Document Retrieval& Classification and Clustering

9조 강명훈 김영헌 김원제

목차 table of contents

1

Document Retrieval

Part 1-1

Get search Engine result-Document

```python
#%%
from nltk.corpus import stopwords
from nltk.tokenize import RegexpTokenizer
from nltk import Text


stopwords = set(stopwords.words('english'))
tokenizer = RegexpTokenizer(r"\w+")

with open('doc/document.txt', 'r', encoding='utf-8') as f:
    doc = f.read()
    doc = tokenizer.tokenize(doc)
    docwords = [word.lower() for word in doc if word.lower() not in stopwords]
    nostop = Text(docwords)
    a = nostop.vocab()
    print(a.most_common(200))
```

code

>

```
[('patients', 10694), ('0', 5523), ('1', 4198), ('2', 3465), ('treatment', 2891), ('p', 2777), ('clinical', 2714), ('study', 2712), ('patient', 2682)
('acute', 2442), ('risk', 2428), ('results', 2419), ('3', 2401), ('disease', 2327), ('case', 2259), ('5', 2130), ('associated', 2023), ('diagnosis', 1974)
('may', 1784), ('methods', 1769), ('group', 1764), ('6', 1743), ('4', 1666), ('year', 1575), ('years', 1507), ('cases', 1505), ('pain', 1449), ('therapy'
1444), ('high', 1422), ('background', 1373), ('coronary', 1373), ('7', 1344), ('8', 1328), ('mortality', 1305), ('two', 1279), ('pregnancy', 1273)
('abdominal', 1272), ('age', 1262), ('significant', 1230), ('conclusion', 1223), ('symptoms', 1223), ('old', 1216), ('95', 1211), ('one', 1208)
('care', 1199), ('management', 1193), ('9', 1192), ('common', 1160), ('blood', 1154), ('hospital', 1152), ('non', 1151), ('af', 1151), ('diabetes'
1133), ('factors', 1094), ('also', 1093), ('women', 1093), ('present', 1078), ('use', 1073), ('compared', 1071), ('used', 1066), ('infection', 1047)
('however', 1042), ('severe', 1040), ('performed', 1037), ('stroke', 1026), ('ci', 1018), ('rate', 1018), ('data', 1002), ('using', 1002), ('10', 1001)
('atrial', 979), ('report', 978), ('studies', 973), ('increased', 967), ('rare', 965), ('ct', 951), ('levels', 943), ('first', 941), ('complications', 941)
('artery', 940), ('showed', 939), ('days', 934), ('conclusions', 932), ('chronic', 929), ('significantly', 923), ('cause', 922), ('bleeding', 921)
('higher', 913), ('analysis', 912), ('without', 909), ('due', 907), ('cancer', 902), ('surgery', 897), ('early', 897), ('control', 889), ('syndrome'
836), ('lung', 831), ('well', 828), ('reported', 826), ('presented', 821), ('history', 818), ('mean', 815), ('treated', 809), ('time', 803), ('failure'
803), ('outcome', 788), ('type', 785), ('bowel', 784), ('based', 781), ('diagnostic', 779), ('presentation', 774), ('n', 772), ('medical', 771), ('total'
769), ('12', 766), ('groups', 758), ('heart', 756), ('review', 755), ('cardiac', 753), ('outcomes', 748), ('including', 745), ('fibrillation', 745)
('findings', 738), ('among', 738), ('small', 738), ('related', 735), ('pulmonary', 734), ('found', 732), ('positive', 731), ('revealed', 720), ('months'
719), ('chest', 717), ('left', 716), ('day', 715), ('low', 713), ('injury', 706), ('included', 705), ('lower', 700), ('surgical', 700), ('vs', 699)
('emergency', 696), ('primary', 692), ('pancreatitis', 678), ('incidence', 668), ('renal', 664), ('weeks', 663), ('obstruction', 662), ('follow', 661)
('normal', 649), ('insulin', 645), ('30', 642), ('diagnosed', 641), ('health', 641), ('respectively', 640), ('level', 630), ('evidence', 626), ('life', 623)
('gastrointestinal', 620), ('three', 618), ('important', 609), ('respiratory', 600), ('long', 600), ('following', 598), ('imaging', 597), ('serum', 586)
('developed', 581), ('pressure', 576), ('within', 575), ('new', 573), ('001', 572), ('glucose', 571), ('term', 568), ('underwent', 567), ('mg', 567)
('period', 565), ('population', 559), ('test', 552), ('infections', 545), ('although', 544), ('urinary', 542), ('effects', 541), ('15', 524), ('function'
522), ('11', 522), ('right', 521), ('score', 517), ('drug', 516), ('literature', 515), ('major', 509), ('events', 508), ('multiple', 507), ('rates', 506)
('tract', 499), ('condition', 497), ('number', 497), ('considered', 496), ('20', 495), ('diseases', 492), ('cell', 488), ('18', 485)]
```

