In [1]:
```
# Initialize Otter
import otter
grader = otter.Notebook("hw7.ipynb")
```

CPSC 330 - Applied Machine Learning

Homework 7: Word embeddings and topic modeling

Due date: See the Calendar.

Imports

In [2]:
```
import os

%matplotlib inline
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
from sklearn.feature_extraction.text import CountVectorizer

from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.pipeline import Pipeline, make_pipeline
```

Submission instructions

rubric={points}
Please be aware that this homework assignment requires installation of several packages in your course environment. It's possible that you'll encounter installation challenges, which might be frustrating. However, remember that solving these issues is not wasting time but it is an essential skill for anyone aspiring to work in data science or machine learning.

Follow the homework submission instructions.

You may work in a group on this homework and submit your assignment as a group. Below are some instructions on working as a group.

- The maximum group size is 2.
- Use group work as an opportunity to collaborate and learn new things from each other.

- Be respectful to each other and make sure you understand all the concepts in the assignment well.
- It's your responsibility to make sure that the assignment is submitted by one of the group members before the deadline.
- You can find the instructions on how to do group submission on Gradescope here.

When you are ready to submit your assignment do the following:

1. Run all cells in your notebook to make sure there are no errors by doing `Kernel -> Restart Kernel and Clear All Outputs` and then `Run -> Run All Cells`.
2. Notebooks with cell execution numbers out of order or not starting from "1" will have marks deducted. Notebooks without the output displayed may not be graded at all (because we need to see the output in order to grade your work).
3. Upload the assignment using Gradescope's drag and drop tool. Check out this Gradescope Student Guide if you need help with Gradescope submission.
4. Make sure that the plots and output are rendered properly in your submitted file.
5. If the .ipynb file is too big and doesn't render on Gradescope, also upload a pdf or html in addition to the .ipynb. If the pdf or html also fail to render on Gradescope, please create two files for your homework: hw6a.ipynb with Exercise 1 and hw6b.ipynb with Exercises 2 and 3 and submit these two files in your submission.

Points: 2

Exercise 1: Exploring pre-trained word embeddings

In lecture 18, we talked about natural language processing (NLP). Using pre-trained word embeddings is very common in NLP. It has been shown that pre-trained word embeddings work well on a variety of text classification tasks. These embeddings are created by training a model like Word2Vec on a huge corpus of text such as a dump of Wikipedia or a dump of the web crawl.

A number of pre-trained word embeddings are available out there. Some popular ones are:

- GloVe
  - trained using the GloVe algorithm
  - published by Stanford University
- fastText pre-trained embeddings for 294 languages
  - trained using the fastText algorithm
  - published by Facebook

In this exercise, you will be exploring GloVe Wikipedia pre-trained embeddings. The code below loads the word vectors trained on Wikipedia using an algorithm called Glove. You'll need `gensim` package in your cpsc330 conda environment to run the code below.

```
> conda activate cpsc330
> conda install -c anaconda gensim
```

In [3]:
```
import gensim
import gensim.downloader

print(list(gensim.downloader.info()["models"].keys()))
```

```
['fasttext-wiki-news-subwords-300', 'conceptnet-numberbatch-17-06-300', 'word2vec
-ruscorpora-300', 'word2vec-google-news-300', 'glove-wiki-gigaword-50', 'glove-wi
ki-gigaword-100', 'glove-wiki-gigaword-200', 'glove-wiki-gigaword-300', 'glove-tw
itter-25', 'glove-twitter-50', 'glove-twitter-100', 'glove-twitter-200', '__testi
ng_word2vec-matrix-synopsis']
```

In [4]:
```
# This will take a while to run when you run it for the first time.
import gensim.downloader as api

glove_wiki_vectors = api.load("glove-wiki-gigaword-100")
```

In [5]:
```
len(glove_wiki_vectors)
```

Out[5]:
```
400000
```

There are 400,000 word vectors in this pre-trained model.

Now that we have GloVe Wiki vectors loaded in `glove_wiki_vectors`, let's explore the embeddings.

1.1 Word similarity using pre-trained embeddings

rubric={points}

Your tasks:

- Come up with a list of 4 words of your choice and find similar words to these words using `glove_wiki_vectors` embeddings.

Solution_1.1

Points: 2

In [6]:
```
...
four_words = ['up', 'data', 'medicine', "economics"]
```

In [7]:
```
...
# similar words to the first word "up":
glove_wiki_vectors.most_similar(four_words[0])
```

Out[7]:
```
[('down', 0.9159865975379944),
 ('out', 0.8888224363327026),
 ('back', 0.8425763845443726),
 ('put', 0.8326882123947144),
 ('just', 0.8320373892784119),
 ('off', 0.8262820839881897),
 ('over', 0.810619056224823),
 ('away', 0.8059401512145996),
 ('them', 0.804772675037384),
 ('into', 0.803209662437439)]
```

In [8]:
```
...
# similar words to the second word "data":
glove_wiki_vectors.most_similar(four_words[1])
```

Out[8]:
```
[('information', 0.7920401096343994),
 ('analysis', 0.756445586681366),
 ('tracking', 0.7226751446723938),
 ('database', 0.7215273976325989),
 ('system', 0.6736716628074646),
 ('computer', 0.6733997464179993),
 ('statistics', 0.6705882549285889),
 ('systems', 0.669585645198822),
 ('applications', 0.6660566926002502),
 ('numbers', 0.6655443906784058)]
```

In [9]:
```
...
# similar words to the first word "medicine":
glove_wiki_vectors.most_similar(four_words[2])
```

