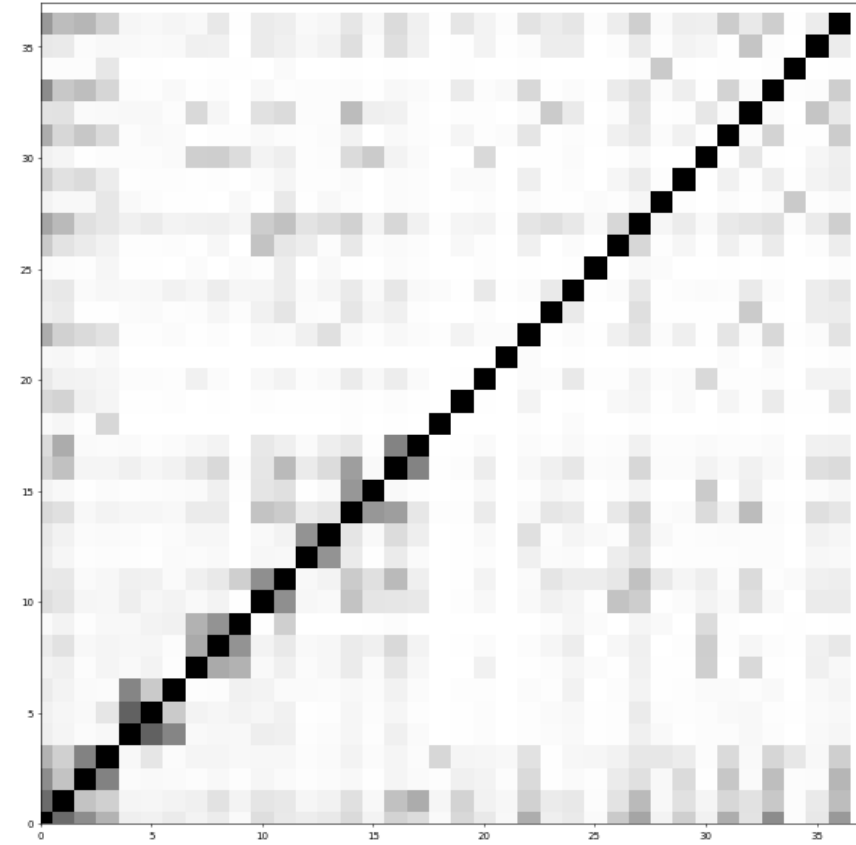
# visualize pairwise distance matrix of grouped cluster centroids
fig = visualize_pairwise_distance(group_centers, max_dist=.7, sort=True)
```

<https://github.com/lovit/clustering4docs>

Defining k



n clusters = 270



n clusters = 37

Defining max iter

- k -means type 은 몇 번의 반복으로 거의 수렴합니다.
 - 그렇다면 “repeat until converged” 동안의 학습은 더 좋은 걸까요?
 - Uniform effect 가 일어나는 과정일 수도 있습니다.

k=100, n = 30,091

iteration	num_changed	inertia	times [sec]
1	29969	15323.44	4.435
2	5062	11127.62	4.466
3	2179	10675.31	4.463
4	1040	10491.64	4.449
5	487	10423.5	4.437
6	297	10392.49	4.483
7	178	10373.65	4.442
8	119	10362.63	4.449
9	78	10355.91	4.438
10	80	10350.7	4.452

Centroid based cluster labeling

Centroid based cluster labeling

- k -means 의 결과는 centroid vectors 와 labels 두 가지 입니다.
- A centroid vector 는 해당 군집에서의 term frequency proportion 과 비슷합니다.

Centroid based cluster labeling

- 다른 군집보다 상대적으로 자주 등장한 단어를 키워드로 정의합니다.
- 군집 c_i 에서의 단어 w_{ij} 의 키워드 점수, $s(w_j, c_i)$ 를 정의합니다.

$$s(w_j, c_i) = \frac{p_i(w_j)}{p_i(w_j) + p_{-i}(w_j)}$$

$p_i(w_j)$: c_i 에서 단어 w_j 의 비율

$p_{-i}(w_j)$: c_i 외에서 단어 w_j 의 비율

Centroid based cluster labeling

- 군집의 크기를 고려하여 p_i, p_{-i} 를 정의합니다.

