KM_Topic Modeling

October 6, 2024

1 Katie Mears

2 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.

2.1 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.*

[38]:
```
%%capture
!pip install pypandoc
!apt-get install texlive texlive-xetex texlive-latex-extra pandoc
```

[41]:
```
from google.colab import drive
drive.mount('/content/drive')
```

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

[3]:
```
# !pip install pyLDAvis==3.4.1 --user  #You need to restart the Kernel after␣
↪installation.
```

```
# You also need a Python version => 3.9.0
```

[1]:
```
# These libraries may be useful to you


from nltk.corpus import brown

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
import en_core_web_sm

from collections import Counter, defaultdict

nlp = en_core_web_sm.load()
```

[13]:
```
# add any additional libaries you need here

import nltk
nltk.download('brown')

import warnings

# Suppress DeprecationWarning
warnings.filterwarnings("ignore", category=DeprecationWarning)
```

```
/usr/local/lib/python3.10/dist-packages/ipykernel/ipkernel.py:283:
DeprecationWarning: `should_run_async` will not call `transform_cell`
automatically in the future. Please pass the result to `transformed_cell`
argument and any exception that happen during thetransform in
`preprocessing_exc_tuple` in IPython 7.17 and above.
  and should_run_async(code)
[nltk_data] Downloading package brown to /root/nltk_data…
[nltk_data]   Package brown is already up-to-date!
```

[14]:
```
# 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)))
```

2.2 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.

[15]:
```
# 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.

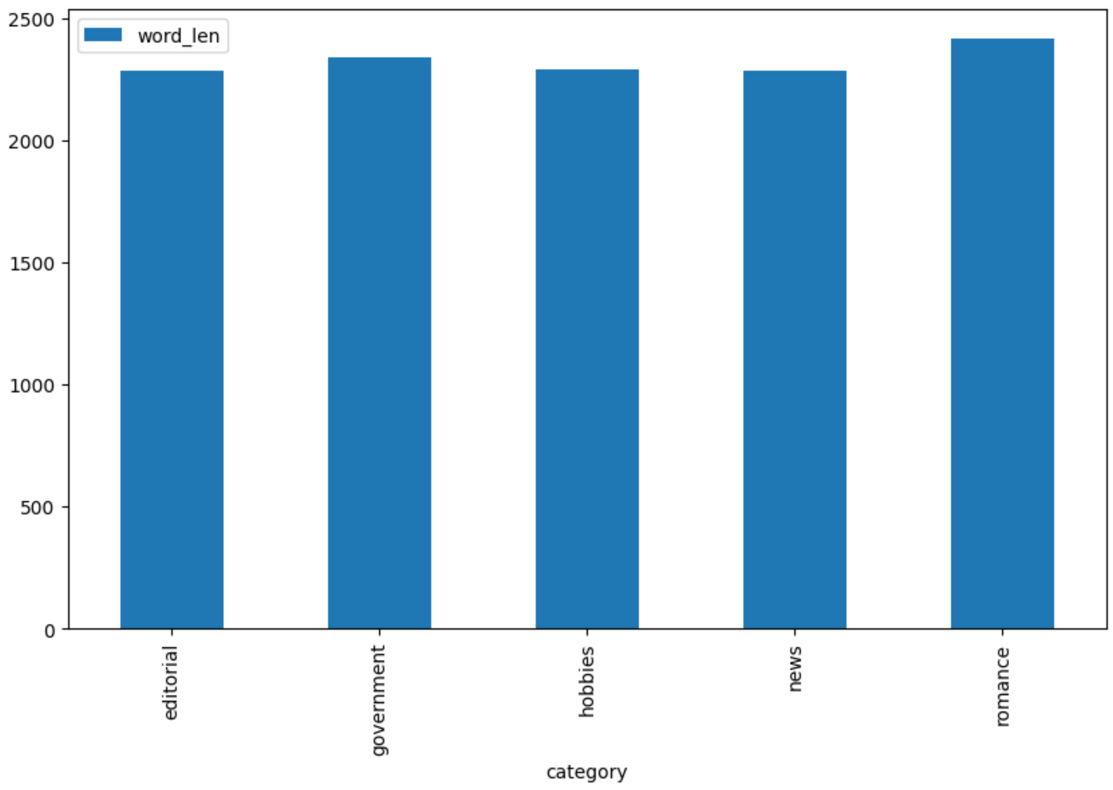
[16]:
```
categories = ['editorial','government','news','romance','hobbies']

