Building an Ensemble LDA model with Gensim

我的編輯在最後面

```
開始使用 AI 編寫或生成程式碼。
```

Mount your google drive

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

Preprocessing the training data

Let’s load the training data:

```
import pandas as pd
import numpy as np
pd.set_option('display.max_colwidth', None)
path = "/content/drive/My Drive/Colab Notebooks/Dataset"
train = pd.read_csv(path + "/ag_news_train.csv")
```

We will tokenize the words in each sentence:

```
!pip install pyldavis

from gensim.parsing.preprocessing import preprocess_string
text_tokenized = []
for doc in train['Description']:
   k = preprocess_string(doc)
   text_tokenized.append(k)
text_tokenized[0:3]
```

```
Collecting pyldavis
  Downloading pyLDAvis-3.4.1-py3-none-any.whl.metadata (4.2 kB)
Requirement already satisfied: numpy>=1.24.2 in /usr/local/lib/python3.12/dist-packages (from pyldavis) (2.0.2)
Requirement already satisfied: scipy in /usr/local/lib/python3.12/dist-packages (from pyldavis) (1.16.3)
Requirement already satisfied: pandas>=2.0.0 in /usr/local/lib/python3.12/dist-packages (from pyldavis) (2.2.2)
Requirement already satisfied: joblib>=1.2.0 in /usr/local/lib/python3.12/dist-packages (from pyldavis) (1.5.2)
Requirement already satisfied: jinja2 in /usr/local/lib/python3.12/dist-packages (from pyldavis) (3.1.6)
Requirement already satisfied: numexpr in /usr/local/lib/python3.12/dist-packages (from pyldavis) (2.14.1)
Collecting funcy (from pyldavis)
  Downloading funcy-2.0-py2.py3-none-any.whl.metadata (5.9 kB)
Requirement already satisfied: scikit-learn>=1.0.0 in /usr/local/lib/python3.12/dist-packages (from pyldavis) (1
Collecting gensim (from pyldavis)
  Downloading gensim-4.4.0-cp312-cp312-manylinux_2_24_x86_64.manylinux_2_28_x86_64.whl.metadata (8.4 kB)
Requirement already satisfied: setuptools in /usr/local/lib/python3.12/dist-packages (from pyldavis) (75.2.0)
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.12/dist-packages (from pandas>=2
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.12/dist-packages (from pandas>=2.0.0->pylc
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.12/dist-packages (from pandas>=2.0.0->py
Requirement already satisfied: threadpoolctl>=3.1.0 in /usr/local/lib/python3.12/dist-packages (from scikit-lear
Requirement already satisfied: smart_open>=1.8.1 in /usr/local/lib/python3.12/dist-packages (from gensim->pyldav
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.12/dist-packages (from jinja2->pyldavis
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.12/dist-packages (from python-dateutil>=2.8.2-
Requirement already satisfied: wrapt in /usr/local/lib/python3.12/dist-packages (from smart_open>=1.8.1->gensim-
Downloading pyLDAvis-3.4.1-py3-none-any.whl (2.6 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 2.6/2.6 MB 100.6 MB/s eta 0:00:00
Downloading funcy-2.0-py2.py3-none-any.whl (30 kB)
Downloading gensim-4.4.0-cp312-cp312-manylinux_2_24_x86_64.manylinux_2_28_x86_64.whl (27.9 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 27.9/27.9 MB 98.3 MB/s eta 0:00:00
Installing collected packages: funcy, gensim, pyldavis
Successfully installed funcy-2.0 gensim-4.4.0 pyldavis-3.4.1
[['reuter',
  'short',
  'seller',
  'wall',
  'street',
  'dwindl',
  'band',
  'ultra',
  'cynic',
  'see',
  'green'],
 ['reuter',
  'privat',
  'invest',
  'firm',
  'carlyl',
  'group',
  'reput',
  'make',
  'time',
  'occasion',
  'controversi',
  'plai',
  'defens',
  'industri',
  'quietli',
  'place',
  'bet',
```

Creating text representation with BOW and TF-IDF

The text representation using BOW is done as follows:

```
from gensim.corpora import Dictionary
gensim_dictionary = Dictionary()
bow_corpus = [gensim_dictionary.doc2bow(doc, allow_update=True) for doc in text_tokenized]
print(bow_corpus[:3])
id_words = [[(gensim_dictionary[id], count) for id, count in line] for line in bow_corpus]
print(id_words)
```

```
[[(0, 1), (1, 1), (2, 1), (3, 1), (4, 1), (5, 1), (6, 1), (7, 1), (8, 1), (9, 1), (10, 1)], [(4, 1), (11, 1), (12
[[('band', 1), ('cynic', 1), ('dwindl', 1), ('green', 1), ('reuter', 1), ('see', 1), ('seller', 1), ('short', 1),
```