- $p_i(w_j) = c_{i,w_j}$

- $p_{-i}(w_j) = \frac{1}{\sum_{k \neq i} DF(C_k)} \times \sum_{k \neq i} DF(C_k) * c_{k,w_j}$

Centroid based cluster labeling

- 한 군집의 키워드는 군집 내 여러 문서들에서 등장해야 합니다
 - Coverage 가 큰 단어는 centroids 의 값이 큼니다. 이 값이 큰 k_1 개의 단어 중에서 $s(w_j, c_i)$ 의 값이 큰 k_2 개의 단어를 labels 로 선택합니다.

1. for each cluster and its centroid c_i , select top k_1 words order by c_{i,w_j}
2. compute $s(w_j, c_i)$ and select top k_2

Centroid based cluster labeling

- IMDB reviews ($k=1,000$)
 - 2,514 개의 영화 리뷰가 포함되어 있기 때문에 $k=1,000$ 으로 학습합니다.
 - 5 개의 예시 군집들의 의미가 잘 파악됩니다.

영화 "타이타닉"	iceberg, zane, sinking, titanic, rose, winslet, camérons, 1997, leonardo, leo, ship, cameron, dicaprio, kate, tragedy, jack, disaster, james, romance, love, effects, special, story, people, best, ever, made
Marvle comics 의 heros (Avengers)	zemo, chadwick, boseman, bucky, panther, holland, cap, infinity, mcu, russo, civil, bvs, antman, winter, ultron, airport, avengers, marvel, captain, superheroes, soldier, stark, evans, america, iron, spiderman, downey, tony, superhero, heroes
Cover-field, District 9 등 외계인 관련 영화	skyline, jarrod, balfour, strause, invasion, independence, cloverfield, angeles, district, los, worlds, aliens, alien, la, budget, scifi, battle, cgi, day, effects, war, special, ending, bad, better, why, they, characters, their, people
살인자가 출연하는 공포 영화	gayheart, loretta, candyman, legends, urban, witt, campus, tara, reid, legend, alicia, englund, leto, rebecca, jared, scream, murders, slasher, helen, killer, student, college, students, teen, summer, cut, horror, final, sequel, scary
영화 "매트릭스 "	neo, morpheus, neos, oracle, trinity, zion, architect, hacker, reloaded, revolutions, wachowski, fishburne, machines, agents, matrix, keanu, smith, reeves, agent, jesus, machine, computer, humans, fighting, fight, world, cool, real, special, effects

Centroid based cluster labeling

- “소나타”가 포함된 네이버 블로그 (k=500)
 - 소나타는 다의어이기 때문에 다양한 문맥에서 이용됩니다.
 - 각 군집의 레이블로부터 소나타의 문맥을 유추할 수 있습니다.

렌트카 광고	제주렌트카, 부산출발제주도, 제주신, 이골림, 제주올레, 왕복항공, 불포함, 제주도렌트카, 064, 롯데호텔, 자유여행, 객실, 제주여행, 특가, 해비치, 제주시, 제주항, 티몬, 2박3일, 올레, 유류, 항공권, 조식, 제주도여행, 제주공항, 2인
중고차 매매	최고급형중고, 최고급, 프리미어, 프라임, 2011년식, YF소나타TOP, 2010년식, 풀옵션, 2011년, YF소나타PR, 1인, Y20, 2010년, 완전무사고, 판매완료, 군포, 검정색, YF쏘나타, 2011, 하이패스, 2010, 무사고, 등급, 파노라마, 허위매물
클래식 음악	금관악기, 아이엠, Tru, 트럼펫, 트럼, 나팔, 금관, 텔레만, Eb, 호른, 오보에, Tr, Concerto, 하이든, 협주곡, Ha, 악기, 연주하는, 오케, 오케스트라, 독주, 악장, 작곡가, 곡
아이비 “유혹의 소나타 ”	Song, 공부할, 부른, 노래, 가사, 부르는, 가수, 보컬, 목소리, 발라드, 명곡, 신나, 들으면, 듣기, 유혹의, 앨범,아이비, 제목
광염 소나타 및 일제강점기 소설들	백성수, 발가락, 현진, 이광수, 김유, 자연주의, 친일, 평양, 운수, 유미, 저지르, 야성, 탐미, 김동인, 복녀, 광염, 닦았다, 사실주의, 광기, 저지, 1920, 단편소설, 범죄, 감자, 동인, 한국문학

Visualization: pyLDAvis + k-means

LDA review

- LDA 는 토픽 모델링에 이용되는 대표적인 알고리즘입니다.