Out[9]:
```
[('medical', 0.821526288986206),
 ('medicines', 0.7068259716033936),
 ('health', 0.6808973550796509),
 ('veterinary', 0.6703415513038635),
 ('nutrition', 0.6656507253646851),
 ('studies', 0.6543422341346741),
 ('physicians', 0.652547299861908),
 ('dentistry', 0.6510076522827148),
 ('science', 0.6494085192680359),
 ('psychiatry', 0.6493012309074402)]
```

In [10]:
```
...
# similar words to the first word "economics":
glove_wiki_vectors.most_similar(four_words[3])
```

Out[10]:
```
[('sociology', 0.7726682424545288),
 ('professor', 0.7641003727912903),
 ('science', 0.7270671725273132),
 ('philosophy', 0.7264132499694824),
 ('psychology', 0.7084123492240906),
 ('mathematics', 0.6896660327911377),
 ('sciences', 0.677860677242279),
 ('physics', 0.6771532297134399),
 ('harvard', 0.6714091300964355),
 ('institute', 0.6638572216033936)]
```

1.2 Word similarity using pre-trained embeddings

rubric={points}

Your tasks:

1. Calculate cosine similarity for the following word pairs (`word_pairs`) using the `similarity` method of `glove_wiki_vectors` .

In [11]:
```
word_pairs = [
    ("coast", "shore"),
    ("clothes", "closet"),
    ("old", "new"),
    ("smart", "intelligent"),
    ("dog", "cat"),
    ("tree", "lawyer"),
]
```

Solution_1.2

Points: 2

In [12]:
```
...
for (word1, word2) in word_pairs:
    print(glove_wiki_vectors.similarity(word1, word2))
```

```
0.7000272
0.546276
0.6432488
0.7552732
0.8798075
0.076719455
```

1.3 Representation of all words in English

rubric={points}

Your tasks:

1. The vocabulary size of Wikipedia embeddings is quite large. The `test_words` list below contains a few new words (called neologisms) and biomedical domain-specific abbreviations. Write code to check whether `glove_wiki_vectors` have representation for these words or not.

   > If a given word `word` is in the vocabulary, `word in glove_wiki_vectors` will return True.

In [13]:
```
test_words = [
    "covididiot",
    "fomo",
    "frenemies",
    "anthropause",
    "photobomb",
    "selfie",
    "pxg",  # Abbreviation for pseudoexfoliative glaucoma
    "pacg",  # Abbreviation for primary angle closure glaucoma
    "cct",  # Abbreviation for central corneal thickness
    "escc",  # Abbreviation for esophageal squamous cell carcinoma
]
```

Solution_1_3

Points: 2

In [14]:
```
...
for word in test_words:
    if word in glove_wiki_vectors:
        print(f"glove_wiki_vectors have representation for the word '{word}'")
    else:
        print(f"glove_wiki_vectors do not have representation for the word '{wor
```

```
glove_wiki_vectors do not have representation for the word 'covididiot'
glove_wiki_vectors do not have representation for the word 'fomo'
glove_wiki_vectors have representation for the word 'frenemies'
glove_wiki_vectors do not have representation for the word 'anthropause'
glove_wiki_vectors do not have representation for the word 'photobomb'
glove_wiki_vectors do not have representation for the word 'selfie'
glove_wiki_vectors do not have representation for the word 'pxg'
glove_wiki_vectors do not have representation for the word 'pacg'
glove_wiki_vectors have representation for the word 'cct'
glove_wiki_vectors have representation for the word 'escc'
```

1.4 Stereotypes and biases in embeddings

rubric={points}

Word vectors contain lots of useful information. But they also contain stereotypes and biases of the texts they were trained on. In the lecture, we saw an example of

gender bias in Google News word embeddings. Here we are using pre-trained embeddings trained on Wikipedia data.

Your tasks:

1. Explore whether there are any worrisome biases or stereotypes present in these embeddings by trying out at least 4 examples. You can use the following two methods or other methods of your choice to explore this.
   - the `analogy` function below which gives word analogies (an example shown below)
   - similarity or distance methods (an example is shown below)

> Note that most of the recent embeddings are de-biased. But you might still observe some biases in them. Also, not all stereotypes present in pre-trained embeddings are necessarily bad. But you should be aware of them when you use them in your models.

In [15]:
```
def analogy(word1, word2, word3, model=glove_wiki_vectors):
    """
    Returns analogy word using the given model.

    Parameters
    --------------
    word1 : (str)
        word1 in the analogy relation
    word2 : (str)
        word2 in the analogy relation
    word3 : (str)
        word3 in the analogy relation
    model :
        word embedding model

    Returns
    ---------------
        pd.dataframe
    """
    print("%s : %s :: %s : ?" % (word1, word2, word3))
    sim_words = model.most_similar(positive=[word3, word2], negative=[word1])
    return pd.DataFrame(sim_words, columns=["Analogy word", "Score"])
```

Examples of using analogy to explore biases and stereotypes.

In [16]:
```
analogy("man", "doctor", "woman")
```

```
man : doctor :: woman : ?
```

Out[16]:

	Analogy word	Score
0	nurse	0.773523
1	physician	0.718943
2	doctors	0.682433
3	patient	0.675068
4	dentist	0.672603
5	pregnant	0.664246
6	medical	0.652045
7	nursing	0.645348
8	mother	0.639333
9	hospital	0.638750

In [17]:
```
glove_wiki_vectors.similarity("aboriginal", "success")
```

Out[17]:
```
0.14283238
```

In [18]:
```
glove_wiki_vectors.similarity("white", "success")
```

Out[18]:
```
0.351824
```

Solution_1_4

Points: 4

In [19]:
```
...
analogy("man", "manager", "woman")
```

```
man : manager :: woman : ?
```

Out[19]:

	Analogy word	Score
0	assistant	0.656641
1	job	0.626392
2	supervisor	0.593479
3	owner	0.587832
4	hired	0.577390
5	consultant	0.570659
6	director	0.565653
7	office	0.564568
8	employee	0.561287
9	ceo	0.556422

In [20]:
```
...
print(glove_wiki_vectors.similarity("asian", "creative"))
print(glove_wiki_vectors.similarity("american", "creative"))
```

```
0.2578922
0.40354294
```

In [21]:
```
...
print(glove_wiki_vectors.similarity("christian", "extremism"))
print(glove_wiki_vectors.similarity("muslim", "extremism"))
```

```
0.275774
0.49818176
```

In [22]:
```
...
print(glove_wiki_vectors.similarity("young", "brave"))
print(glove_wiki_vectors.similarity("old", "brave"))
```

```
0.48625344
0.274084
```

1.5 Discussion

rubric={points}

Your tasks:

1. Discuss your observations from 1.4. Are there any worrisome biases in these embeddings trained on Wikipedia?
2. Give an example of how using embeddings with biases could cause harm in the real world.

Solution_1_5

Points: 4

1. There are worrisome biases in these embeddings: the embeddings analogize the word pair "man" and "manager" to the word pair of "woman" and "assistant", which shows the bias on gender; in my second example, the similarity between "asian" and "creative" is lower than the one between "american" and "creative" by roughly 0.15, which shows a bias on different cultures; for the thrid example, the word pair of "muslim" and "extremism" are more similar than "christian" and "extremism" are by almost 0.23, showing the bias on religions; in the last example, the word "brave" is more similar to "young" than to "old" by 0.212, showing the age stereotypes in these embeddings trained on Wikipedia.
2. When we build a recommendation system using word embeddings with biases, for example, a job recommendation system, it might recommend male

for leadership roles based on a stronger assocation of "man" with "manager" and of "woman" with "assistant". This would encourage workplace gender inequality, avoid more suitable candidates getting the positions, and also hinder individuals' career development due to their gender.

Exercise 2: Topic modeling

The goal of topic modeling is discovering high-level themes in a large collection of texts.

In this homework, you will explore topics in the 20 newsgroups text dataset using `scikit-learn`'s `LatentDirichletAllocation` (LDA) model.

Usually, topic modeling is used for discovering abstract "topics" that occur in a collection of documents when you do not know the actual topics present in the documents. But 20 newsgroups text dataset is labeled with categories (e.g., sports, hardware, religion), and you will be able to cross-check the topics discovered by your model with these available topics.

The starter code below loads the train and test portion of the data and convert the train portion into a pandas DataFrame. For speed, we will only consider documents with the following 8 categories.

In [23]:
```
from sklearn.datasets import fetch_20newsgroups
```

In [24]:
```
cats = [
    "rec.sport.hockey",
    "rec.sport.baseball",
    "soc.religion.christian",
    "alt.atheism",
    "comp.graphics",
    "comp.windows.x",
    "talk.politics.mideast",
    "talk.politics.guns",
]  # We'll only consider these categories out of 20 categories for speed.

newsgroups_train = fetch_20newsgroups(
    subset="train", remove=("headers", "footers", "quotes"), categories=cats
)
X_news_train, y_news_train = newsgroups_train.data, newsgroups_train.target
df = pd.DataFrame(X_news_train, columns=["text"])
df["target"] = y_news_train
df["target_name"] = [
    newsgroups_train.target_names[target] for target in newsgroups_train.target
]
df
```

Out[24]:

	text	target	target_name
0	You know, I was reading 18 U.S.C. 922 and some...	6	talk.politics.guns
1	\n\n\nIt's not a bad question: I don't have an...	1	comp.graphics
2	\nActuallay I don't, but on the other hand I d...	1	comp.graphics
3	The following problem is really bugging me,\na...	2	comp.windows.x
4	\n\n This is the latest from UPI \n\n For...	7	talk.politics.mideast
...	...	...	...
4558	Hi Everyone ::\n\nI am looking for some soft...	1	comp.graphics
4559	Archive-name: x-faq/part3\nLast-modified: 1993...	2	comp.windows.x
4560	\nThat's nice, but it doesn't answer the quest...	6	talk.politics.guns
4561	Hi,\n I just got myself a Gateway 4DX-33V ...	2	comp.windows.x
4562	\n\n[h] \tThe Armenians in Nagarno-Karabagh ar...	7	talk.politics.mideast

4563 rows × 3 columns

In [25]:
```
newsgroups_train.target_names
```

Out[25]:
```
['alt.atheism',
 'comp.graphics',
 'comp.windows.x',
 'rec.sport.baseball',
 'rec.sport.hockey',
 'soc.religion.christian',
 'talk.politics.guns',
 'talk.politics.mideast']
```

2.1 Preprocessing using spaCy

rubric={points}

Preprocessing is a crucial step before carrying out topic modeling and it markedly affects topic modeling results. In this exercise, you'll prepare the data using spaCy for topic modeling.

Your tasks:

- Write code using spaCy to preprocess the `text` column in the given dataframe `df` and save the processed text in a new column called `text_pp` within the same dataframe.

If you do not have spaCy in your course environment, you'll have to install it and download the pretrained model en_core_web_md.

```
python -m spacy download en_core_web_md
```

Note that there is no such thing as "perfect" preprocessing. You'll have to make your own judgments and decisions on which tokens are likely to be more informative for the given task. Some common text preprocessing steps for topic modeling include:

- getting rid of slashes, new-line characters, or any other non-informative characters
- sentence segmentation and tokenization
- replacing urls, email addresses, or numbers with generic tokens such as "URL", "EMAIL", "NUM".
- getting rid of other fairly unique tokens which are not going to help us in topic modeling
- excluding stopwords and punctuation
- lemmatization

> Check out these available attributes for `token` in spaCy which might help you with preprocessing.