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
```

[16]:
```
(166, 3)
```

[17]:
```
# 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()))
```

[18]:
```
%matplotlib inline
df.groupby('category').agg({'word_len': 'mean'}).plot.bar(figsize=(10,6))
```

[18]:
```
<Axes: xlabel='category'>
```



Now do our TF-IDF and Count vectorizations.

```
[19]: 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
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/feature_extraction/text.py:406:
UserWarning: Your stop_words may be inconsistent with your preprocessing.
Tokenizing the stop words generated tokens ['ll', 've'] not in stop_words.
  warnings.warn(
```

```
[19]: (166, 4941)
```

```
[20]: 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
```

```
[20]: (166, 4941)
```

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

A: The data frame `count_text_vectors` is a holding the count based representation of the text data. The values in this matrix are essentially word counts, excluding stop words and only accounting for words that appear in more than 5 documents and no more than 70% of the documents.

The data frame `tfidf_text_vectors` is holding the TF-IDF representation of the data. In this data frame, each element represents a weighted score for how important a word is in the document relative to its importance across the entire corpus. These scores help to apply less weight to the commonly used words and more weight to the more rare and more meaningful words. It is also only accounting for words that appear in more than 5 documents and no more than 70% of the documents.

2.3 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.

```
[21]: 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_
```

```
[22]: 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 categorization to the original categories from the Brown Corpus.

We are interested in the extent to which our NMF categorization 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.

[23]:
```python
# Your code here
nmf_topics = np.argmax(W_text_matrix, axis=1)
nmf_df = pd.DataFrame({
    'topic': nmf_topics,
    'brown_category': df['category']
})
```

```
tally = nmf_df.groupby(['topic', 'brown_category']).size().unstack(fill_value=0)
print(tally)
```

```
brown_category  editorial  government  hobbies  news  romance
topic
0                      20           4        0     8        0
1                       4           0        8     0       29
2                       2          26       26    11        0
3                       0           0        1    17        0
4                       1           0        1     8        0
```

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

A: The NMF model seems to align well with the original brown categories. The brown tally tells us that topic 0 is related to editorial corpi with some news corpi and also government. Based on the display topics function for the NMF model, we see that the top words are for Mr, president and kennedy, which suggests that there is government topics, perhaps also news articles which seem to be aligned with the brown categories. Topic 1 in the brown tally seems to be related to romance, hobbies and editorial, which is aligned with the top words in the NMF model. Topic 2 of the brown tally seems to be related to government and hobbies, news and editorial which is somewhat aligned with the top words for NMF model. in the NMF model for topic 2, goverment words seems to be the most frequent. Topic 3 is less similar when comparing the two. Topic 4 in the NMF model seems related to baseball and/or sports, and the brown tally seems to indicate that the categories were news, hobbies and editorial. I would have expected to see more counts in the hobbies column but it could be articles from news sources related to baseball or sports.

2.4 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.

```
[29]: # Fit the LSA model
lsa_model = TruncatedSVD(n_components=5, random_state=314)
lsa_topic_matrix = lsa_model.fit_transform(tfidf_text_vectors)
lsa_components = lsa_model.components_
```

```
[28]: # Create a DataFrame to tally the Brown categories for LSA topics
lsa_topic_brown_tally = pd.DataFrame(0, index=range(5), columns=['editorial',␣
  ↪'government', 'hobbies', 'news', 'romance'])

for i, topic in enumerate(lsa_topic_matrix):
    category = df['category'][i]
    lsa_topic_brown_tally.loc[topic.argmax(), category] += 1
```

```
# Display the tally
lsa_topic_brown_tally
```

[28]:
```
   editorial  government  hobbies  news  romance
0         27          30       36    34       21
1          0           0        0     0        8
2          0           0        0     0        0
3          0           0        0     3        0
4          0           0        0     7        0
```

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

A: It doesnt do as well as the NMF categorization. We see that topic 0 has substantical representation across multiple categories, but doesnt clearly align with the original brown categories for topic 0. For topic 0 in the brown categories, our tally showed that there was mostly editorial topics, with some news and some government. There were no hobbies nor romance articles found. In the LSA model, it shows the highest counts for hobbies and also 21 for romance which doesnt align with browns categories. We also had 0 for all categories for topic 1 and 2, further proving that there is disalignment between the original Brown categories and the LSA model.