Likewise, the text representation using TF-IDF is done with the following code:

```
from gensim.models import TfidfModel
tfidf = TfidfModel(bow_corpus) #, smartirs='npu')
tfidf_corpus = tfidf[bow_corpus]
print(tfidf_corpus[:3])
id_words = [[(gensim_dictionary[id], count) for id, count in line] for line in bow_corpus]
print(id_words)
```

```
<gensim.interfaces.TransformedCorpus object at 0x7ae1e774a240>
[[('band', 1), ('cynic', 1), ('dwindl', 1), ('green', 1), ('reuter', 1), ('see', 1), ('seller', 1), ('short', 1),
```

```
開始使用 AI 編寫或生成程式碼。
```

Saving the dictionary

Later, when we use the model to score new documents, we will need the dictionary and the BOW or TF-IDF. So, we save the dictionary, BOW, and TF-IDF now:

```
from gensim.test.utils import datapath
dict_file = datapath(path + "/gensim_dictionary_AGnews")
gensim_dictionary.save(dict_file)
```

Gensim's corpora class offers the `MmCorpus()` function that saves matrix data in the Matrix Market (MM) exchange format. The MM format is a basic ASCII file format for matrix data. We will use it to

```
from gensim import corpora
# Save in the matrix market format
corpora.MmCorpus.serialize(path + "/BoW_AGnews_corpus.mm", bow_corpus)
# Load
bow_corpus = corpora.MmCorpus(path + "/BoW_AGnews_corpus.mm")
bow_corpus[0]
```

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

You can also use the Python pickle tool to dump and load the BOW or TF-IDF data:

```
import pickle
file = open(path + "/BoW_AGnews_corpus.pkl", 'wb')
pickle.dump(bow_corpus, file)
file.close()
# open a file, where you stored the pickled data
file = open(path + "/BoW_AGnews_corpus.pkl", 'rb')
bow_corpus = pickle.load(file)
# close the file
file.close()
bow_corpus[0]
```

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

Build the Ensemble LDA model

In LDA, the number of topics needs to be assigned externally as a hyperparameter. In Ensemble LDA, the number of topics will be determined internally. By running multiple LDA models, the Ensemble LDA model can extract stable topics that exist in multiple LDA models. Ensemble LDA has the benefit that we do not need to specify the exact number of topics:

```
from gensim.corpora.dictionary import Dictionary
from gensim.models import ensemblelda
# Although we will still set the number of topics,
# as shown later in the code, this determines the maximum number of topics,
# but not the final number of topics:
elda_bow = ensemblelda.EnsembleLda(corpus=bow_corpus, id2word=gensim_dictionary, num_topics=100)
elda_tfidf = ensemblelda.EnsembleLda(corpus=tfidf_corpus, id2word=gensim_dictionary, num_topics=100)
```

```
WARNING:gensim.models.ldamulticore:too few updates, training might not converge; consider increasing the number o
WARNING:gensim.models.ldamulticore:too few updates, training might not converge; consider increasing the number o
WARNING:gensim.models.ldamulticore:too few updates, training might not converge; consider increasing the number o
WARNING:gensim.models.ldamulticore:too few updates, training might not converge; consider increasing the number o
WARNING:gensim.models.ldamulticore:too few updates, training might not converge; consider increasing the number o
WARNING:gensim.models.ldamulticore:too few updates, training might not converge; consider increasing the number o
WARNING:gensim.models.ldamulticore:too few updates, training might not converge; consider increasing the number o
WARNING:gensim.models.ldamulticore:too few updates, training might not converge; consider increasing the number o
```

Although we will still set the number of topics, as shown later in the code, this determines the maximum number of topics, but not the final number of topics:

The training process can take a while. The best practice is to save the model for future use:

```
from gensim.test.utils import datapath
# Save the eLDA model trained on BOW data
elda_bow_file = datapath(path + "/eLDA_bow_AGnews")
elda_bow.save(elda_bow_file)
# Save the eLDA model trained on TF-IDF data
elda_tfidf_file = datapath(path + "/eLDA_tfidf_AGnews")
elda_tfidf.save(elda_tfidf_file)
```

