Implemenation-LSA

March 21, 2024

[44]:
```
from gensim import corpora, models, similarities
```

[45]:
```
documents = ["Human machine interface for lab computer applications",
             "A survey of user opinion of computer system response time",
             "The EPS user interface management system",
             "System and human system engineering testing of EPS",
             "Relation of user perceived response time to error measurement",
             "The generation of random binary unordered trees",
             "The intersection graph of paths in trees",
             "Graph minors IV Widths of trees and well quasi ordering",
             "Graph minors A survey"]
```

[46]:
```
print(documents)
```

```
['Human machine interface for lab computer applications', 'A survey of user
opinion of computer system response time', 'The EPS user interface management
system', 'System and human system engineering testing of EPS', 'Relation of user
perceived response time to error measurement', 'The generation of random binary
unordered trees', 'The intersection graph of paths in trees', 'Graph minors IV
Widths of trees and well quasi ordering', 'Graph minors A survey']
```

[47]:
```
# remove common words and tokenize them
stoplist = set('for a of the and to in'.split())
```

[48]:
```
texts = [[word for word in document.lower().split() if word not in stoplist]␣
 ↪for document in documents]
```

[49]:
```
print(texts)
```

```
[['human', 'machine', 'interface', 'lab', 'computer', 'applications'],
['survey', 'user', 'opinion', 'computer', 'system', 'response', 'time'], ['eps',
'user', 'interface', 'management', 'system'], ['system', 'human', 'system',
'engineering', 'testing', 'eps'], ['relation', 'user', 'perceived', 'response',
'time', 'error', 'measurement'], ['generation', 'random', 'binary', 'unordered',
'trees'], ['intersection', 'graph', 'paths', 'trees'], ['graph', 'minors', 'iv',
'widths', 'trees', 'well', 'quasi', 'ordering'], ['graph', 'minors', 'survey']]
```

[50]:
```
# remove words those appear only once
all_tokens = sum(texts, [])
```

```
print(all_tokens)
```

```
['human', 'machine', 'interface', 'lab', 'computer', 'applications', 'survey',
'user', 'opinion', 'computer', 'system', 'response', 'time', 'eps', 'user',
'interface', 'management', 'system', 'system', 'human', 'system', 'engineering',
'testing', 'eps', 'relation', 'user', 'perceived', 'response', 'time', 'error',
'measurement', 'generation', 'random', 'binary', 'unordered', 'trees',
'intersection', 'graph', 'paths', 'trees', 'graph', 'minors', 'iv', 'widths',
'trees', 'well', 'quasi', 'ordering', 'graph', 'minors', 'survey']
```

[51]:
```
tokens_once = set(word for word in set(all_tokens) if all_tokens.count(word)␣
↪==1)

print(tokens_once)
```

```
{'error', 'management', 'generation', 'perceived', 'random', 'quasi',
'relation', 'widths', 'machine', 'paths', 'measurement', 'opinion', 'lab',
'unordered', 'testing', 'engineering', 'intersection', 'well', 'binary', 'iv',
'ordering', 'applications'}
```

[52]:
```
texts = [[word for word in text if word not in tokens_once]
         for text in texts]

print(texts)
```

```
[['human', 'interface', 'computer'], ['survey', 'user', 'computer', 'system',
'response', 'time'], ['eps', 'user', 'interface', 'system'], ['system', 'human',
'system', 'eps'], ['user', 'response', 'time'], ['trees'], ['graph', 'trees'],
['graph', 'minors', 'trees'], ['graph', 'minors', 'survey']]
```

[53]:
```
dictionary = corpora.Dictionary(texts)

print(dictionary)
```

```
Dictionary<12 unique tokens: ['computer', 'human', 'interface', 'response',
'survey']…>
```

[54]:
```
dictionary.save('/home/nmit/Documents/deerwester.dict')  # save as binary file␣
↪at the dictionary at local directory
```

[55]:
```
dictionary.save_as_text('/home/nmit/Documents/deerwester_text.dict')  # save as␣
↪text file at the local directory
```