200 most common Word list

Document.txt 속 사용된 단어 중 많이 사용된 상위 200개 단어 List 작성

two, 7, 8등 숫자 발견

Part 1-1

Get search Engine result-Document

```
stopWords = set(stopwords.words('english'))
stopWords.update(['patients', 'treatment', 'p', 'clinical', 'study', 'patient', 'results'])
stopWords.update(['case','associated', 'diagnosis', 'may', 'methods', 'group', 'diseases'])
stopWords.update(['year', 'years', 'cases', 'pain', 'therapy', 'background', 'two', 'age'])
stopWords.update(['significant', 'conclusion', 'old', '95', 'one', 'care', 'management', 'common'])
stopWords.update(['hospital', 'non', 'af', 'factors', 'also', 'present', 'use'])
stopWords.update(['compared', 'used', 'however', 'severe', 'performed', 'rate', 'data', 'using',])
stopWords.update(['report', 'studies', 'rare', 'showed', 'days', 'conclusions', 'significantly', 'cause', 'higher', 'analysis'])
stopWords.update(['without', 'due', 'well', 'reported', 'presented', 'history', 'mean', 'treated', 'time'])
stopWords.update(['failure', 'outcome', 'type', 'based', 'presentation', 'n', 'medical', 'total', 'groups'])
stopWords.update(['review', 'outcomes', 'including', 'findings', 'among', 'small', 'related', 'found', 'positive', 'revealed'])
stopWords.update(['months', 'day', 'low', 'injury', 'included', 'lower', 'vs', 'primary', 'incidence'])
stopWords.update(['weeks', 'follow', 'normal', 'within', 'new', 'term', 'underwent'])
stopWords.update(['period', 'population', 'test', 'although', 'effects', 'function', 'right', 'score'])
stopWords.update(['literature', 'major', 'events', 'multiple', 'rates', 'condition', 'number', 'considered'])

schema = Schema(docID=NUMERIC(stored=True),
                contents=TEXT(analyzer=StemmingAnalyzer(stoplist=stopWords))
                )
```

\>

단어 빈도 분석으로 자주 나오는 단어 중 모호한 단어 **stop words** 제거

Disease, risk, patient, 숫자 etc

Stop word list

```
def tagToPos(tag):
    if 'JJ' in tag:
        return 'a'
    elif 'NN' in tag:
        return 'n'
    elif 'VB' in tag:
        return 'v'
    elif 'RB' in tag:
        return 'r'
    else:
        return None

def importantTag(tag):
    if 'JJ' in tag:
        return True
    elif 'NN' in tag:
        return True
    elif 'VB' in tag:
        return True
    elif 'RB' in tag:
        if tag == 'WRB':
            return False
        return True
    return False
index_dir = "index"

if not os.path.exists(index_dir):
    os.makedirs(index_dir)

ix = create_in(index_dir, schema)
writer = ix.writer()

lm = WordNetLemmatizer()
```

```
with open('./doc/document.txt', 'r', encoding='UTF-8') as f:
    text = f.read()
    docs = text.split('////\n')[:-1]

    for doc in docs:
        br = doc.find('\n')
        docID = int(doc[:br])
        doc_text = doc[br+1:]

        tagged_list = pos_tag(word_tokenize(doc_text))
        new_doc_text = ''
        for (word, tag) in tagged_list:
            if not importantTag(tag):
                new_doc_text += '//////////' + ' '
                continue
            word = lm.lemmatize(word, pos=tagToPos(tag))
            if '-' in word:
                for w in word.split('-'):
                    new_doc_text += w + ' '
            else:
                new_doc_text += word + ' '
```

Pos tagging

\>

word tokenize 후 pos tag list에서

lemmatizing important tag, 하이픈 처리

Part 1

Get search Engine result-Query

RegexpTokenizer

WordNetLemmatizer vs stemming

Stopwords 동일

Query handling

Parser = QueryParser(" contents " , schema=ix.schema, group=OrGroup.factory(0.9))