[32]:
```
# call display_topics on your model
display_topics(lsa_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: Topic 0 seems to be some type of dialogue or conversations, where people are addressed as Mr. and Mrs. relatively often. The top word being said, indicated to me that there is descriptions of what people said and the word state, indicates to me that the conversations may have been about United States or government related issues. Topic 1 seems to be personal narratives or opinions, based on the workds thought and didnt. Topic 2 seems to be political or history related topics. Also Using Mr. and Mrs. which indicates a formality in how people are being referred to. Topic 3 seems to be related to activities or organizations, perhaps school related events. Topic 4 seems to be based on athetics or sports events, mostly baseball events.

2.5 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.

[33]:
```python
# Fit your LDA model here
lda_model = LatentDirichletAllocation(n_components=5, random_state=314)
lda_topic_matrix = lda_model.fit_transform(count_text_vectors)
lsa_components = lsa_model.components_
```

[34]:
```python
# Call `display_topics` on your fitted model here
display_topics(lda_model, count_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)
```

[35]:
```
# Create a DataFrame to tally the Brown categories for LDA topics
lda_topic_brown_tally = pd.DataFrame(0, index=range(5), columns=['editorial',␣
 ↪'government', 'hobbies', 'news', 'romance'])

for i, topic in enumerate(lda_topic_matrix):
    category = df['category'][i]
    lda_topic_brown_tally.loc[topic.argmax(), category] += 1

# Display the tally
lda_topic_brown_tally
```

[35]:
```
   editorial  government  hobbies  news  romance
0          3           1       11     4       28
1          1          12        9     3        0
2         21           3        2    32        1
3          2           4        8     3        0
4          0          10        6     2        0
```

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

A: Topic 0 seems to be similar to that of the LSA model, it seems to be personal narratives or dialogues, where Mrs. and said are used. Topic 1 seems to be related to government or business related topics, which is not similar to the LSA model. Topic 2 seems to be related to government with words such as president and Mr. being popular, which is somewhat similar to LSA but not the exact terms. Topic 3 is related to schools, universities which is also similar to the LSA model. Topic 4 in LDA seems to also be related to government which is dissimilar to LSA, as LSA seemed to be related to baseball.

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

A: I would say that the LDA did a better job in LSA when comparing the models to the original Browns categories, there are still some discrepancies but it does capture a lot of the same categories.

[36]:
```
lda_display = pyLDAvis.lda_model.prepare(lda_model, count_text_vectors,␣
 ↪count_text_vectorizer, sort_topics=False)
```

[37]:
```
pyLDAvis.display(lda_display)
```

[37]:
```
<IPython.core.display.HTML object>
```

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

A: This is a really cool visulaization! Each point on the scatterplot represents a topics that was identified by the LDA model. The proximity between the points indicates teh similarity between the topics, and the size seems to indicates the proportion of documents that are assigned to the topic. For Topic 0, there is a large portion of documents that were grouped into cluster 3, and based on the salient terms graph, this cluster seems to be most related to government, but the top word being said, so it does seem to indicate that there are interpersonal discussions about government, president, etc. Cluster 1 seems to be a large cluster as well and based on the salient terms, this seems to capture a lot of the words that indicate interpersonal conversation as well and aligns with what we saw when we displayed the topics for the LDA model for topic 0. Cluster 2 is the third biggest, and seems to represent business development topics. At the top, you can select the topic and see the results for each of the topics.

[42]:
```
%%capture
!jupyter nbconvert --to pdf "/content/drive/MyDrive/KM_Topic Modeling.ipynb"
```