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 [108…

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

import nltk
nltk.download('brown')
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

from collections import Counter, defaultdict

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

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

In [110…

```
# add any additional libaries you need here
```

In [112…

```
# 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 [21]:
```
# 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.

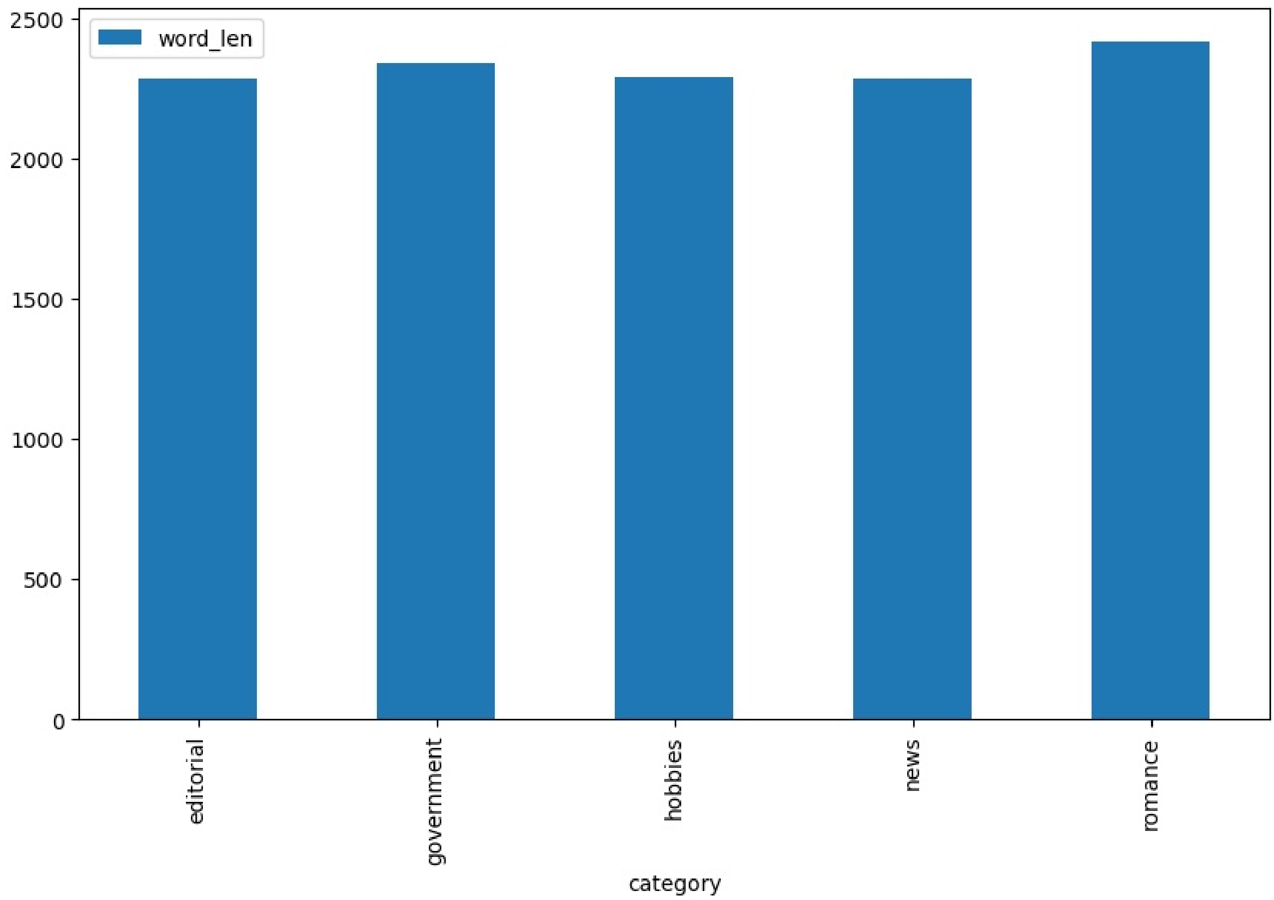
In [24]:
```
categories = ['editorial','government','news','romance','hobbies']

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[24]:
```
(166, 3)
```

In [26]:
```
# 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 [28]:
```
%matplotlib inline
df.groupby('category').agg({'word_len': 'mean'}).plot.bar(figsize=(10,6))
```

Out[28]:
```
<AxesSubplot:xlabel='category'>
```



Now do our TF-IDF and Count vectorizations.