Pos tagging 동일

Query 수정

>

```
for qid, q in query_dict.items():
    new_q = ''

    sentence = q.lower()
    tagged_list = pos_tag(retokenize.tokenize(sentence))

    for (word, tag) in tagged_list:
        if not importantTag(tag):
            continue
        if word in stopWords:
            continue
        word = lemmatizer.lemmatize(word, pos=tagToPos(tag))
        if '-' in word:
            for w in word.split('-'):
                new_q += w + ' '
        else: new_q += word + ' '
    query = parser.parse(new_q.lower())

    results = searcher.search(query, limit=None)
    result_dict[qid] = [result.fields()['docID'] for result in results]
```

Query parsing code

Part 2-1

Scoring function

```python
def bm25L(idf, tf, fl, avgfl, K1, B, eps):
    m = tf/(1 - B + B * fl/avgfl)
    return idf * (K1 + 1) * (m + eps) / (K1 + m + eps)

def bm25_plus(idf, tf, fl, avgfl, K1, B, eps):
    return idf * (tf * (K1 + 1) / (K1 * (1 - B + B * (fl / avgfl)) + tf) + eps)
```

BM 25 L , BM25 +[1][2]

eps = smoothing
B = 문서길이 무시
K 1= 특정 수치 이상의 용어 빈도가 점수에 미치는 영향을 억제

B = 0.7, K1 = 3, eps = 0

BEST score

Scoring function 선택

Random seed(6) 10개 쿼리 평균 점수

```
✓ import numpy as np ···
0.30925283713524737
```

\>

모든 쿼리 평균 점수

```
✓ import numpy as np ···
0.28266054927422096
```

BPREF Scoring 결과

1[] Trotman, Andrew, Antti Puurula, and Blake Burgess. "Improvements to BM25 and language models examined." *Proceedings of the 19th Australasian Document Computing Symposium*. 2014.

[2] Lv, Yuanhua, and ChengXiang Zhai. "Lower-bounding term frequency normalization." *Proceedings of the 20th ACM international conference on Information and knowledge management*. 2011

2

Classification & Clusterin

Part 2-1

NaiveBayes Classifier

Metric	algebraic geometry	computer vision	general economics	quantitative biology	quantum physics	statistics theory	Overall
Precision	0.92	0.81	0.67	0.62	0.66	0.57	0.71
Recall	0.78	0.74	0.79	0.71	0.70	0.51	0.71
F1-Score	0.85	0.77	0.72	0.66	0.68	0.54	0.71
Support (샘플 개수)	88	80	80	75	80	77	480

GausianNB성능평가

> **(Accuracy): 71% (339/480)**

Metric	algebraic geometry	computer vision	General economics	quantitative biology	quantum physics	statistics theory	Overall
Precision	0.95	0.82	0.90	0.85	0.83	0.61	0.81
Recall	0.93	0.91	0.70	0.69	0.81	0.82	0.81
F1-Score	0.94	0.86	0.79	0.76	0.82	0.70	0.81
Support (샘플 개수)	88	80	80	75	80	77	480

MultinomialNB 성능평가

> **(Accuracy): 81% (391/480)**

NaiveBayes Classifier

GausianNB MultinomialNB BernouliNB ComplementNB 비교

CategoricalNB 범주형x

S

NaiveBayes Classifier

Metric	algebraic geometry	computer vision	general economics	quantitative biology	quantum physics	statistics theory	Overall
Precision	0.82	0.85	0.86	0.85	0.77	0.58	0.78
Recall	0.97	0.84	0.75	0.63	0.71	0.74	0.78
F1-Score	0.89	0.84	0.80	0.72	0.74	0.65	0.78
Support (샘플 개수)	88	80	80	75	80	77	480

BernoulliNB 성능평가

> (Accuracy): 78% (373/480)

Metric	algebraic geometry	computer vision	general economics	quantitative biology	quantum physics	statistics theory	Overall
Precision	0.90	0.82	0.88	0.87	0.83	0.70	0.83
Recall	0.97	0.91	0.79	0.72	0.85	0.74	0.83
F1-Score	0.93	0.86	0.83	0.79	0.84	0.72	0.83
Support (샘플 개수)	88	80	80	75	80	77	480

Complete NB 성능평가

> (Accuracy): 83% (400/480)

text를 분류하는 데 특화된 MultinomialNB와 ComplementNB가 좋은 성능

Feature extraction 개선

NaiveBayes Classifier

Min_df: 3~8 (1)

Max_df: 0.1~1 (0.1)

Stopword: default or ‘English’

Max feature: 500~2500 (500)

Ngram = (1,1) or (2,2)

