MSADS509 M5 UE Wang

ADS 509 Assignment 5.1: Topic Modeling

This notebook holds Assignment 5.1 for Module 5 in ADS 509, Applied Text Mining. Work through this notebook, writing code and answering questions where required.

In this assignment you will work with a categorical corpus that accompanies `nltk`. You will build the three types of topic models described in Chapter 8 of *Blueprints for Text Analytics using Python*: NMF, LSA, and LDA. You will compare these models to the true categories.

General Assignment Instructions

These instructions are included in every assignment, to remind you of the coding standards for the class. Feel free to delete this cell after reading it.

One sign of mature code is conforming to a style guide. We recommend the Google Python Style Guide. If you use a different style guide, please include a cell with a link.

Your code should be relatively easy-to-read, sensibly commented, and clean. Writing code is a messy process, so please be sure to edit your final submission. Remove any cells that are not needed or parts of cells that contain unnecessary code. Remove inessential `import` statements and make sure that all such statements are moved into the designated cell.

Make use of non-code cells for written commentary. These cells should be grammatical and clearly written. In some of these cells you will have questions to answer. The questions will be marked by a "Q:" and will have a corresponding "A:" spot for you. *Make sure to answer every question marked with a* `Q:` *for full credit.*

In [1]:
```python
#!python -V ## added
from nltk.corpus import brown
import nltk ## addded
nltk.download('brown')
from nltk.corpus import brown

#!pip install spacy -q
#!python -m spacy download en_core_web_sm -q

import warnings
warnings.filterwarnings("ignore")
```

```
[nltk_data] Downloading package brown to /Users/UE/nltk_data...
[nltk_data]   Package brown is already up-to-date!
```

In [2]:
```python
# These libraries may be useful to you

#!pip install pyLDAvis==3.4.1 --user  #You need to restart the Kernel after installation.
# You also need a Python version => 3.9.0

import numpy as np
import pandas as pd
from tqdm.auto import tqdm

import pyLDAvis
import pyLDAvis.lda_model
import pyLDAvis.gensim_models

import spacy
from sklearn.feature_extraction.text import TfidfVectorizer, CountVectorizer
from sklearn.decomposition import NMF, TruncatedSVD, LatentDirichletAllocation

from spacy.lang.en.stop_words import STOP_WORDS as stopwords

from collections import Counter, defaultdict

nlp = spacy.load('en_core_web_sm')
```

In [3]:
```python
# This function comes from the BTAP repo.

def display_topics(model, features, no_top_words=5):
    for topic, words in enumerate(model.components_):
        total = words.sum()
        largest = words.argsort()[::-1] # invert sort order
        print("\nTopic %02d" % topic)
        for i in range(0, no_top_words):
            print("  %s (%2.2f)" % (features[largest[i]], abs(words[largest[i]]*100.0/total)))
```

Getting to Know the Brown Corpus

Let's spend a bit of time getting to know what's in the Brown corpus, our NLTK example of an "overlapping" corpus.

In [4]:
```python
# categories of articles in Brown corpus
for category in brown.categories() :
    print(f"For {category} we have {len(brown.fileids(categories=category))} articles.")
```

```
For adventure we have 29 articles.
For belles_lettres we have 75 articles.
For editorial we have 27 articles.
For fiction we have 29 articles.
For government we have 30 articles.
For hobbies we have 36 articles.
For humor we have 9 articles.
For learned we have 80 articles.
For lore we have 48 articles.
For mystery we have 24 articles.
For news we have 44 articles.
For religion we have 17 articles.
For reviews we have 17 articles.
For romance we have 29 articles.
For science_fiction we have 6 articles.
```

Let's create a dataframe of the articles in of hobbies, editorial, government, news, and romance.