The model identifies 20 topics automatically. Let’s see the topics with the following command:

```
import pprint as pp
pp.pprint(elda_bow.print_topics())
len(elda_bow.print_topics())
```

```
[(0,
  '0.031*"reuter" + 0.028*"fullquot" + 0.022*"stock" + 0.017*"investor" + '
  '0.015*"com" + 0.015*"target" + 0.015*"new" + 0.015*"ticker" + 0.015*"http" '
  '+ 0.015*"www"'),
 (1,
  '0.017*"said" + 0.014*"reuter" + 0.012*"new" + 0.008*"compani" + '
  '0.008*"quarter" + 0.007*"sale" + 0.007*"year" + 0.007*"percent" + '
  '0.006*"york" + 0.006*"stock"')]
2
```

Remember that this Ensemble LDA topic already removed the pseudo topics, leaving only the true and reliable topics. Since they are reliable topics, they are reproducible as well. Next, let's learn how to use the saved Ensemble LDA model to score new documents.

Scoring new documents

Let's use the Ensemble LDA model to score new documents. Suppose there are two new documents with meaningful words: ['champion', 'hockey', 'qualify'] and ['survey', 'tournament', and 'world']:

```
# Create a new corpus, made of previously unseen documents.
other_texts = [
    ['champion', 'hockei', 'qualifi'],
    ['survey', 'tournament', 'world']
]
# We will create a new corpus by applying the saved dictionary to these new documents:
other_corpus = [gensim_dictionary.doc2bow(text) for text in other_texts]
# Let's take the first document:
unseen_doc = other_corpus[0]
vector = elda_bow[unseen_doc]  # get topic probability distribution for a document
vector
```

```
[(0, np.float32(0.83856225)), (1, np.float32(0.16143776))]
```

Notice the words in the new document ['champion', 'hockei', 'qualifi'] are about sports tournaments. If we look at the previously printed output of elda_bow.print_topics(), we can guess this document belongs to topic 78. Indeed, when we print out the prediction for this new document, it is topic 78:

```
pp.pprint(vector)
```

```
[(0, np.float32(0.83856225)), (1, np.float32(0.16143776))]
```

[我的調整] 介面主題提示，檢查句子向量是否反映設計語氣

- 想法：把 UI 主題以色彩+質感描述，量測相似/最近鄰。
- 用法：可用於 embedding 搜尋或評估分類閾值。

```
# [我的調整] UI 色彩/材質主題描述，方便做語義搜尋 demo
custom_texts = [
    "午夜藍玻璃介面，按鈕帶細緻霓虹邊框",
    "沙岩米白搭配手寫體，營造輕手作感",
    "鈦金屬灰深色模式，字重偏細，適合專業儀表板",
    "薄荷綠卡片式佈局，加入微動效讓導覽更柔和",
    "夕陽橘漸層背景，標題用圓角字體強調溫度",
    "極地白留白搭配靛青重點色，凸顯內容",
    "霓虹紫暗色系聊天介面，訊息泡泡有內發光",
]
for i, text in enumerate(custom_texts, 1):
    print(f'示例 {i}: {text}')
print('可將 custom_texts 直接替換/插入到上方流程進行測試。')
```

```
示例 1: 午夜藍玻璃介面，按鈕帶細緻霓虹邊框
示例 2: 沙岩米白搭配手寫體，營造輕手作感
示例 3: 鈦金屬灰深色模式，字重偏細，適合專業儀表板
示例 4: 薄荷綠卡片式佈局，加入微動效讓導覽更柔和
示例 5: 夕陽橘漸層背景，標題用圓角字體強調溫度
示例 6: 極地白留白搭配靛青重點色，凸顯內容
示例 7: 霓虹紫暗色系聊天介面，訊息泡泡有內發光
可將 custom_texts 直接替換/插入到上方流程進行測試。
```

[我的調整-EnsembleLDA] 不同色彩情緒的短文

- 想法：測 ensemble 模型對多情緒小語料的穩定性。
- 用法：把 ensemble_docs 放進多模型跑，觀察主題分布差異。

```
# [我的調整-EnsembleLDA] 不同色彩情緒的短文
ensemble_docs = [
    "桃粉的公園午後讓人想慢跑",
    "石墨灰的加班夜充滿鍵盤聲",
    "琥珀橙的講座把觀眾氣氛炒熱",
    "靛青的圖書館自習室讓心跳放慢",
    "薄荷綠的瑜伽教室讓呼吸變得輕",
    "赭紅的演唱會現場鼓點強烈",
]
for i, text in enumerate(ensemble_docs, 1):
    print(f'示例 {i}: {text}')
print('可直接把這批文本丟進前面的處理/模型流程。')
```

```
示例 1: 桃粉的公園午後讓人想慢跑
示例 2: 石墨灰的加班夜充滿鍵盤聲
示例 3: 琥珀橙的講座把觀眾氣氛炒熱
示例 4: 靛青的圖書館自習室讓心跳放慢
示例 5: 薄荷綠的瑜伽教室讓呼吸變得輕
示例 6: 赭紅的演唱會現場鼓點強烈
可直接把這批文本丟進前面的處理/模型流程。
```