 - 토픽은 특정 주제에 대한 단어 집합입니다.
 - 단어 빈도 벡터 x 로 표현된 문서 집합으로부터 토픽을 학습합니다.
 - LDA 는 (1) 토픽 – 단어 확률, (2) 문서 – 토픽 확률을 학습합니다.
 - $P(w | t)$: topic – term prob.
 - $P(t | d)$: doc – topic prob.

LDA review

- 단어 빈도 벡터 x 로부터 (1) 토픽 - 단어 확률과 (2) 문서 - 토픽 확률을 학습합니다.

- n, m, t : 단어, 문서, 토픽 개수

The diagram illustrates the matrix equation for LDA. On the left is a green square representing the word frequency matrix x , with dimensions $m \times n$ indicated below it. This is followed by an equals sign. To the right of the equals sign is a blue rectangle representing the document-term probability matrix, labeled "doc - term prob." inside, with dimensions $m \times t$ indicated below it. This is followed by a multiplication symbol \times . To the right of the multiplication symbol is an orange rectangle representing the topic-term probability matrix, labeled "topic - term prob." inside, with dimensions $t \times n$ indicated below it.

$$\begin{matrix} x \\ m \times n \end{matrix} = \begin{matrix} \text{doc - term} \\ \text{prob.} \\ m \times t \end{matrix} \times \begin{matrix} \text{topic - term prob.} \\ t \times n \end{matrix}$$

LDA review

```
import gensim
from gensim.corpora.dictionary import Dictionary
from gensim.models import LdaModel

# training
corpus = gensim.matutils.Sparse2Corpus(x, documents_columns=False)
dictionary = Dictionary.from_corpus(corpus, id2word=id2word_dict)
lda_model = LdaModel(corpus, id2word=dictionary, num_topics=50)

# get topic – term prob.
def get_topic_term_prob(lda_model):
    topic_term_freqs = lda_model.state.get_lambda()
    topic_term_prob = topic_term_freqs / topic_term_freqs.(axis=1)[: , None]
    return topic_term_prob

# get doc – topic prob.
doc_topic_prob, _ = topic_model.inference(corpus)
```

pyLDavis

- 그러나 LDA 의 학습 결과도 해석하기가 어렵습니다.
 - 토픽은 단어 크기의 고차원 벡터이며, 토픽의 개수도 많기 때문입니다.
- LDavis 는 (1) 고차원 토픽 벡터의 시각화와 (2) 토픽 키워드 추출을 통하여 효과적으로 LDA 를 해석할 수 있도록 도와줍니다.

pyLDAvis

- LDAvis 는 Principal Component Analysis (PCA) 를 이용하여 (t, n) 크기의 topic – term prob. 행렬을 $(t, 2)$ 크기로 축소합니다.
 - 비슷한 토픽은 비슷한 2 차원 좌표로 표현됩니다.

pyLDAvis

- 토픽의 키워드는 **한 토픽에서 자주** 등장하거나, **다른 토픽보다 유독 많이** 등장하는 단어입니다.
- LDAvis 는 두 기준의 score 의 가중합으로 **키워드 점수**를 정의합니다.