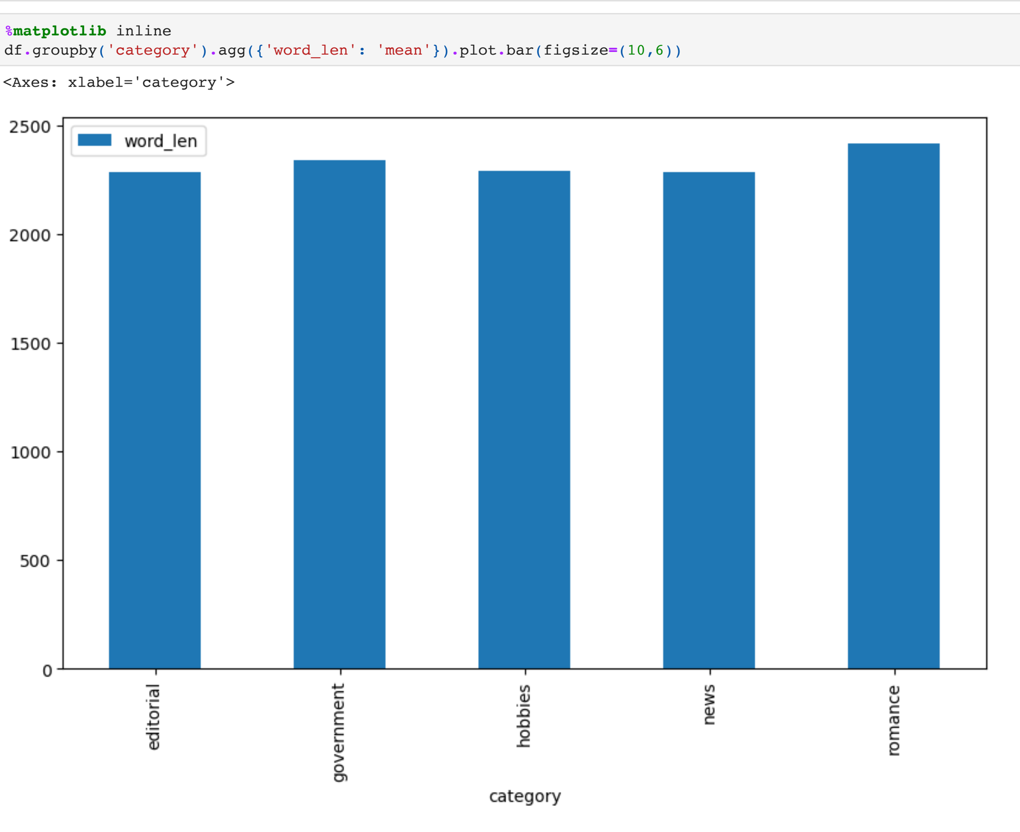
In [5]:
```python
categories = ['editorial','government','news','romance','hobbies']

category_list = []
file_ids = []
texts = []

for category in categories :
    for file_id in brown.fileids(categories=category) :

        # build some lists for a dataframe
        category_list.append(category)
        file_ids.append(file_id)

        text = brown.words(fileids=file_id)
        texts.append(" ".join(text))



df = pd.DataFrame()
df['category'] = category_list
df['id'] = file_ids
df['text'] = texts

df.shape
```

Out[5]:
```
(166, 3)
```

In [6]:
```python
# Let's add some helpful columns on the df
df['char_len'] = df['text'].apply(len)
df['word_len'] = df['text'].apply(lambda x: len(x.split()))
```

In [7]:
```python
%matplotlib inline
df.groupby('category').agg({'word_len': 'mean'}).plot.bar(figsize=(10,6))
```

Out[7]:
```
<Axes: xlabel='category'>
```



Now do our TF-IDF and Count vectorizations.

In [8]:
```python
count_text_vectorizer = CountVectorizer(stop_words=list(stopwords), min_df=5, max_df=0.7)
count_text_vectors = count_text_vectorizer.fit_transform(df["text"])
count_text_vectors.shape
```

Out[8]:
```
(166, 4941)
```

In [9]:
```python
tfidf_text_vectorizer = TfidfVectorizer(stop_words=list(stopwords), min_df=5, max_df=0.7)
tfidf_text_vectors = tfidf_text_vectorizer.fit_transform(df['text'])
tfidf_text_vectors.shape
```

Out[9]:
```
(166, 4941)
```

Q: What do the two data frames `count_text_vectors` and `tfidf_text_vectors` hold?