[56]:
```
print(dictionary.token2id) # show pairs of "word : word-ID number"
```

```
{'computer': 0, 'human': 1, 'interface': 2, 'response': 3, 'survey': 4,
'system': 5, 'time': 6, 'user': 7, 'eps': 8, 'trees': 9, 'graph': 10, 'minors':
11}
```

[57]:
```
new_doc = "Human computer interaction" # temporary data to see role of below␣
↪function

new_vec = dictionary.doc2bow(new_doc.lower().split()) # return "word-ID :␣
↪Frequency of appearance""
print(new_vec)
```

```
[(0, 1), (1, 1)]
```

[58]:
```
corpus = [dictionary.doc2bow(text) for text in texts]

print(corpus)
```

```
[[(0, 1), (1, 1), (2, 1)], [(0, 1), (3, 1), (4, 1), (5, 1), (6, 1), (7, 1)],
[(2, 1), (5, 1), (7, 1), (8, 1)], [(1, 1), (5, 2), (8, 1)], [(3, 1), (6, 1), (7,
1)], [(9, 1)], [(9, 1), (10, 1)], [(9, 1), (10, 1), (11, 1)], [(4, 1), (10, 1),
(11, 1)]]
```

[59]:
```
corpora.MmCorpus.serialize('deerwester.mm', corpus) # save corpus at local␣
↪directory
```

[60]:
```
corpus = corpora.MmCorpus('deerwester.mm') # try to load the saved corpus from␣
↪local

print(list(corpus)) # to show corpus which was read above, need to print(list(␣
↪))
```

```
[[(0, 1.0), (1, 1.0), (2, 1.0)], [(0, 1.0), (3, 1.0), (4, 1.0), (5, 1.0), (6,
1.0), (7, 1.0)], [(2, 1.0), (5, 1.0), (7, 1.0), (8, 1.0)], [(1, 1.0), (5, 2.0),
(8, 1.0)], [(3, 1.0), (6, 1.0), (7, 1.0)], [(9, 1.0)], [(9, 1.0), (10, 1.0)],
[(9, 1.0), (10, 1.0), (11, 1.0)], [(4, 1.0), (10, 1.0), (11, 1.0)]]
```

[62]:
```
dictionary = corpora.Dictionary.load('/home/nmit/Documents/deerwester.dict') #␣
↪try to load saved dic.from local

print(dictionary)
```

```
Dictionary<12 unique tokens: ['computer', 'human', 'interface', 'response',
'survey']…>
```

[63]:
```
print(corpus)
```

```
MmCorpus(9 documents, 12 features, 28 non-zero entries)
```

[64]:
```
tfidf = models.TfidfModel(corpus) # step 1 -- initialize a model

print(tfidf)
```

```
TfidfModel<num_docs=9, num_nnz=28>
```

[65]:
```
corpus_tfidf = tfidf[corpus]  # map corpus object into tfidf space

print(corpus_tfidf)
```

```
<gensim.interfaces.TransformedCorpus object at 0x72aa4ed8d450>
```

[66]:
```
for doc in corpus_tfidf: # show tfidf-space mapped words
    print(doc)
```

```
[(0, 0.5773502691896257), (1, 0.5773502691896257), (2, 0.5773502691896257)]
[(0, 0.44424552527467476), (3, 0.44424552527467476), (4, 0.44424552527467476),
(5, 0.3244870206138555), (6, 0.44424552527467476), (7, 0.3244870206138555)]
[(2, 0.5710059809418182), (5, 0.4170757362022777), (7, 0.4170757362022777), (8,
0.5710059809418182)]
[(1, 0.49182558987264147), (5, 0.7184811607083769), (8, 0.49182558987264147)]
[(3, 0.6282580468670046), (6, 0.6282580468670046), (7, 0.45889394536615247)]
[(9, 1.0)]
[(9, 0.7071067811865475), (10, 0.7071067811865475)]
[(9, 0.5080429008916749), (10, 0.5080429008916749), (11, 0.695546419520037)]
[(4, 0.6282580468670046), (10, 0.45889394536615247), (11, 0.6282580468670046)]
```

[67]:
```
lsi = models.LsiModel(corpus_tfidf, id2word=dictionary, num_topics=2) #␣
 ↪initialize LSI
print(lsi)
```

```
LsiModel<num_terms=12, num_topics=2, decay=1.0, chunksize=20000>
```