[我的調整-Ensemble跑法] 色彩情緒語料 quick run

- 用 ensemble_docs 跑 3 主題，複製字典並 allow_update 讓色彩詞存在。
- 加入 jieba 斷詞 fallback 與迭代次數，避免主題為空。

```
# [我的調整-Ensemble跑法] 色彩情緒語料 quick run
import re
from copy import deepcopy
from gensim.parsing.preprocessing import preprocess_string
from gensim.models import LdaModel

def tokenize_color(doc):
    try:
        import jieba
        tokens = [t.strip().lower() for t in jieba.lcut(doc) if t.strip()]
    except Exception:
        # Fallback if jieba is not available or fails
        tokens = [t.lower() for t in re.findall(r'[A-Za-z0-9]+', doc)]
        tokens += re.findall(r'[一-褁]', doc) # Add CJK characters

    if not tokens: # If tokens are still empty, use gensim's preprocess_string as a last resort
        tokens = preprocess_string(doc)
    return [t for t in tokens if t]

# 注意：gensim_dictionary 在此處沒有被定義，您可能需要先執行包含 gensim_dictionary 定義的儲存格。
# 這裡假設 gensim_dictionary 已經在其他地方被定義。
temp_dict = deepcopy(gensim_dictionary)
tokens = [tokenize_color(doc) for doc in ensemble_docs]
tokens = [t for t in tokens if t]
if not tokens:
    raise ValueError('色彩語料斷詞為空，請確認內容或安裝 jieba')
corpus = [temp_dict.doc2bow(t, allow_update=True) for t in tokens]

color_lda = LdaModel(corpus, num_topics=3, id2word=temp_dict, random_state=42, passes=40, iterations=300, alpha='a
for tid, terms in color_lda.print_topics():
    print(f'Topic {tid}: {terms}')
```

```
/usr/local/lib/python3.12/dist-packages/jieba/__init__.py:44: SyntaxWarning: invalid escape sequence '\.'
  re_han_default = re.compile("([\u4E00-\u9FD5a-zA-Z0-9+#&\._%\-]+)", re.U)
/usr/local/lib/python3.12/dist-packages/jieba/__init__.py:46: SyntaxWarning: invalid escape sequence '\s'
  re_skip_default = re.compile("(\r\n|\s)", re.U)
/usr/local/lib/python3.12/dist-packages/jieba/finalseg/__init__.py:78: SyntaxWarning: invalid escape sequence '\.
  re_skip = re.compile("([a-zA-Z0-9]+(?:\.\d+)?%?)")
Building prefix dict from the default dictionary ...
DEBUG:jieba:Building prefix dict from the default dictionary ...
Dumping model to file cache /tmp/jieba.cache
DEBUG:jieba:Dumping model to file cache /tmp/jieba.cache
Loading model cost 0.652 seconds.
DEBUG:jieba:Loading model cost 0.652 seconds.
Prefix dict has been built successfully.
DEBUG:jieba:Prefix dict has been built successfully.
/usr/local/lib/python3.12/dist-packages/gensim/models/ldamodel.py:851: RuntimeWarning: overflow encountered in ex
  perwordbound, np.exp2(-perwordbound), len(chunk), corpus_words
Topic 0: 0.000*"演唱" + 0.000*"鼓點" + 0.000*"赭紅" + 0.000*"現場" + 0.000*"會" + 0.000*"強烈" + 0.000*"的" + 0.000*"
Topic 1: 0.000*"的" + 0.000*"觀眾" + 0.000*"講座" + 0.000*"氣氛" + 0.000*"炒熱" + 0.000*"琥珀" + 0.000*"把" + 0.000*"
Topic 2: 0.000*"的" + 0.000*"讓" + 0.000*"呼吸" + 0.000*"輕" + 0.000*"教室" + 0.000*"瑜伽" + 0.000*"變得" + 0.000*"綠
```