Countervectorizer() 파라미터 조건

>

NaiveBayes Max accuracy = 0.833

ComplementNB	2	1500	4	0.5	(1, 1)
ComplementNB	2	1500	4	0.6	(1, 1)
ComplementNB	2	1500	4	0.7	(1, 1)
ComplementNB	2	1500	4	0.8	(1, 1)
ComplementNB	2	1500	4	0.9	(1, 1)
ComplementNB	2	1500	4	1.0	(1, 1)
ComplementNB	2	2500	7	0.3	(1, 1)
ComplementNB	2	2500	7	0.4	(1, 1)
ComplementNB	2	2500	7	0.5	(1, 1)
ComplementNB	2	2500	7	0.6	(1, 1)
ComplementNB	2	2500	7	0.7	(1, 1)
ComplementNB	2	2500	7	0.8	(1, 1)
ComplementNB	2	2500	7	0.9	(1, 1)
ComplementNB	2	2500	7	1.0	(1, 1)

NaiveBayes Max accuracy Cases

SVM

SVM 분류

SVC vs NuSVC vs LinearSVC

Classifier	C	Nu	Kernel	Gamma
SVC	0,1 or 1 or 10		Linear or RBF	Scale or Auto
NuSVC		0,1 or 1 or 10	Linear or RBF	Scale or Auto
LinearSVC	0,1 or 1 or 10			

파라미터 조건

>

max accuracy 0.8458

Classifier	C	Nu	Kernel	Gamma
SVC	10		Linear	Scale
SVC	10		Linear	Auto
NuSVC		0.1	Linear	Scale
NuSVC		0.1	Linear	Auto

Max accuracy Cases

파라미터 조작 결과 **SVC(c=10, kernel=linear), NuSVC(Nu=0.1, kernel=linear)**이
가장 좋은 성능을 보임

SVM

Min_df: 3~8 (1)

Max_df: 0.1~1 (0.1)

Stopword: default or ‘English’

Max feature: 500~2500 (500)

Ngram = (1,1) or (2,2)

NB와 동일한 Countervectorizer() 파라미터 조건

>

SVM Max accuracy = 0.833

Classifier	Stop word	Max feature	Min df	Max df	ngram range
SVC	None	2500	4	1.0	(1,1)
NuSVC	None	2500	4	1.0	(1,1)

SVM Max accuracy Cases

K-means clustering

1.최적 군집 개수, 초기 클러스터 중심 설정 반복횟수,
파라미터당 반복 횟수 설정

n_cluster_ranger: 3~9

n_init: 5~50

파라미터 조건

n_iter: 20

>

최적의 V_measure = 0.51878

최적의 파라미터: {'n_clusters': 6, 'n_init': 10}

2. {'n_clusters': 6, 'n_init': 10} 에 대해서 파라미터 변경

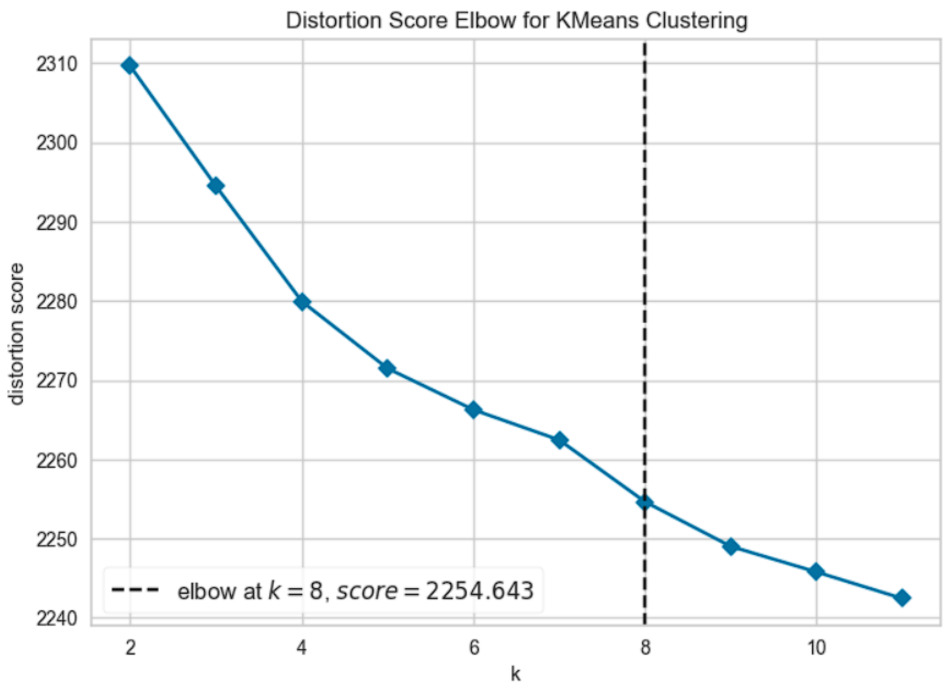
```
max_features_range = [500, 1000,
1500, 2000, 2500]
min_df_range = [1, 2, 3, 4, 5, 6]
max_df_range = [0.4, 0.5, 0.6, 0.7,
0.8, 0.9, 1.0]
```