 - $s(w, t) = \lambda \cdot P(w | t) + (1 - \lambda) \cdot \frac{P(w | t)}{P(w)}$

pyLDAvis

In [15]: `pyLDAvis.display(loader_pyldavis)`

Out [15]:

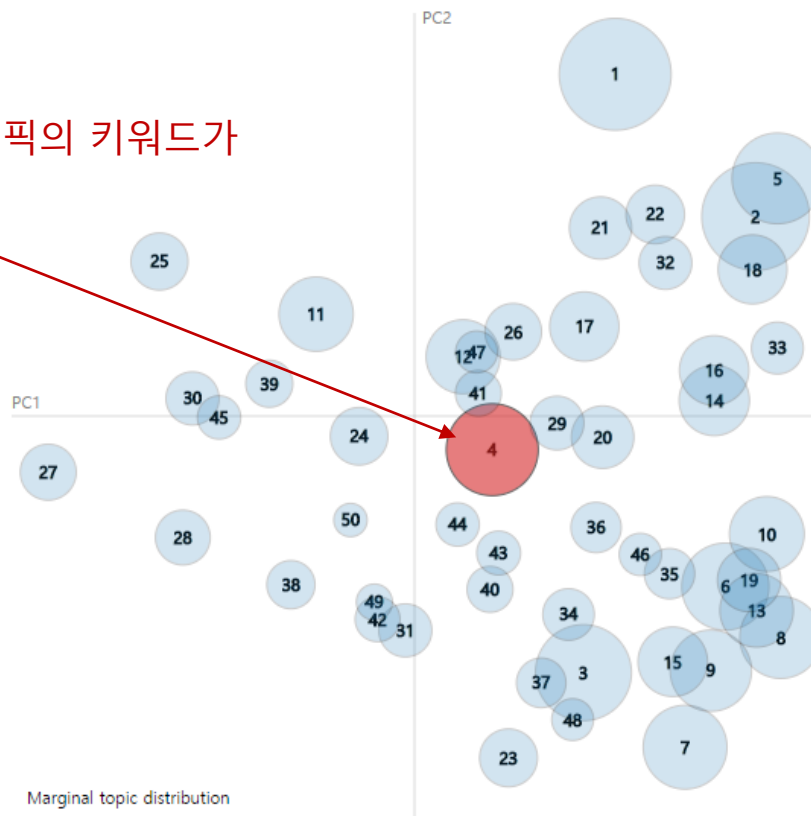
Selected Topic:

Slide to adjust relevance metric:(2)

$\lambda = 1$

0.0 0.2 0.4 0.6 0.8 1

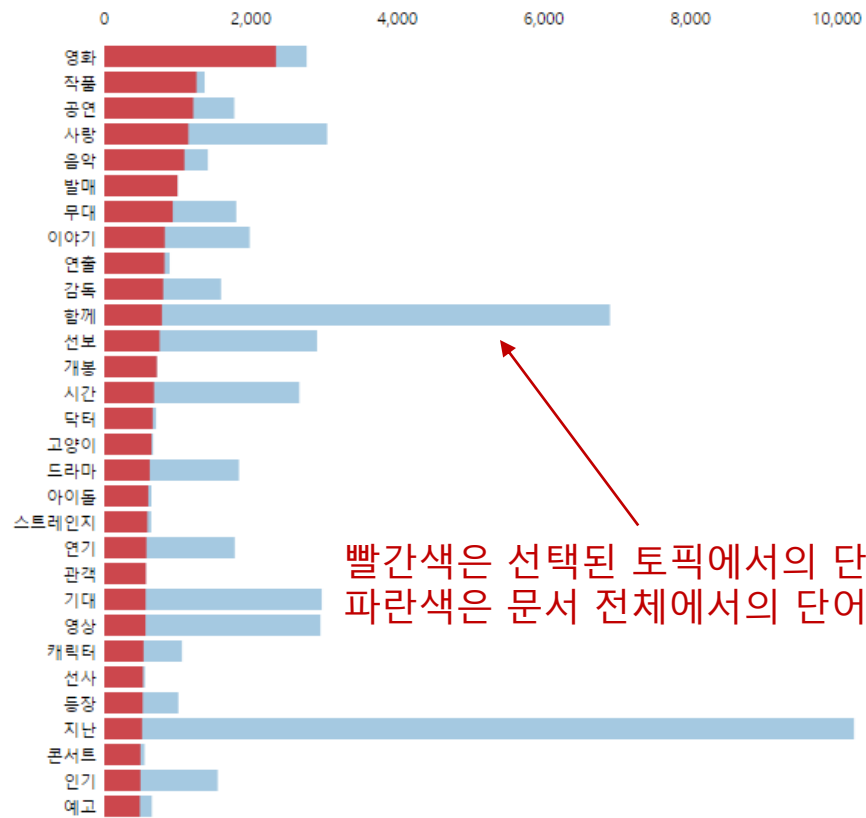
Intertopic Distance Map (via multidimensional scaling)



Marginal topic distribution



Top-30 Most Relevant Terms for Topic 4 (4.1% of tokens)



빨간색은 선택된 토픽에서의 단어 등장 비율,
파란색은 문서 전체에서의 단어 등장 비율입니다

Overall term frequency

Estimated term frequency within the selected topic