A:

count_text_vectors is a sparse matrix where rows represent documents, columns represent words, and each element indicates the frequency of a word in a document. On the other hand, tfidf_text_vectors is also a sparse matrix where rows represent documents, columns represent words, and each element indicates the TF-IDF score of a word in a document. TF-IDF reflects the importance of a word in a document relative to a collection of documents. Both matrices encode the information from df["text"], with count_text_vectors based on word counts (CountVectorizer) and tfidf_text_vectors based on TF-IDF scores (TfidfVectorizer).

Fitting a Non-Negative Matrix Factorization Model

In this section the code to fit a five-topic NMF model has already been written. This code comes directly from the BTAP repo, which will help you tremendously in the coming sections.

In [10]:
```python
nmf_text_model = NMF(n_components=5, random_state=314)
W_text_matrix = nmf_text_model.fit_transform(tfidf_text_vectors)
H_text_matrix = nmf_text_model.components_
```

In [11]:
```python
display_topics(nmf_text_model, tfidf_text_vectorizer.get_feature_names_out())
```

```
Topic 00
  mr (0.51)
  president (0.45)
  kennedy (0.43)
  united (0.42)
  khrushchev (0.40)

Topic 01
  said (0.88)
  didn (0.46)
  ll (0.45)
  thought (0.42)
  man (0.37)

Topic 02
  state (0.39)
  development (0.36)
  tax (0.33)
  sales (0.30)
  program (0.25)

Topic 03
  mrs (2.61)
  mr (0.78)
  said (0.63)
  miss (0.52)
  car (0.51)

Topic 04
  game (1.02)
  league (0.74)
  ball (0.72)
  baseball (0.71)
  team (0.66)
```

Now some work for you to do. Compare the NMF factorization to the original categories from the Brown Corpus.

We are interested in the extent to which our NMF factorization agrees or disagrees with the original categories in the corpus. For each topic in your NMF model, tally the Brown categories and interpret the results.

In [12]:
```python
import numpy as np
from collections import defaultdict, Counter

topic_to_category = defaultdict(list)

# Iterate over W_text_matrix
for idx, row in enumerate(W_text_matrix):

    topic = np.where(row == np.amax(row))[0][0]
    category = df["category"].iloc[idx]
    topic_to_category[topic].append(category)

# Iterate over topic_to_category and print results
for topic, categories in topic_to_category.items():

    category_counts = Counter(categories)
    total_count = sum(category_counts.values())
    print(f"\n Topic {topic} \n")
    print('\n'.join([f"  {category}: {count} ({(count / total_count) * 100:.2f}%)"
                     for category, count in category_counts.items()]))
```

```
 Topic 2

  editorial: 2 (3.08%)
  government: 26 (40.00%)
  news: 11 (16.92%)
  hobbies: 26 (40.00%)

 Topic 0

  editorial: 20 (62.50%)
  government: 4 (12.50%)
  news: 8 (25.00%)

 Topic 1

  editorial: 4 (9.76%)
  romance: 29 (70.73%)
  hobbies: 8 (19.51%)

 Topic 4

  editorial: 1 (10.00%)
  news: 8 (80.00%)
  hobbies: 1 (10.00%)

 Topic 3

  news: 17 (94.44%)
  hobbies: 1 (5.56%)
```

Q: How does your five-topic NMF model compare to the original Brown categories?

A:

Topic 0 contains a total of 32 documents, with 20 (which is 62.5%) falling under the category of "editorial". In Topic 1, the majority of documents (29 out of 41, accounting for 70.73%) are classified as "romance". Topic 2 demonstrates a dual association, with 40% of the documents classified as "government" and another 40% as "hobbies", each consisting of 26 documents. Within Topic 3, the category "news" dominates with 17 out of 18 documents, representing a significant proportion of 94.44%. Similarly, Topic 4 showcases a strong presence of "news" with 8 out of 10 documents, accounting for 80%. Furthermore, the co-occurrence of certain categories across topics indicates associations.

