Final Project - Latent Dirichlet Allocation (LDA)

Group number - 3 Sejal Vyas(10450395)

Shiwani Deo(10454959)

Parth Parab(10444835)

In this project, we have trained an LDA model using gensim. The document used to train this model is a research paper on Generative Adversarial Networks. The two test documents used are a research paper on style-GAN and grail.txt from webtext of nltk.corpus. The model is trained to find 5 topics, ie classify the words from the initial research paper in five topics. Next, this trained model is tested on grail.txt and most of the words (over 50%) fall in one category. The other test document is similar to the training document, ie a research paper from the same domain. Here the words are evenly distributed amongst all the five topics.

Note : Upload the text files attached

Upload the training document

```
import re
sentences = []
f = open("gan1.txt", "r")
for x in f:
  temp = re.sub('[^A-Za-z0-9]+', '', x)
  if len(temp) > 0 :
    sentences.append(x)

print(len(sentences))
print(sentences[:2])
```

```
217
['Generative Adversarial Nets\n', 'Ian J. Goodfellow?,   Jean Pouget-Abadie†,  Mehdi Mi
```

Load Gensim and nltk libraries

```
# pip install gensim
import gensim
from gensim.utils import simple_preprocess
from gensim.parsing.preprocessing import STOPWORDS
```

```
from nltk.stem import WordNetLemmatizer, SnowballStemmer
from nltk.stem.porter import *
import numpy as np
np.random.seed(400)

import pandas as pd
stemmer = SnowballStemmer("english")
```

```
import nltk
nltk.download('wordnet')
```

```
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data]   Unzipping corpora/wordnet.zip.
True
```

Tokenize and lemmatize

```
def lemmatize_stemming(text):
    return stemmer.stem(WordNetLemmatizer().lemmatize(text, pos='v'))

def preprocess(text):
    result=[]
    for token in gensim.utils.simple_preprocess(text) :
        if token not in gensim.parsing.preprocessing.STOPWORDS and len(token) > 3:
            sent = lemmatize_stemming(token)
            if len(sent) > 0 :
              result.append(sent)

    return result
```

```
docs = []

for doc in sentences:
    docs.append(preprocess(doc))
```

Preview Processed docs

```
print(docs)
```

```
[['generat', 'adversari', 'net'], ['goodfellow', 'jean', 'pouget', 'abadi', 'mehdi', 'm
```

BOW for preprocessed document

```
dictionary = gensim.corpora.Dictionary(docs)
```

```
dictionary = gensim.corpora.Dictionary(docs)
```

```
common_corpus = [dictionary.doc2bow(text) for text in docs]
```

```
document_num = 20
bow_doc_x = common_corpus[document_num]

for i in range(len(bow_doc_x)):
    print("Word {} (\"{}\") appears {} time.".format(bow_doc_x[i][0],
                                                     dictionary[bow_doc_x[i][0]],
                                                     bow_doc_x[i][1]))
```

```
Word 0 ("adversari") appears 1 time.
Word 1 ("generat") appears 3 time.
Word 27 ("approxim") appears 3 time.
Word 29 ("backpropag") appears 1 time.
Word 35 ("data") appears 1 time.
Word 36 ("defin") appears 1 time.
Word 38 ("discrimin") appears 3 time.
Word 39 ("distribut") appears 4 time.
Word 42 ("estim") appears 1 time.
Word 43 ("evalu") appears 1 time.
Word 48 ("game") appears 1 time.
Word 55 ("model") appears 8 time.
Word 61 ("probabl") appears 1 time.
Word 72 ("train") appears 4 time.
Word 75 ("work") appears 1 time.
Word 90 ("deep") appears 2 time.
Word 103 ("intract") appears 1 time.
Word 104 ("involv") appears 2 time.
Word 108 ("learn") appears 1 time.
Word 134 ("competit") appears 1 time.
Word 143 ("method") appears 1 time.
Word 175 ("nois") appears 4 time.
Word 201 ("network") appears 1 time.
Word 207 ("requir") appears 1 time.
Word 213 ("allow") appears 1 time.
Word 214 ("approach") appears 2 time.
Word 248 ("variat") appears 1 time.
Word 251 ("abil") appears 1 time.
Word 252 ("bind") appears 1 time.
Word 253 ("contrast") appears 1 time.
Word 254 ("criteria") appears 1 time.
Word 255 ("criterion") appears 1 time.
Word 256 ("densiti") appears 2 time.
Word 257 ("difficult") appears 1 time.
Word 258 ("fix") appears 1 time.
Word 259 ("formal") appears 1 time.
Word 260 ("increas") appears 1 time.
Word 261 ("inform") appears 1 time.
Word 262 ("limit") appears 1 time.
Word 263 ("lower") appears 1 time.
Word 264 ("mechan") appears 1 time.
Word 265 ("petit") appears 1 time.
Word 266 ("previous") appears 2 time.
```

```
Word 267 ("probabil") appears 1 time.
Word 268 ("probabili") appears 1 time.
Word 269 ("qualiti") appears 1 time.
Word 270 ("ratio") appears 2 time.
Word 271 ("see") appears 1 time.
Word 272 ("sequenc") appears 1 time.
Word 273 ("similar") appears 1 time.
Word 274 ("spirit") appears 1 time.
Word 275 ("take") appears 1 time.
Word 276 ("tie") appears 1 time.
Word 277 ("use") appears 1 time.
Word 278 ("weight") appears 1 time.
```