[68]:
```
corpus_lsi = lsi[corpus_tfidf] # create a double wrapper over the original␣
 ↪corpus
print(corpus_lsi)
```

```
<gensim.interfaces.TransformedCorpus object at 0x72aa4d38d7d0>
```

[69]:
```
topic = lsi.print_topics(2)
```

[70]:
```
print(topic)
```

```
[(0, '0.703*"trees" + 0.538*"graph" + 0.402*"minors" + 0.187*"survey" +
0.061*"system" + 0.060*"time" + 0.060*"response" + 0.058*"user" +
0.049*"computer" + 0.035*"interface"'), (1, '-0.460*"system" + -0.373*"user" +
-0.332*"eps" + -0.328*"interface" + -0.320*"time" + -0.320*"response" +
-0.293*"computer" + -0.280*"human" + -0.171*"survey" + 0.161*"trees"')]
```

[71]:
```
for doc in corpus_lsi:
    print(doc)
```

```
[(0, 0.06600783396090407), (1, -0.5200703306361849)]
[(0, 0.1966759285914261), (1, -0.7609563167700046)]
[(0, 0.08992639972446498), (1, -0.724186062675251)]
[(0, 0.0758584765217824), (1, -0.6320551586003431)]
```

```
[(0, 0.10150299184980247), (1, -0.5737308483002955)]
[(0, 0.7032108939378311), (1, 0.16115180214025884)]
[(0, 0.8774787673119829), (1, 0.1675890686465952)]
[(0, 0.9098624686818575), (1, 0.14086553628719123)]
[(0, 0.6165825350569278), (1, -0.05392907566389308)]
```

[72]:
```
lsi.save('/home/nmit/Documents/model.lsi')  # save output model at local␣
↪directory
```

[73]:
```
lsi = models.LsiModel.load('/home/nmit/Documents/model.lsi') # try to load␣
↪above saved model
print(lsi)
```

```
LsiModel<num_terms=12, num_topics=2, decay=1.0, chunksize=20000>
```

[74]:
```
doc = "Human computer interaction"  # give new document to calculate similarity␣
↪degree with already obtained topics

vec_bow = dictionary.doc2bow(doc.lower().split())  # put newly obtained␣
↪document to existing dictionary object
print(vec_bow)  # show result of above
```

```
[(0, 1), (1, 1)]
```

[75]:
```
vec_lsi = lsi[vec_bow] # convert new document (henceforth, call it "query") to␣
↪LSI space
print(vec_lsi)
```

```
[(0, 0.07910475117444937), (1, -0.5732835243079403)]
```

[76]:
```
index = similarities.MatrixSimilarity(lsi[corpus]) # transform corpus to LSI␣
↪space and indexize it
print(index)
```

```
MatrixSimilarity<9 docs, 2 features>
```

[77]:
```
index.save('deerwester.index') # save index object at local directory
```

[78]:
```
index = similarities.MatrixSimilarity.load('deerwester.index')
```

[79]:
```
print(index)
```

```
MatrixSimilarity<9 docs, 2 features>
```

[80]:
```
sims = index[vec_lsi] # calculate degree of similarity of the query to existing␣
↪corpus
print(sims)
```

```
[ 0.9999408   0.9946708   0.9999428   0.999879    0.99935204 -0.08804217
 -0.0515742  -0.02366471  0.1938726 ]
```

[81]:
```
print(list(enumerate(sims))) # output (document_number , document similarity)
```

```
[(0, 0.9999408), (1, 0.9946708), (2, 0.9999428), (3, 0.999879), (4, 0.99935204),
(5, -0.08804217), (6, -0.0515742), (7, -0.023664713), (8, 0.1938726)]
```

[82]:
```
sims = sorted(enumerate(sims), key=lambda item: -item[1])  # sort output object␣
 ↪as per similarity ( largest similarity document comes first )
print(sims)
```

```
[(2, 0.9999428), (0, 0.9999408), (3, 0.999879), (4, 0.99935204), (1, 0.9946708),
(8, 0.1938726), (7, -0.023664713), (6, -0.0515742), (5, -0.08804217)]
```

[ ]:
```

```