Topic 2 encompasses four categories, while topics 0, 1, and 4 include three categories each. Topic 3 captures two categories.

Fitting an LSA Model

In this section, follow the example from the repository and fit an LSA model (called a "TruncatedSVD" in `sklearn`). Again fit a five-topic model and compare it to the actual categories in the Brown corpus. Use the TF-IDF vectors for your fit, as above.

To be explicit, we are once again interested in the extent to which this LSA factorization agrees or disagrees with the original categories in the corpus. For each topic in your model, tally the Brown categories and interpret the results.

In [13]:
```python
# Your code here

svd_text_model = TruncatedSVD(n_components = 5, random_state=314)
W_svd_text_matrix = svd_text_model.fit_transform(tfidf_text_vectors)
H_svd_text_matrix = svd_text_model.components_
```

In [14]:
```python
topic_to_category = defaultdict(list)

# Iterate over W_text_matrix
for idx, row in enumerate(W_svd_text_matrix):

    topic = np.argmax(row)
    category = df["category"].iloc[idx]
    topic_to_category[topic].append(category)

# Iterate over topic_to_category and print results
for topic, categories in topic_to_category.items():

    category_counts = Counter(categories)
    total_count = sum(category_counts.values())
    print(f"\n Topic {topic} \n")
    print('\n'.join([f"  {category}: {count} ({(count / total_count) * 100:.2f}%)"
                     for category, count in category_counts.items()]))
```

```
 Topic 0

  editorial: 27 (18.24%)
  government: 30 (20.27%)
  news: 34 (22.97%)
  romance: 21 (14.19%)
  hobbies: 36 (24.32%)

 Topic 4

  news: 7 (100.00%)

 Topic 3

  news: 3 (100.00%)

 Topic 1

  romance: 8 (100.00%)
```

Q: How does your five-topic LSA model compare to the original Brown categories?

A:

Topics 1, 3, and 4 exhibit a clear correspondence with specific categories. Specifically, topics 3 and 4 are exclusively dedicated to the 'news' category. topic 0 captured a mixture of all five categories.

In [15]:
```python
# call display_topics on your model

display_topics(svd_text_model, tfidf_text_vectorizer.get_feature_names_out())
```

```
Topic 00
  said (0.44)
  mr (0.25)
  mrs (0.22)
  state (0.20)
  man (0.17)

Topic 01
  said (3.89)
  ll (2.73)
  didn (2.63)
  thought (2.20)
  got (1.97)

Topic 02
  mrs (3.12)
  mr (1.70)
  said (1.06)
  kennedy (0.82)
  khrushchev (0.77)

Topic 03
  mrs (29.45)
  club (6.53)
  game (6.12)
  jr (5.60)
  university (5.20)

Topic 04
  game (4.54)
  league (3.27)
  baseball (3.22)
  ball (3.10)
  team (2.94)
```

Q: What is your interpretation of the display topics output?

A:

Topics 00, 01, and 02 all prominently feature the word 'said,' suggesting a focus on conversation. In particular, Topic 02 includes terms like 'kennedy' and 'khrushchev,' indicating a likely emphasis on political conversation.

Topic 03 appears to be centered around entertainment, with words such as 'club,' 'game,' and 'university' indicating leisure activities or social gatherings.

Topic 04 is clearly about sports, with words like 'game,' 'ball,' 'baseball,' 'team,' and 'league' highlighting discussions related to sporting events or leagues.

Fitting an LDA Model

Finally, fit a five-topic LDA model using the count vectors (`count_text_vectors` from above). Display the results using `pyLDAvis.display` and describe what you learn from that visualization.