1. $\text{saliency}(\text{term } w) = \text{frequency}(w) * [\sum_t p(t | w) * \log(p(t | w) / p(t))]$ for topics t ; see Chuang et al (2012)

2. $\text{relevance}(\text{term } w | \text{topic } t) = \lambda * p(w | t) + (1 - \lambda) * p(w | t) / p(w)$; see Sievert & Shirley (2014)

pyLDavis + Gensim

training LDA

```
import gensim
from gensim.corpora.dictionary import Dictionary
from gensim.models import LdaModel

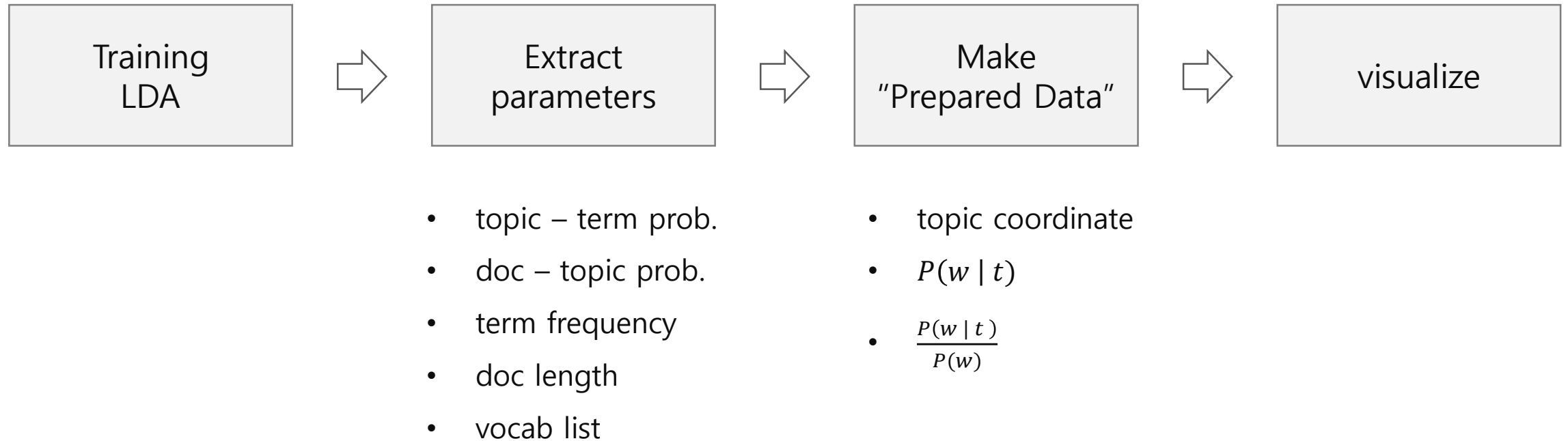
corpus = gensim.matutils.Sparse2Corpus(x, documents_columns=False)
dictionary = Dictionary.from_corpus(corpus, id2word=id2word_dict)
lda_model = LdaModel(corpus, id2word=dictionary, num_topics=50)
```

pyLDavis

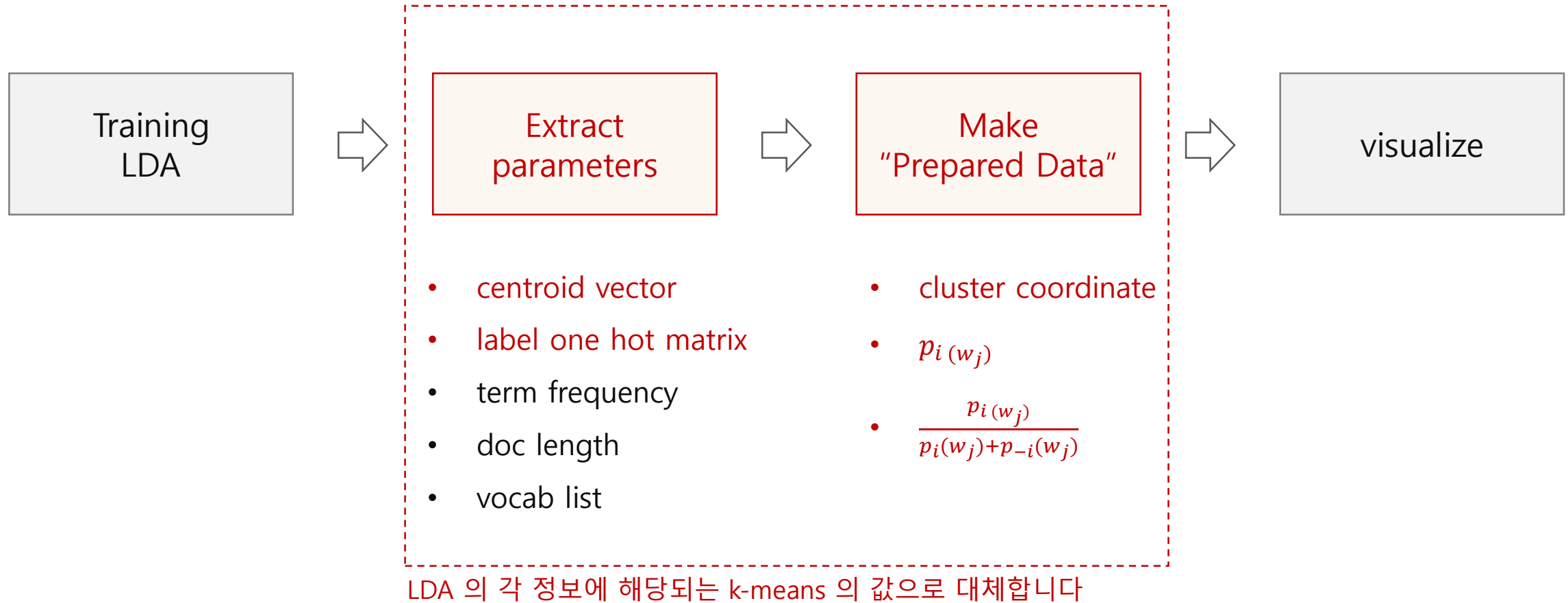
```
import pyLDavis
import pyLDavis.gensim as gensimvis

prepared_data = gensimvis.prepare(lda_model, corpus, dictionary)
pyLDavis.display(prepared_data)
```

Gensim LDA + pyLDAvis



k -means + pyLDAvis



k -means + pyLDAvis

Parameters

LDA	k-means	의미
topic – term prob.	centroid vector	토픽/군집에서 한 단어가 얼마나 등장하는가?
doc – topic prob.	label one hot matrix	각 문서는 어떤 토픽을 가지고 있는가?

Prepared Data

LDA	k-means	의미
topic coordinate	cluster coordinate	토픽/군집 벡터가 2 차원으로 축소된 값
$P(w_j t)$	$p_i(w_j)$	각 토픽/군집에서 단어 w_j 가 얼마나 자주 등장하는가?

$$\frac{P(w_j | t)}{P(w_j)}$$

$$\frac{p_i(w_j)}{p_i(w_j) + p_{-i}(w_j)}$$

다른 토픽/군집보다 단어 w_j 가 유독 자주 등장하는가?