In [31]:
```
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
```

```
/Users/parisakamizi/opt/anaconda3/lib/python3.9/site-packages/sklearn/feature_extraction/text.py:396: UserWarnin
g: Your stop_words may be inconsistent with your preprocessing. Tokenizing the stop words generated tokens ['ll'
, 've'] not in stop_words.
  warnings.warn(
```

Out[31]: (166, 4941)

In [33]:
```
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[33]: (166, 4941)

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

A: count_text_vectors creates a Bag-of-Words model, where each row represents a document, each column represents a unique word, and each cell contains the count of how many times the word appears in the document—without considering its importance. In contrast, tfidf_text_vectors assigns a score to each word based on its importance, balancing its frequency within a document against how commonly it appears across all documents.

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 [37]:
```
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_
```

```
/Users/parisakamizi/opt/anaconda3/lib/python3.9/site-packages/sklearn/decomposition/_nmf.py:289: FutureWarning:
The 'init' value, when 'init=None' and n_components is less than n_samples and n_features, will be changed from
'nndsvd' to 'nndsvda' in 1.1 (renaming of 0.26).
  warnings.warn(
```

In [39]:
```
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 (0.40)
  development (0.36)
  tax (0.33)
  sales (0.30)
  program (0.25)

Topic 03
  mrs (2.61)
  mr (0.78)
  said (0.64)
  miss (0.52)
  car (0.51)

Topic 04
  game (1.01)
  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 [44]:
```
# Resources can be found from here --> https://www.nltk.org/howto/corpus.html
# https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.NMF.html
# https://www.analyticsvidhya.com/blog/2021/06/part-15-step-by-step-guide-to-master-nlp-topic-modelling-using-nm
# and the Textbook

topic_category_map = defaultdict(list)

# Assign each document to its most dominant NMF topic
for idx, row in enumerate(W_text_matrix):
    topic = np.argmax(row)
    category = df["category"].iloc[idx]
    topic_category_map[topic].append(category)

for topic, categories in topic_category_map.items():
    print(f"\nFor Topic {topic}, we have {len(categories)} documents.")
    print("Top 5 categories:", Counter(categories).most_common(5))
```

```
For Topic 2, we have 65 documents.
Top 5 categories: [('government', 26), ('hobbies', 26), ('news', 11), ('editorial', 2)]

For Topic 0, we have 32 documents.
Top 5 categories: [('editorial', 20), ('news', 8), ('government', 4)]

For Topic 1, we have 41 documents.
Top 5 categories: [('romance', 29), ('hobbies', 8), ('editorial', 4)]

For Topic 4, we have 10 documents.
Top 5 categories: [('news', 8), ('editorial', 1), ('hobbies', 1)]

For Topic 3, we have 18 documents.
Top 5 categories: [('news', 17), ('hobbies', 1)]
```

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

A: The analysis suggests that some categories could be merged, while others contain outliers. Editorial, news, and government often appear together, indicating shared themes, whereas hobbies spans multiple topics, suggesting diverse subtopics. Some categories, like editorial, also appear in unexpected topics. I think the model may need fine-tuning or better preprocessing.

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 [54]:
```python
# Your code here
# Resources can be found from here --> https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.

# Fit an LSA model with 5 topics
lsa_model = TruncatedSVD(n_components=5, random_state=314)
lsa_W_matrix = lsa_model.fit_transform(tfidf_text_vectors)
lsa_H_matrix = lsa_model.components_

# Display the top words
display_topics(lsa_model, tfidf_text_vectorizer.get_feature_names_out())

lsa_topic_to_category = defaultdict(list)

for idx, row in enumerate(lsa_W_matrix):
    topic = np.argmax(row)
    category = df["category"].iloc[idx]
    lsa_topic_to_category[topic].append(category)

for topic, categories in lsa_topic_to_category.items():
    print(f"\nFor Topic {topic}, we have {len(categories)} documents.")
    print("Top 5 categories:", Counter(categories).most_common(5))
```

```
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)

For Topic 0, we have 148 documents.
Top 5 categories: [('hobbies', 36), ('news', 34), ('government', 30), ('editorial', 27), ('romance', 21)]

For Topic 4, we have 7 documents.
Top 5 categories: [('news', 7)]

For Topic 3, we have 3 documents.
Top 5 categories: [('news', 3)]

For Topic 1, we have 8 documents.
Top 5 categories: [('romance', 8)]
```