In [16]:
```python
# Fit your LDA model here

lda_text_model = LatentDirichletAllocation(n_components = 5, random_state=314)
W_lda_text_matrix = lda_text_model.fit_transform(count_text_vectors)
H_lda_text_matrix = lda_text_model.components_
```

In [17]:
```python
topic_to_category = defaultdict(list)

# Iterate over W_text_matrix
for idx, row in enumerate(W_lda_text_matrix):

    topic = np.argmax(row)
    category = df["category"].iloc[idx]
    topic_to_category[topic].append(category)

# Iterate over topic_to_category and print results
for topic, categories in topic_to_category.items():

    category_counts = Counter(categories)
    total_count = sum(category_counts.values())
    print(f"\n Topic {topic} \n")
    print('\n'.join([f"  {category}: {count} ({(count / total_count) * 100:.2f}%)"
                     for category, count in category_counts.items()]))
```

```
 Topic 2

  editorial: 21 (35.59%)
  government: 3 (5.08%)
  news: 32 (54.24%)
  romance: 1 (1.69%)
  hobbies: 2 (3.39%)

 Topic 0

  editorial: 3 (6.38%)
  government: 1 (2.13%)
  news: 4 (8.51%)
  romance: 28 (59.57%)
  hobbies: 11 (23.40%)

 Topic 3

  editorial: 2 (11.76%)
  government: 4 (23.53%)
  news: 3 (17.65%)
  hobbies: 8 (47.06%)

 Topic 1

  editorial: 1 (4.00%)
  government: 12 (48.00%)
  news: 3 (12.00%)
  hobbies: 9 (36.00%)

 Topic 4

  government: 10 (55.56%)
  news: 2 (11.11%)
  hobbies: 6 (33.33%)
```

In [18]:
```python
# Call `display_topics` on your fitted model here

display_topics(lda_text_model, tfidf_text_vectorizer.get_feature_names_out())
```

```
Topic 00
  said (1.05)
  mrs (0.82)
  little (0.56)
  good (0.51)
  way (0.50)

Topic 01
  state (0.67)
  development (0.63)
  000 (0.57)
  program (0.48)
  business (0.44)

Topic 02
  said (1.18)
  mr (0.72)
  president (0.51)
  city (0.43)
  state (0.37)

Topic 03
  feed (0.55)
  college (0.54)
  general (0.44)
  university (0.43)
  work (0.37)

Topic 04
  states (1.14)
  state (1.02)
  united (0.84)
  shall (0.66)
  government (0.61)
```

Q: What inference do you draw from the displayed topics for your LDA model?

A:

Topic 00: Appears diverse, potentially involving various contexts.

Topic 01: Focuses on business and development, evident from words like 'state,' 'development,' 'program,' and 'business'.

Topic 02: Government-related, indicated by terms like 'president,' 'city,' and 'state'.

Topic 03: Likely about education, with words such as 'college' and 'university' suggesting educational institutions or activities.

Topic 04: Also government-related, with terms like 'states', 'united', and 'government' pointing towards governmental structures or policies.

Q: Repeat the tallying of Brown categories within your topics. How does your five-topic LDA model compare to the original Brown categories?

A:

Topic 0: Appears to be predominantly about romance, with 59.57% of documents categorized as such. This topic also contains some documents from the hobbies category.

Topic 1: This topic seems to be primarily associated with government-related documents, constituting 48% of the total. It also contains some documents from the hobbies category.

Topic 2: Contains a significant proportion of news articles (54.24%), along with some editorial articles and a small number of documents from other categories.

Topic 3: Consists mostly of documents from the hobbies category, comprising 47.06% of the total. It also includes government-related documents.