> You can also get rid of words with specific POS tags. Here is the list of part-of-speech tags used in spaCy.

> You may have to use regex to clean text before passing it to spaCy. Also, you might have to go back and forth between preprocessing in this exercise and and topic modeling in Exercise 2 before finalizing preprocessing steps.

> Note that preprocessing the corpus might take some time. So here are a couple of suggestions: 1) During the debugging phase, work on a smaller subset of the data. 2) Once you finalize the preprocessing part, you might want to save the preprocessed data in a CSV and work with this CSV so that you don't run the preprocessing part every time you run the notebook.

In [26]:
```
import spacy
nlp = spacy.load("en_core_web_md", disable=["parser", "ner"])
```

Solution_2_1

Points: 8

In [27]:
```
...
# take a small subset and see the current text:
subset_df = df.head(100)  # select the first 100 rows of data

pd.set_option('display.max_colwidth', None)
print(subset_df.iloc[6:8]["text"])
```

6 \n\nVery simple.\n\n"X-Soviet Armenian government must pay for their crime o
f genocide \n against 2.5 million Muslims by admitting to the crime and making \n
reparations to the Turks and Kurds."\n\nAfter all, your criminal grandparents ext
erminated 2.5 million Muslim\npeople between 1914 and 1920.\n\n\n<C5yyBt.5zo@new
s.cso.uiuc.edu>\nhovig@uxa.cso.uiuc.edu (Hovig Heghinian)\n\n\nYou must be a new
'Arromdian'. You are counting on ASALA/SDPA/ARF \ncrooks and criminals to prove s
omething for you? No wonder you are in \nsuch a mess. That criminal idiot and 'it
s' forged/non-existent junk has \nalready been trashed out by Mutlu, Cosar, Akgu
n, Uludamar, Akman, Oflazer \nand hundreds of people. Moreover, ASALA/SDPA/ARF cr
iminals are responsible \nfor the massacre of the Turkish people that also preven
t them from entering \nTurkiye and TRNC. SDPA has yet to renounce its charter whi
ch specifically \ncalls for the second genocide of the Turkish people. This racis
t, barbarian \nand criminal view has been touted by the fascist x-Soviet Armenian
government \nas merely a step on the road to said genocide. \n\nNow where shall I
begin?\n\n#From: ahmet@eecg.toronto.edu (Parlakbilek Ahmet)\n#Subject: YALANCI, L
IAR : DAVIDIAN\n#Keywords: Davidian, the biggest liar\n#Message-ID: <1991Jan10.12
2057.11613@jarvis.csri.toronto.edu>\n\nFollowing is the article that Davidian cla
ims that Hasan Mutlu is a liar:\n\n\n\n\n\n\n[some parts are deleted]\n\n\n\n\n\t
\t\t\t- - -\n\n\n\n\nReceiving this message, I checked the reference, L.Kuper,"Ge
nocide..." and\nwhat I have found was totally consistent with what Davidian said.
The book\nwas like "voice of Armenian revolutionists" and although I read the who
le book,\nI could not find the original quota.\nBut there was one more thing to c
heck:The original posting of Mutlu.I found \nthe original article of Mutlu.It is
as follows:\n\n\n\n ==========
==========\n ======\n\nQUATO IS THE SAME, REFERENCE IS DIFFERENT !\n\nDAVID
IAN LIED AGAIN, AND THIS TIME HE CHANGED THE ORIGINAL POSTING OF MUTLU\nJUST TO A
CCUSE HIM TO BE A LIAR.\n\nDavidian, thank you for writing the page number correc
tly...\n\nYou are the biggest liar I have ever seen.This example showed me that t
omorrow\nyou can lie again, and you may try to make me a liar this time.So I deci
ded\nnot to read your articles and not to write answers to you.I also advise\nall
the netters to do the same.We can not prevent your lies, but at least\nwe may sav
e time by not dealing with your lies.\n\nAnd for the following line:\n\nI also re
turn all the insults you wrote about Mutlu to you.\nI hope you will be drowned in
your lies.\n\nAhmet PARLAKBILEK\n\n#From: vd8@cunixb.cc.columbia.edu (Vedat Doga
n)\n#Message-ID: <1993Apr8.233029.29094@news.columbia.edu>\n\n \nn>crap posted by
Mr. [(*]!\n \n o boy! \n \n Please, can you tell us why those quotes are "cra
p"?..because you do not \n like them!!!...because they really exist...why?\n \n A
s I said in my previous posting, those quotes exactly exist in the source \n give
n by Serdar Argic .. \n \n You couldn't reject it...\n \n \n Here we go agai
n..\n In the book I have, both the front page and the Author's preface give \n th
e same year: 1923 and 15 January, 1923, respectively!\n (Anyone can check it at h
er/his library,if not, I can send you the copies of\n pages, please ask by sct)
\n \n \nI really don't care what year it was first published(1923 or 1924)\nWhat
I care about is what the book writes about murders, tortures,et..in\nthe given qu
otes by Serdar Argic, and your denial of these quotes..and your\ngroundless accus
sations, etc. \n \n[...]\n \n \n I claim I have a book in my hand published in 19
23(first publication)\n and it exactly has the same quoted info as the book publi
shed\n in 1934(Serdar Argic's Reference) has..You couldn't reject it..but, now yo
u\n are avoiding the real issues by twisting around..\n \n Let's see how you li
e!..(from 'non-existing' quotes to re-publication)\n \n First you said there was
no such a quote in the given reference..You\n called Serdar Argic a liar!..\n I s
aid to you, NO, MR.Davidian, there exactly existed such a quote...\n (I even gave
the call number, page numbers..you could't reject it.)\n \n And now, you are lyin
g again and talking about "modified,re-published book"\n(without any proof :how,
when, where, by whom, etc..)..\n (by the way, how is it possible to re-publish th
e book in 1923 if it was\n first published in 1924(your claim).I am sure that yo
u have some 'pretty \n well suited theories', as usual)\n \n And I am ready to s
end the copies of the necessary pages to anybody who\n wants to compare the fact
and Mr.Davidian's lies...I also give the call number\n and page numbers again for
the library use, which are: \n 949.6 R 198\n \n and the page

```
numbers to verify the quotes:218 and 215\n                \n     \n \n \n Now, are
you claiming that there can't be such a reference by saying "it is\n not possibl
e..." ..If not, what is your point?\n \n Differences in the number of pages?\n Mi
ne was published in 1923..Serdar Argic's was in 1934..\n No need to use the same
book size and the same letter \n charachter in both publications,etc, etc.. does
it give you an idea!!\n \n The issue was not the number of pages the book has..or
the year\n first published.. \n And you tried to hide the whole point..\n the poi
nt is that both books have the exactly the same quotes about\n how moslems are ki
lled, tortured,etc by Armenians..and those quotes given \n by Serdar Argic exis
t!! \n It was the issue, wasn't-it?  \n \n you were not able to object it...Does
it bother you anyway? \n \n You name all these tortures and murders (by Armenian
s) as a "crap"..\n People who think like you are among the main reasons why the W
orld still\n has so many "craps" in the 1993. \n \n Any question?\n \n\n<C5wwqA.9
wL@news.cso.uiuc.edu>\nhovig@uxa.cso.uiuc.edu (Hovig Heghinian)\n\n\nWell, appare
ntly we have another son of Dro 'the Butcher' to contend with. \nYou should indee
d be happy to know that you rekindled a huge discussion on\ndistortions propagate
d by several of your contemporaries. If you feel \nthat you can simply act as an
Armenian governmental crony in this forum \nyou will be sadly mistaken and duly e
mbarrassed. This is not a lecture to \nanother historical revisionist and a genoc
ide apologist, but a fact.\n\nI will dissect article-by-article, paragraph-by-par
agraph, line-by-line, \nlie-by-lie, revision-by-revision, written by those on thi
s net, who plan \nto 'prove' that the Armenian genocide of 2.5 million Turks and
Kurds is \nnothing less than a classic un-redressed genocide. We are neither in
\nx-Soviet Union, nor in some similar ultra-nationalist fascist dictatorship, \nt
hat employs the dictates of Hitler to quell domestic unrest. Also, feel \nfree to
distribute all responses to your nearest ASALA/SDPA/ARF terrorists,\nthe Armenian
pseudo-scholars, or to those affiliated with the Armenian\ncriminal organization
s.\n\nArmenian government got away with the genocide of 2.5 million Turkish me
n,\nwomen and children and is enjoying the fruits of that genocide. You, and \nth
ose like you, will not get away with the genocide's cover-up.\n\nNot a chance.\n
\nSerdar Argic
7
\nNo no no!!!  It's a squid!  Keep the tradition alive!  (Kinda like the\nfish at
UNH games....)
Name: text, dtype: object
```