k-means + pyLDavis

```
# spherical k-means
```

```
from soyclustering import SphericalKMeans  
from soyclustering import proportion_keywords
```

```
spherical_kmeans = SphericalKMeans(n_clusters=1000, max_iter=10, verbose=1, init='similar_cut')  
labels = spherical_kmeans.fit_predict(x)
```

```
# k-means + pyLDavis
```

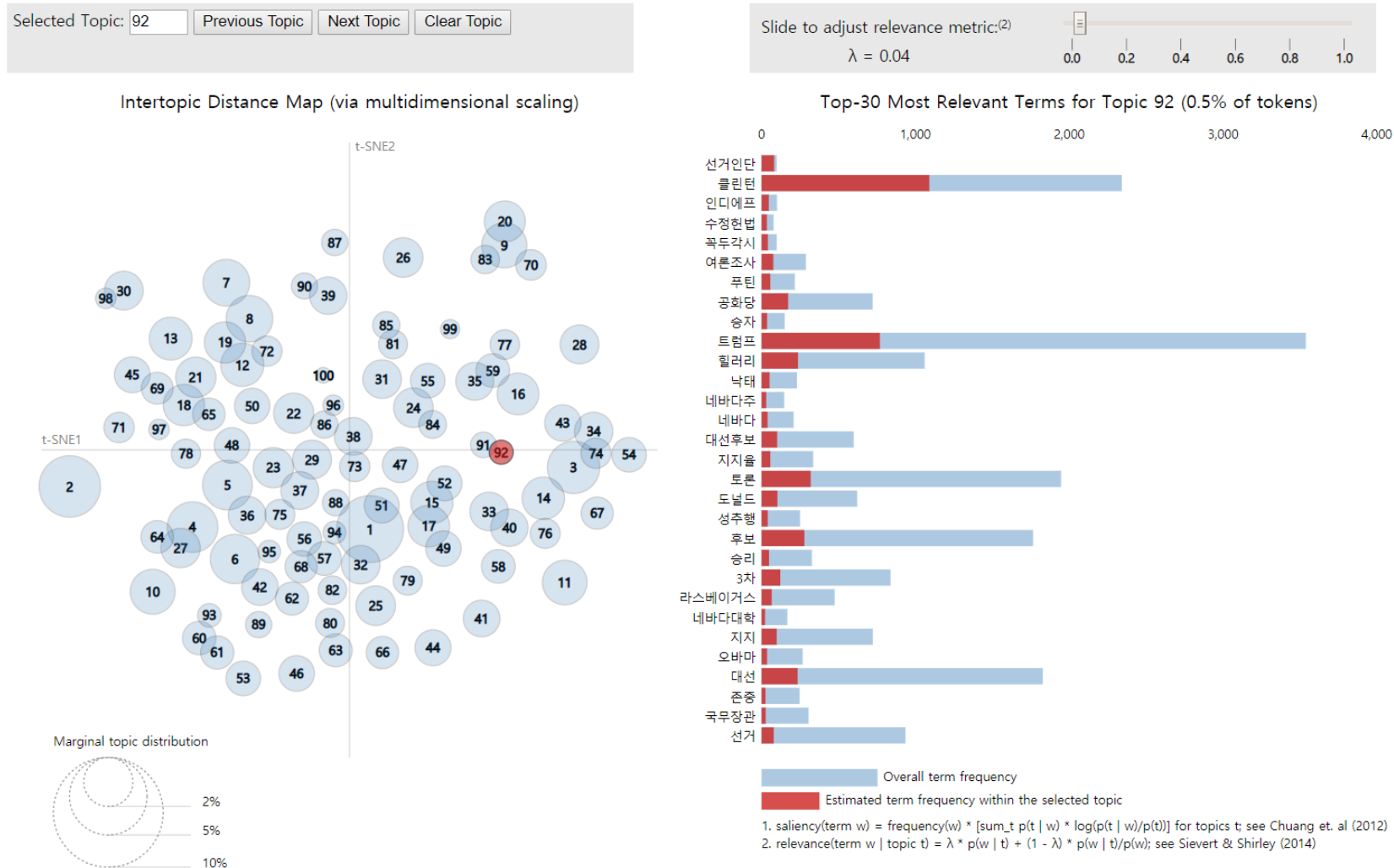
```
import pyLDavis  
from kmeans_visualizer import kmeans_to_prepared_data
```

```
prepared_data = kmeans_to_prepared_data(x, index2word, centers, labels)  
pyLDavis.display(prepared_data)
```

https://github.com/lovit/kmeans_to_pyLDavis

k-means + pyLDAvis

- *k*-means 도 시각화하여 이해할 수 있습니다



Summary

-
- 문서 군집화를 위해서는 Cosine distance 를 이용하는 Spherical k-means 를 이용해야 합니다.
 - 고차원의 sparse vector 에 대해서는 k -means++ 이 잘 작동하지 않습니다. 효율적인 k -means initializer 를 제안합니다.

-
- 일단 k 는 크게 설정해야 작은 군집들이 살아남습니다. 중복된 군집은 후처리로 뭉칩니다.
 - Centroids 만 이용해도 군집의 레이블을 자동으로 달 수 있습니다.
 - pyLDAvis 를 이용하면 k-means 의 군집화 결과도 시각적으로 살펴볼 수 있습니다.