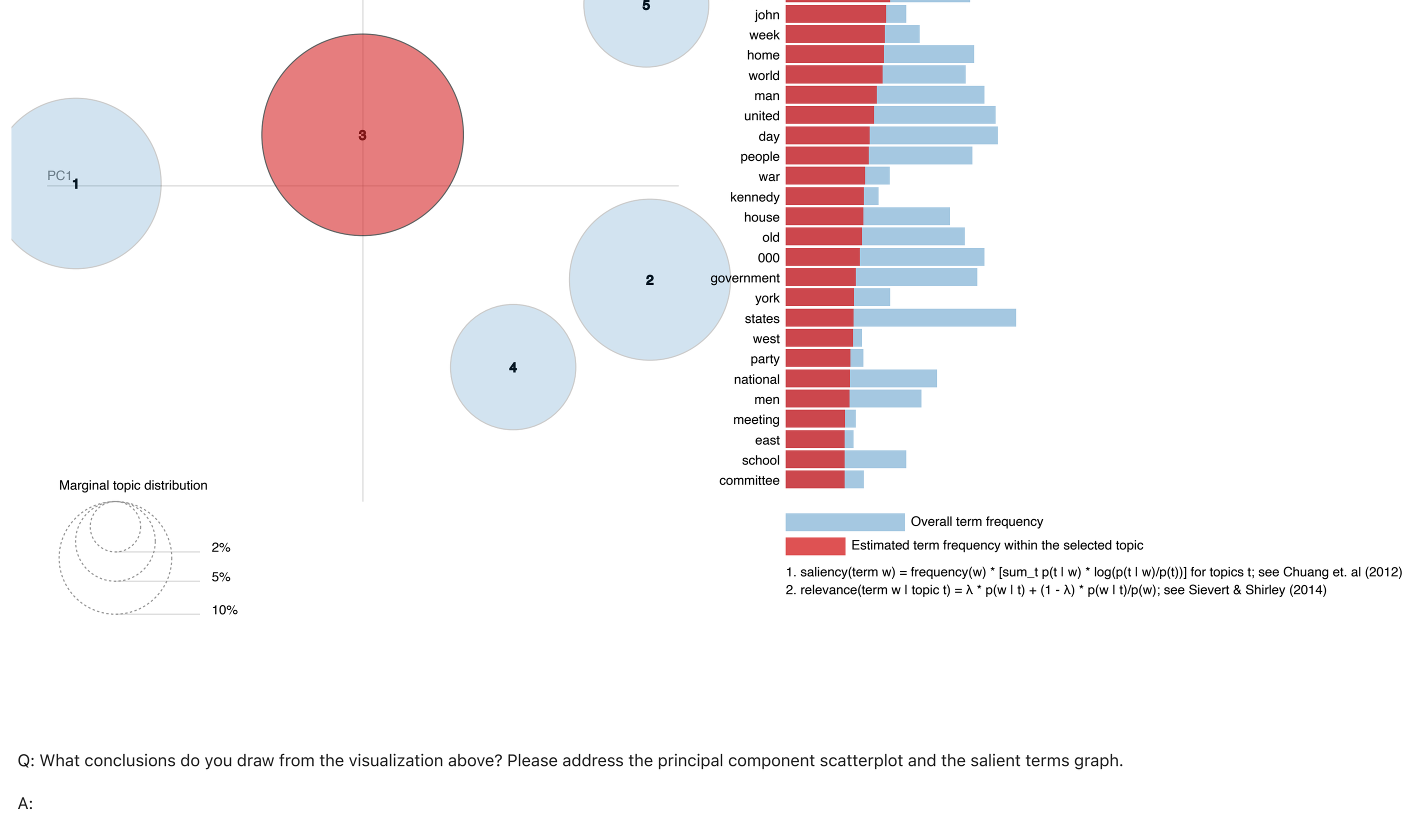
Topic 4: Dominated by government-related documents (55.56%), with a smaller proportion of hobbies and news articles.

Both Topic 1 and Topic 4 are primarily associated with government-related documents. Editorial doesn't appear to dominate in any topic. This LDA model captures categories in each topic more distinctly compared to previous two models.

In [19]:
```python
lda_display = pyLDAvis.lda_model.prepare(lda_text_model, count_text_vectors, count_text_vectorizer, sort_topics=False)
```

In [20]:
```python
pyLDAvis.display(lda_display)
```

Out[20]:



Q: What conclusions do you draw from the visualization above? Please address the principal component scatterplot and the salient terms graph.

A:

Each topic circle in the pyLDAvis visualization represents a distinct topic, and they do not overlap. Topics 2 and 4 are positioned very close to each other on the scatterplot, indicating that they are more similar in terms of their word distributions.

Topic 3, being closest to the center circle, has the lowest marginal topic distribution among all topics, indicating its lesser dominance in the corpus compared to other topics. This is supported by salient terms such as 'president', 'government', 'school', 'house', and 'party', suggesting a broad and diverse range of words within it.