In [28]:
```python
import re   # regex
```

In [ ]:
```python
...
# reference: Lecture 18

def preprocess(doc, irrelevant_pos=["ADV", "PRON", "CCONJ", "PUNCT", "PART", "DE
    clean_text = []

    text = doc.text
    # getting rid of slashes, new-line characters, or any other non-informative
    # replacing urls, email addresses, or numbers with generic tokens such as "U
    # reference: https://stackoverflow.com/questions/11331982/how-to-remove-any-
    text = re.sub(r'http\S+|www\S+|https\S+', 'URL', text, flags=re.MULTILINE)
    text = re.sub(r'\S+@\S+\.\S+', 'EMAIL', text)
    text = re.sub(r'\b\d+\b', 'NUM', text)
    text = re.sub(r'[\n/\\-]', ' ', text)
    text = text.strip()
    # sentence segmentation and tokenization
    doc = nlp(text)

    # Filter tokens based on various criteria:
    # getting rid of other fairly unique tokens which are not going to help us i
    # excluding stopwords and punctuation
    for token in doc:
```

```
        if (
            token.is_stop == False  # Check if it's not a stopword
            and token.is_punct == False  # punctuation
            and token.is_space == False
            and token.pos_ not in irrelevant_pos
            ):  # Check if the POS is in the acceptable POS tags
            # lemmatization:
            lemma = token.lemma_
            clean_text.append(lemma.lower())

    return " ".join(clean_text)
```

In [30]:
```
# try the preprocess function on the subset
subset_df["text_pp"] = [preprocess(text) for text in nlp.pipe(subset_df["text"])]
```

```
C:\Users\sunxi\AppData\Local\Temp\ipykernel_20612\243745135.py:2: SettingWithCopy
Warning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stabl
e/user_guide/indexing.html#returning-a-view-versus-a-copy
  subset_df["text_pp"] = [preprocess(text) for text in nlp.pipe(subset_df["tex
t"])]
```