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

A: This model did not perform very well. Topic 0 is too broad, covering 148 documents across multiple categories, which suggests that the model struggles to differentiate between them. On the other hand, Topics 3 and 4 contain only news articles and could potentially be merged. This indicates that the model might need better parameter tuning or additional preprocessing,

In [52]:
```
# call display_topics on your model

# Already displayed above.
```

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

A: The topics generated by LSA model indicate some overlap and lack of clear separation between distinct themes. Topic 0 appears to be a mix of general words that are not strongly tied to a specific subject. Topic 1 is dominated by conversational words, making it hard to determine its category. Topic 2, with names like "Kennedy" and "Khrushchev," suggests a political theme. Topic 4 is related to sports, while Topic 3, with words like "Mrs.," "Mr.," "university," and "club," is also difficult to categorize.

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 [87]:
```
from sklearn.decomposition import LatentDirichletAllocation

# Fit the LDA model with 6 topics
lda_text_model = LatentDirichletAllocation(n_components=6, random_state=314)
W_lda_text_matrix = lda_text_model.fit_transform(count_text_vectors)  # Document-topic distribution
H_lda_text_matrix = lda_text_model.components_  # Topic-word distribution
```

In [95]:
```
# Call `display_topics` on your fitted model here
display_topics(lda_text_model, count_text_vectorizer.get_feature_names())
```

```
Topic 00
  mrs (1.80)
  water (0.71)
  clay (0.67)
  use (0.64)
  shelter (0.59)

Topic 01
  business (0.58)
  state (0.58)
  1960 (0.49)
  development (0.48)
  sales (0.47)

Topic 02
  mr (0.89)
  president (0.77)
  united (0.53)
  american (0.52)
  said (0.50)

Topic 03
  feed (0.61)
  college (0.60)
  university (0.50)
  work (0.42)
  student (0.37)

Topic 04
  state (1.23)
  states (1.00)
  tax (0.73)
  united (0.69)
  government (0.57)

Topic 05
  said (1.69)
  old (0.51)
  little (0.48)
  man (0.47)
  ll (0.44)
```

In [97]:

```
topic_category_mapping = defaultdict(list)

for idx, row in enumerate(W_lda_text_matrix):
    topic = np.where(row == np.amax(row))[0]
    category = df["category"].iloc[idx]
    topic_category_mapping[topic[0]].append(category)

for topic, categories in topic_category_mapping.items():
    print(f"\nFor Topic {topic}, we have {len(categories)} documents.")
    print("Top 5 categories:", Counter(categories).most_common(5))
```

```
For Topic 4, we have 21 documents.
Top 5 categories: [('government', 11), ('news', 5), ('hobbies', 3), ('editorial', 2)]

For Topic 2, we have 37 documents.
Top 5 categories: [('editorial', 19), ('news', 11), ('government', 4), ('hobbies', 2), ('romance', 1)]

For Topic 5, we have 57 documents.
Top 5 categories: [('romance', 28), ('news', 17), ('hobbies', 9), ('editorial', 3)]

For Topic 3, we have 18 documents.
Top 5 categories: [('hobbies', 9), ('news', 4), ('editorial', 3), ('government', 2)]

For Topic 1, we have 21 documents.
Top 5 categories: [('government', 12), ('hobbies', 5), ('news', 4)]

For Topic 0, we have 12 documents.
Top 5 categories: [('hobbies', 8), ('news', 3), ('government', 1)]
```

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

A: The displayed topics suggest distinct themes. Topic 00 is hard to distinguish by category. Topic 01 focuses on business and economic development, mentioning "business," "state," "development," and "sales." Topic 02 seems political, containing "president," "united," "american," and "said," common in speeches or news. Topic 03 is linked to education, with words like "college," "university," "work," and "student." Topic 04 revolves around government and taxation, featuring "state," "tax," and "government." Topic 05 is heavily dialogue-driven but hard to distinguish by category.