>

최적의 V-measure: 0.53928,

최적의 특징 추출기 파라미터:
{'max_features': 2000, 'min_df': 3, 'max_df': 0.4}

K-means clustering

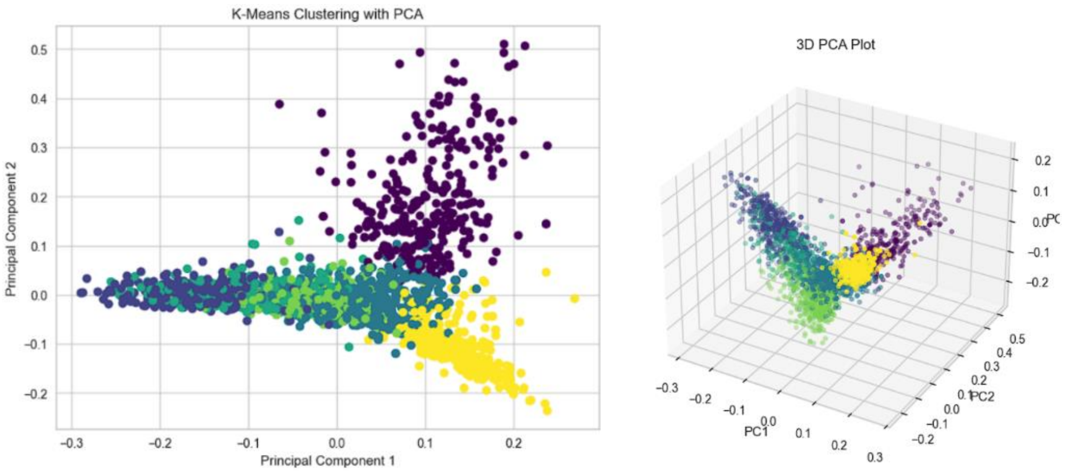


Elbow 최적 cluster = 8

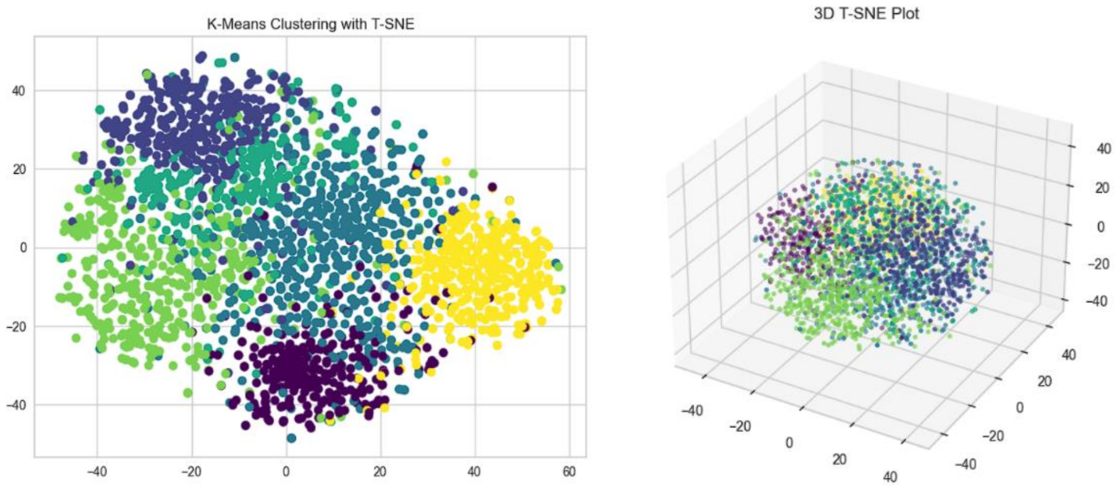
>

Elbow Method는 관성(WCSS)을 기준으로 최적의 클러스터 수를 찾으려는 시도이기 때문에 **v-measure**을 기준으로 따져 본 **6개의 클러스터**가 더 좋은 성능을 보인다고 할 수 있다
주어진 data의 카테고리가 6개이므로 클러스터의 수가 6개인 것이 합리적이다.

K-means clustering



Pca 시각화



T-NSE 시각화

감사합니다