In [31]:
```
# check on the first two rows:
print(subset_df.iloc[6:8]["text_pp"])
```

```
6    simple x soviet armenian government pay crime genocide num.num million musli
ms admit crime make reparation turks kurds criminal grandparent exterminate num.n
um million muslim people num num email email hovig heghinian new arromdian count
asala sdpa arf crook criminal prove wonder mess criminal idiot forge non existent
junk trash mutlu cosar akgun uludamar akman oflazer hundred people asala sdpa arf
criminal responsible massacre turkish people prevent enter turkiye trnc sdpa reno
unce charter call second genocide turkish people racist barbarian criminal view t
out fascist x soviet armenian government step road say genocide shall begin email
parlakbilek ahmet subject yalanci liar davidian keywords davidian big liar messag
e id email follow article davidian claim hasan mutlu liar part delete receive mes
sage check reference l.kuper,"genocide find consistent davidian say book voice ar
menian revolutionist read book find original quota thing check original posting m
utlu find original article mutlu follow = = = = = = = = = = = = = = = = = = = = =
= = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =
= = = = = = = = = = = = = quato reference different davidian lie time change orig
inal posting mutlu accuse liar davidian thank write page number big liar see exam
ple show tomorrow lie try liar time decide read article write answer advise nette
r prevent lie save time deal lie follow line return insult write mutlu hope drown
lie ahmet parlakbilek email vedat dogan message id email > crap post mr. o boy te
ll quote crap"? them!!! exist say previous posting quote exist source give serdar
argic reject book page author preface year num num january num check library send
copy page ask sct care year published(num num care book write murder torture et g
ive quote serdar argic denial quote groundless accussation etc claim book hand pu
blish num(first publication quote info book publish num(serdar argic reference re
ject avoid real issue twist let lie! non exist quote publication say quote give r
eference call serdar argic liar say mr.davidian exist quote give number page numb
er could't reject lie talk modify publish book proof etc way possible publish boo
k num publish num(your claim).i sure suited theory usual ready send copy necessar
y page want compare fact mr. davidian lie number page number library use num.num
r num page number verify quote num num claim reference say possible point differe
nce number page publish num serdar argic num need use book size letter charachter
publication etc etc idea issue number page book year publish try hide point point
book quote moslem kill torture etc armenians quote give serdar argic exist issue
able object bother torture murder armenians crap people think main reason world c
rap num question email email hovig heghinian son dro butcher contend happy know r
ekindle huge discussion distortion propagate contemporary feel act armenian gover
nmental crony forum mistaken embarrassed lecture historical revisionist genocide
apologist fact dissect article article paragraph paragraph line line lie lie revi
sion revision write net plan prove armenian genocide num.num million turks kurds
classic un redress genocide x soviet union similar ultra nationalist fascist dict
atorship employ dictate hitler quell domestic unrest feel free distribute respons
e near asala sdpa arf terrorist armenian pseudo scholar affiliate armenian crimin
al organization armenian government get genocide num.num million turkish man woma
n child enjoy fruit genocide genocide cover chance serdar argic
7
squid tradition alive fish unh game
Name: text_pp, dtype: object
```

In [32]:

```
...
# df["text_pp"] = [preprocess(text) for text in nlp.pipe(df["text"])]
# the code above crahsed my kernel so I devide the df into 3 parts and after run
# therefore, i commented out the followed preprocessing code and save the prepro
# finally i merged them into one final df, which will be used later in the model
n = len(df)
df1 = df.iloc[:n//3]
df2 = df.iloc[n//3:2*n//3]
df3 = df.iloc[2*n//3:]
df3_1 = df3.iloc[:n//3]  # actually i failed to further split the third part, bu
```

In [33]:
```
# df1["text_pp"] = [preprocess(text) for text in nlp.pipe(df1["text"])]
# df1.to_csv("df1_preprocessed.csv", index=False)
```

In [34]:
```
# df2["text_pp"] = [preprocess(text) for text in nlp.pipe(df2["text"])]
# df2.to_csv("df2_preprocessed.csv", index=False)
```

In [35]:
```
# df3_1["text_pp"] = [preprocess(text) for text in nlp.pipe(df3_1["text"])]
# df3_1.to_csv("df3_1_preprocessed.csv", index=False)
```

In [36]:
```
print(n)
print(len(df1)+len(df2)+len(df3_1))
```

```
4563
4563
```

In [37]:
```
df1 = pd.read_csv("df1_preprocessed.csv")
df2 = pd.read_csv("df2_preprocessed.csv")
df3 = pd.read_csv("df3_1_preprocessed.csv")

# combine the three files
final_df = pd.concat([df1, df2, df3], ignore_index=True)

# save the final preprocessing result
final_df.to_csv("final_preprocessed.csv", index=False)
```

In [38]:
```
df.iloc[2:6]
```

Out[38]:

	text	target	target_r
2	\nActuallay I don't, but on the other hand I don't support the idea of having\none newsgroup for every aspect of graphics programming as proposed by Brian,\nin his reply to my original posting.\nI would suggest a looser structure more like a comp.graphics.programmer,\ncomp.graphics.hw_and_sw\nThe reason for making as few groups as possible is for the same reason you\nsay we shouldn't spilt up, not to get to few postings every day.\nI takes to much time to browse through all postings just to find two or \nthree I'm interested in.\n\nI understand and agree when you say you want all aspects of graphics in one\nmeeting. I agree to some extension. I see news as a forum to exchange ideas,\nhelp others or to be helped. I think this is difficult to achive if there\nare so many different things in one meeting.\n\nGood evening netters|-)	1	comp.gra
3	The following problem is really bugging me,\nand I would appreciate any help.\n\nI create two windows:\n\nw1 (child to root) with event_mask = ButtonPressMask|KeyPressMask;\nw2 (child to w1) with do_not_propagate_mask = ButtonPressMask|KeyPressMask;\n\n\nKeypress events in w2 are discarded, but ButtonPress events fall through\nto w1, with subwindow set to w2.\n\nFYI, I'm using xnews/olvwm.\n\nAm I doing something fundamentally wrong here?	2	comp.windo
4	\n\n This is the latest from UPI \n\n Foreign Ministry spokesman Ferhat Ataman told journalists Turkey was\n closing its air space to all flights to and from Armenia and would\n prevent humanitarian aid from reaching the republic overland across\n Turkish territory.\n\n \n Historically even the most uncivilized of peoples have exhibited \n signs of compassion by allowing humanitarian aid to reach civilian\n populations. Even the Nazis did this much.\n\n It seems as though from now on Turkey will publicly pronounce \n themselves 'hypocrites' should they choose to continue their\n condemnation of the Serbians.\n\n\n\n--	7	talk.politics.mi
5	Hi,\n I'd like to subscribe to Leadership Magazine but wonder if there is one on\ndisk instead of on paper. Having it on disk would save me retyping\nillustrations, etc into a word processor. It's just cut and paste.\n If there are other good Christian magazines like Leadership on disk media,\nI'd appreciate any info.	5	soc.religion.chr