Train LDA model

```
# TODO
lda_model =  gensim.models.LdaMulticore(common_corpus,
                                        num_topics = 5,
                                      id2word = dictionary)
```

For each topic, explore the words occuring in that topic and its relative weight

```
for idx, topic in lda_model.print_topics(-1):
    print("Topic: {} \nWords: {}".format(idx, topic ))
    print("\n")
```

```
Topic: 0
Words: 0.019*"sampl" + 0.018*"generat" + 0.015*"adversari" + 0.014*"discrimin" + 0.014*


Topic: 1
Words: 0.024*"train" + 0.023*"pdata" + 0.020*"learn" + 0.020*"model" + 0.014*"generat"


Topic: 2
Words: 0.019*"generat" + 0.014*"network" + 0.012*"deep" + 0.012*"function" + 0.011*"con


Topic: 3
Words: 0.036*"model" + 0.035*"generat" + 0.027*"train" + 0.015*"distribut" + 0.013*"adv


Topic: 4
Words: 0.036*"model" + 0.028*"generat" + 0.018*"sampl" + 0.013*"deep" + 0.011*"train" +
```

Download the document grail.txt from nltk.corpus

```
import nltk
nltk.download()
nltk.download('inaugural')          #downloading required dependencies
nltk.download('nps_chat')
nltk.download('webtext')
nltk.download('treebank')
nltk.download('gutenberg')
nltk.download('genesis')
```

```
NLTK Downloader
---------------------------------------------------------------------------
    d) Download   l) List    u) Update   c) Config   h) Help   q) Quit
---------------------------------------------------------------------------
Downloader> q
[nltk_data] Downloading package inaugural to /root/nltk_data...
[nltk_data]   Unzipping corpora/inaugural.zip.
[nltk_data] Downloading package nps_chat to /root/nltk_data...
[nltk_data]   Unzipping corpora/nps_chat.zip.
[nltk_data] Downloading package webtext to /root/nltk_data...
[nltk_data]   Unzipping corpora/webtext.zip.
[nltk_data] Downloading package treebank to /root/nltk_data...
[nltk_data]   Unzipping corpora/treebank.zip.
[nltk_data] Downloading package gutenberg to /root/nltk_data...
[nltk_data]   Unzipping corpora/gutenberg.zip.
[nltk_data] Downloading package genesis to /root/nltk_data...
[nltk_data]   Unzipping corpora/genesis.zip.
True
```

```
from nltk.book import *
```

```
*** Introductory Examples for the NLTK Book ***
Loading text1, ..., text9 and sent1, ..., sent9
Type the name of the text or sentence to view it.
Type: 'texts()' or 'sents()' to list the materials.
text1: Moby Dick by Herman Melville 1851
text2: Sense and Sensibility by Jane Austen 1811
text3: The Book of Genesis
text4: Inaugural Address Corpus
text5: Chat Corpus
text6: Monty Python and the Holy Grail
text7: Wall Street Journal
text8: Personals Corpus
text9: The Man Who Was Thursday by G . K . Chesterton 1908
```

```
from nltk.corpus import webtext
test2 = webtext.raw('grail.txt')
```

```
test2.split('\n')
```

```
GALAHAD: Yup. ',
'KNIGHTS: That's it...",
TIM: Yes!',
ROBIN: Oh.',
```

```
ROBIN: Oh.',
ARTHUR: Oh.  Thank you.',
ROBIN: Ahh.',
GALAHAD: Oh.  Fine.',
ARTHUR: Thank you.',
ROBIN: Splendid.',
KNIGHTS: Aah... [boom pweeng boom boom] ',
'ARTHUR: Look, um, you're a busy man, uh--",
TIM: Yes, I can help you find the Holy Grail.',
KNIGHTS: Oh, thank you.  Oh...',
TIM: To the north there lies a cave-- the cave of Caerbannog-- wherein, carved ir
ARTHUR: Where could we find this cave, O Tim?',
TIM: Follow.  But!  Follow only if ye be men of valor, for the entrance to this c
ARTHUR: What an eccentric performance',
SCENE 21: [clop clop clop] [whinny whinny] ',
'GALAHAD: They're nervous, sire.",
'ARTHUR: Then we'd best leave them here and carry on on foot.  Dis-mount!",
TIM: Behold the cave of Caerbannog!',
ARTHUR: Right!  Keep me covered.',
GALAHAD: What with?',
ARTHUR: W-- just keep me covered.',
TIM: Too late! [dramatic chord] ',
ARTHUR: What?',

TIM: There he is!',
ARTHUR: Where?',
TIM: There!',
ARTHUR: What, behind the rabbit?',
TIM: It is the rabbit!',
ARTHUR: You silly sod!',
TIM: What?',
ARTHUR: You got us all worked up!',
'TIM: Well, that's no ordinary rabbit.",
ARTHUR: Ohh.',
'TIM: That's the most foul, cruel, and bad-tempered rodent you ever set eyes on.",
ROBIN: You tit!  I soiled my armor I was so scared!',
'TIM: Look, that rabbit's got a vicious streak a mile wide; it's a killer!",
GALAHAD: Get stuffed!',
'TIM: He'll do you up a treat mate!",
GALAHAD: Oh, yeah?',
ROBIN: You mangy scots git!',
'TIM: I'm warning you!",
'ROBIN: What's he do, nibble your bum?",
'TIM: He's got huge, sharp-- eh-- he can leap about-- look at the bones!",
ARTHUR: Go on, Bors.  Chop his head off!',
'BORS: Right!  Silly little bleeder.  One rabbit stew comin' right up!",
TIM: Look! [squeak] ',
BORS: Aaaugh! [dramatic chord] [clunk] ',
ARTHUR: Jesus Christ!',
TIM: I warned you!',
ROBIN: I done it again!',
'TIM: I warned you, but did you listen to me?  Oh, no, you knew it all, didn't you
ARTHUR: Oh, shut up!',
..]
```