In [39]:
```
final_df.iloc[2:6]
```

Out[39]:

	text	target	target_r
2	\nActuallay I don't, but on the other hand I don't support the idea of having\none newsgroup for every aspect of graphics programming as proposed by Brian,\nin his reply to my original posting.\nI would suggest a looser structure more like a comp.graphics.programmer,\ncomp.graphics.hw_and_sw\nThe reason for making as few groups as possible is for the same reason you\nsay we shouldn't spilt up, not to get to few postings every day.\nI takes to much time to browse through all postings just to find two or \nthree I'm interested in.\n\nI understand and agree when you say you want all aspects of graphics in one\nmeeting. I agree to some extension. I see news as a forum to exchange ideas,\nhelp others or to be helped. I think this is difficult to achive if there\nare so many different things in one meeting.\n\nGood evening netters|-)	1	comp.gra
3	The following problem is really bugging me,\nand I would appreciate any help.\n\nI create two windows:\n\nw1 (child to root) with event_mask = ButtonPressMask|KeyPressMask;\nw2 (child to w1) with do_not_propagate_mask = ButtonPressMask|KeyPressMask;\n\n\nKeypress events in w2 are discarded, but ButtonPress events fall through\nto w1, with subwindow set to w2.\n\nFYI, I'm using xnews/olvwm.\n\nAm I doing something fundamentally wrong here?	2	comp.wind
4	\n\n This is the latest from UPI \n\n Foreign Ministry spokesman Ferhat Ataman told journalists Turkey was\n closing its air space to all flights to and from Armenia and would\n prevent humanitarian aid from reaching the republic overland across\n Turkish territory.\n\n \n Historically even the most uncivilized of peoples have exhibited \n signs of compassion by allowing humanitarian aid to reach civilian\n populations. Even the Nazis did this much.\n\n It seems as though from now on Turkey will publicly pronounce \n themselves 'hypocrites' should they choose to continue their\n condemnation of the Serbians.\n\n\n\n--	7	talk.politics.mi
5	Hi,\n I'd like to subscribe to Leadership Magazine but wonder if there is one on\ndisk instead of on paper. Having it on disk would save me retyping\nillustrations, etc into a word processor. It's just cut and paste.\n If there are other good Christian magazines like Leadership on disk media,\nI'd appreciate any info.	5	soc.religion.chr

2.2 Justification

rubric={points}

Your tasks:

- Outline the preprocessing steps you carried out in the previous exercise (bullet point format is fine), providing a brief justification when necessary.

> You might want to wait to answer this question till you are done with Exercise 2 and you have finalized the preprocessing steps in 2.1.

Solution_2_2

Points: 2

- the input is not a text object, so first I use .text() to convert it to text.
- removing the non-informative characters including slashes, new-line characters, hyphens, and other white spaces using .re.sub(r'[\n/\-]', ' ', text) and .strip();
- using functions in regex to replace urls, email addresses, or numbers with generic tokens such as "URL", "EMAIL", "NUM".
- using doc = nlp(text), i get a structured text object, which allows us to extract information such as tokenization results, part-of-speech (POS) tagging more easily. Here we realized sentence segmentation and tokenization with it.
- filter tokens: removing stopwords, punctuation and space, also getting rid of the POS which are not going to be helpful in topic modeling
- lemmatization using .lemma_ to get the original word format.
- using .lower() to get the lowercase of the words.
- add the result words into the final clean text list, and convert them to a string in the end.

2.3 Build a topic model using sklearn's LatentDirichletAllocation

rubric={points}

Your tasks:

1. Build LDA models on the preprocessed data using using sklearn's `LatentDirichletAllocation` and random state 42. Experiment with a few values for the number of topics (`n_components`). Pick a reasonable number for the number of topics and briefly justify your choice.

Solution_2_3

Points: 4

below, I build a LDA models on the preprocessed data using LatentDirichletAllocation, and I try with three values for n_components. Comparing these three results, I think 4 is the most reasonable one for the the number of topics (programming, religion, politics, and sports for topic 0, 1, 2, 3, respectively). For other higher values, some topics are seemingly good to be merged with other group, such as topic 2 and 3 (both look like a topic of sports or game) for n=6, and topic 0 and 5 (programming), topic 2 and 3 for n=8.

In [40]:
```
# get rid of missing values
final_df["text_pp"] = final_df["text_pp"].fillna('')
```

In [41]:
```
from sklearn.feature_extraction.text import CountVectorizer

vec = CountVectorizer(stop_words='english')
X = vec.fit_transform(final_df["text_pp"])
```

In [42]:
```
import mglearn
from sklearn.decomposition import LatentDirichletAllocation

n_components = [4, 6, 8]   # Experiment with different numbers of topics
lda_models = {}
feature_names = np.array(vec.get_feature_names_out())

for n in n_components:
    lda = LatentDirichletAllocation(n_components=n, random_state=42)
    lda.fit_transform(X)
    sorting = np.argsort(lda.components_, axis=1)[:, ::-1]

    mglearn.tools.print_topics(
        topics=range(n),
        feature_names=feature_names,
        sorting=sorting,
        topics_per_chunk=5,
        n_words=10,
    )
```

topic 0	topic 1	topic 2	topic 3
num	god	num	num
file	people	people	email
email	num	say	game
program	think	year	play
use	know	think	team
window	believe	gun	season
image	jesus	know	new
entry	say	time	period
include	thing	come	la
server	time	good	goal

topic 0	topic 1	topic 2	topic 3	topic 4
num	god	say	num	num
entry	people	think	game	gun
file	know	year	play	people
armenian	think	people	team	law
output	believe	know	season	jews
program	jesus	come	email	right
armenians	thing	num	period	weapon
line	say	good	goal	state
armenia	question	time	la	know
rule	church	team	new	firearm

topic 5
num
email
file
image
use
window
program
available
run
server

topic 0	topic 1	topic 2	topic 3	topic 4
num	god	think	num	num
entry	believe	say	game	gun
file	people	year	play	people
output	jesus	know	team	law
program	know	good	season	say
line	think	come	period	weapon
build	church	game	goal	right
rule	question	team	la	know
section	thing	time	new	firearm
char	say	people	year	think

topic 5	topic 6	topic 7
num	num	num
email	armenian	israel

```
file          turkish       jews
image         armenians     israeli
window        turkey        people
use           email         jewish
program       people        state
available     turks         time
run           armenia       war
server        greek         right
```

In [43]:
```
# reference: Lecture 18

n_topics = 4
lda = LatentDirichletAllocation(
    n_components=n_topics, learning_method="batch", max_iter=10, random_state=42
)
document_topics = lda.fit_transform(X)
```

2.4 Exploring word topic association

rubric={points}

Your tasks:

1. For the number of topics you picked in the previous exercise, show top 10 words for each of your topics and suggest labels for each of the topics (similar to how we came up with labels "health and nutrition", "fashion", and "machine learning" in the toy example we saw in class).

> If your topics do not make much sense, you might have to go back to preprocessing in Exercise 2.1, improve it, and train your LDA model again.

Solution_2_4

Points: 5

In [44]:
```
lda = LatentDirichletAllocation(n_components=4, random_state=42)
document_topics = lda.fit_transform(X)
sorting = np.argsort(lda.components_, axis=1)[:, ::-1]

mglearn.tools.print_topics(
    topics=range(4),
    feature_names=feature_names,
    sorting=sorting,
    topics_per_chunk=5,
    n_words=10,
)
```

```
topic 0       topic 1       topic 2       topic 3
--------      --------      --------      --------
num           god           num           num
file          people        people        email
email         num           say           game
program       think         year          play
use           know          think         team
window        believe       gun           season
image         jesus         know          new
entry         say           time          period
include       thing         come          la
server        time          good          goal
```

topic 0: computer science, topic 1: religion, topic 2: politics, topic 3: sports.

In [ ]:

2.5 Exploring document topic association

rubric={points}

Your tasks:

1. Show the document topic assignment of the first five documents from `df`.
2. Comment on the document topic assignment of the model.

Solution_2_5

Points: 5

In [45]:
```
first_five_topics = document_topics[:5]
for i, topic_probs in enumerate(first_five_topics):
    most_probable_topic = topic_probs.argmax()  # get the topic with the highest
    print(f"Document {i+1}: topic = {most_probable_topic}")
```

```
Document 1: topic = 2
Document 2: topic = 0
Document 3: topic = 1
Document 4: topic = 0
Document 5: topic = 2
```

According to:

- topic 0: computer science,
- topic 1: religion,
- topic 2: politics,
- topic 3: sports.

- Document 1: topic = politics
- Document 2: topic = computer science
- Document 3: topic = religion
- Document 4: topic = computer science
- Document 5: topic = politics

In [46]:
```
df["target_name"][:5]
```

Out[46]:
```
0       talk.politics.guns
1            comp.graphics
2            comp.graphics
3          comp.windows.x
4    talk.politics.mideast
Name: target_name, dtype: object
```

Here we can see document 1, 2, 4, 5 all correspond to the correct topics. There might not be very detailed information like either guns or mideast in politics, but the big theme are nearly all right.

Exercise 3: Short answer questions

---

rubric={points}

1. Briefly explain how content-based filtering works in the context of recommender systems.
2. Discuss at least two negative consequences of recommender systems.
3. What is transfer learning in natural language processing? Briefly explain.

Solution_3

Points: 6

1. content-based filtering filters out and then recommends the items to users based on the features of the items, like the movie themes for movies, and some demographic and preference information on users, such as age and education. The items similar to the items with higher ratings will be recommended to the users.
2. first negative consequence of recommender systems is the effect of filter bubbles, which limits a user's exposure to diverse viewpoints and information and narrows their thinkings. The second negative consequence can be reinforcing extreme views: Recommendation systems often optimize for engagement (likes, clicks, shares), which may recommend increasingly

extreme content since such content tends to capture attention more effectively.

3. Transfer learning leverages knowledge learned from one task or domain and apply it to another, typically related, task or domain. It involves training a model on a large, diverse dataset and then fine-tuning it on a specific target corpus (domain-specific dataset), such as healthcare or social media. This can improve performance and reduce training costs since general linguistic patterns have been learned.

Before submitting your assignment, please make sure you have followed all the instructions in the Submission instructions section at the top.