Test the performance of grail.txt on our trained model

```
other_corpus = dictionary.doc2bow(preprocess(test2))

vector = lda_model[other_corpus]
```

```
print(vector)
```

```
[(0, 0.24040693), (1, 0.17126565), (2, 0.046760492), (3, 0.51869595), (4, 0.022870982)]
```

```
print(test2)
```

```
type(test2)
```

```
str
```

Note that most of the words are classified in the 4th topic and very less words are classified in 5th topic.

```
# Data preprocessing step for the unseen document
bow_vector = dictionary.doc2bow(preprocess(test2))

for index, score in sorted(lda_model[bow_vector], key=lambda tup: -1*tup[1]):
    print("Score: {}\t Topic: {}".format(score, lda_model.print_topic(index, 5)))
```

```
Score: 0.5186960101127625        Topic: 0.036*"model" + 0.035*"generat" + 0.027*"train"
Score: 0.24040697515010834       Topic: 0.019*"sampl" + 0.018*"generat" + 0.015*"advers
Score: 0.17126554250717163       Topic: 0.024*"train" + 0.023*"pdata" + 0.020*"learn" +
Score: 0.04676049202680588       Topic: 0.019*"generat" + 0.014*"network" + 0.012*"deep
Score: 0.022870982065796852      Topic: 0.036*"model" + 0.028*"generat" + 0.018*"sampl"
```

Open the next test document, style-gan

```
f2 = open("style-gan.txt", "r")
test1 = f2.read()
print(type(test1))
```

```
<class 'str'>
```

Test the performance on our trained model

```
# Data preprocessing step for the unseen document
bow_vector2 = dictionary.doc2bow(preprocess(test1))

for index, score in sorted(lda_model[bow_vector2], key=lambda tup: -1*tup[1]):
    print("Score: {}\t Topic: {}".format(score, lda_model.print_topic(index, 5)))
```

```
 Score: 0.3853776454925537       Topic: 0.036*"model" + 0.035*"generat" + 0.027*"train"
 Score: 0.2775808870792389       Topic: 0.024*"train" + 0.023*"pdata" + 0.020*"learn" +
 Score: 0.180592343211174        Topic: 0.036*"model" + 0.028*"generat" + 0.018*"sampl"
 Score: 0.12912648916244507      Topic: 0.019*"sampl" + 0.018*"generat" + 0.015*"advers
 Score: 0.02732263319194317      Topic: 0.019*"generat" + 0.014*"network" + 0.012*"deep
```

Here we notice that the words are evenly distributed in all topics since the training document was similar to the test document