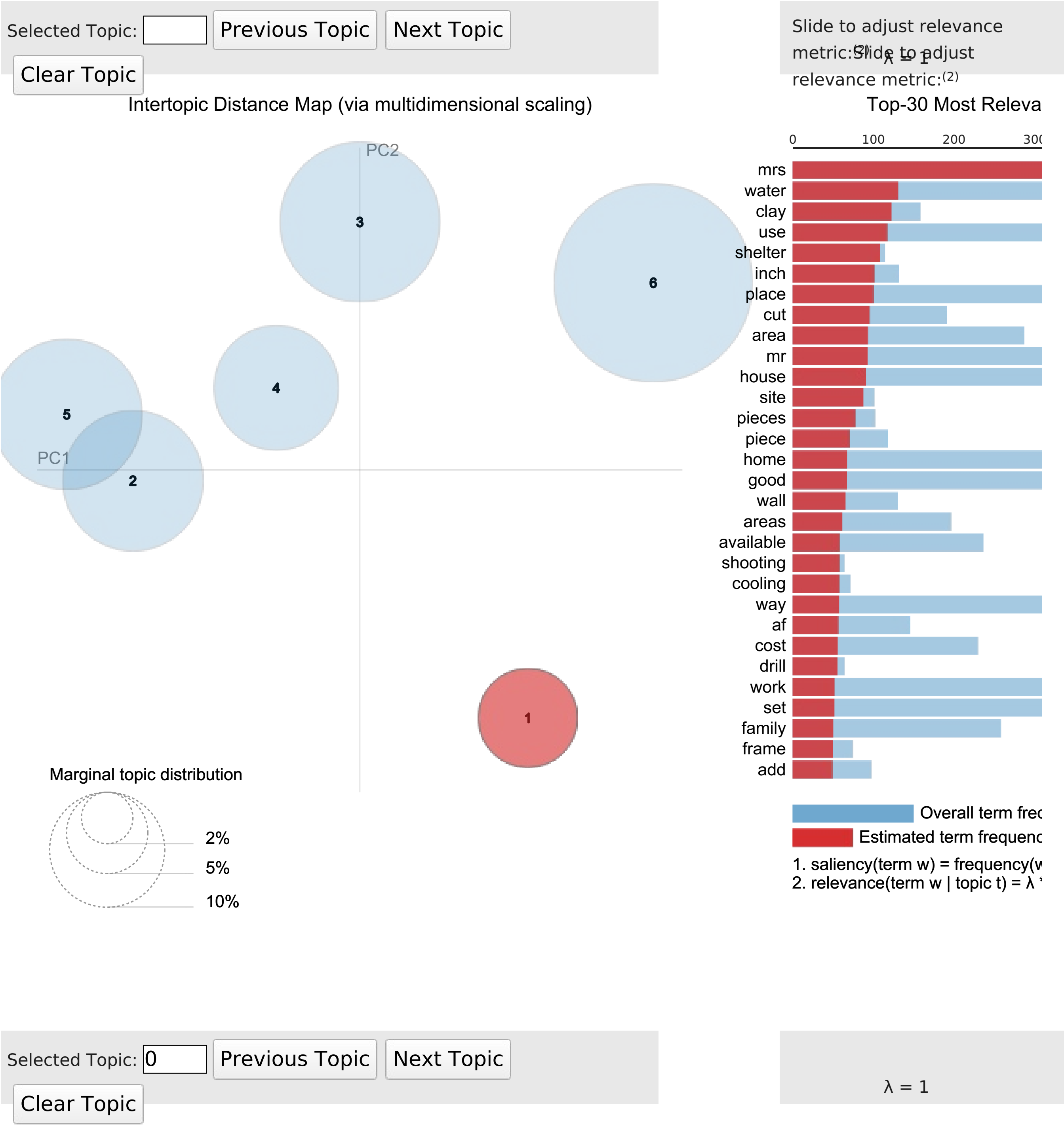
Q: Repeat the tallying of Brown categories within your topics. How does your five-topic LDA model compare to the original Brown categories?

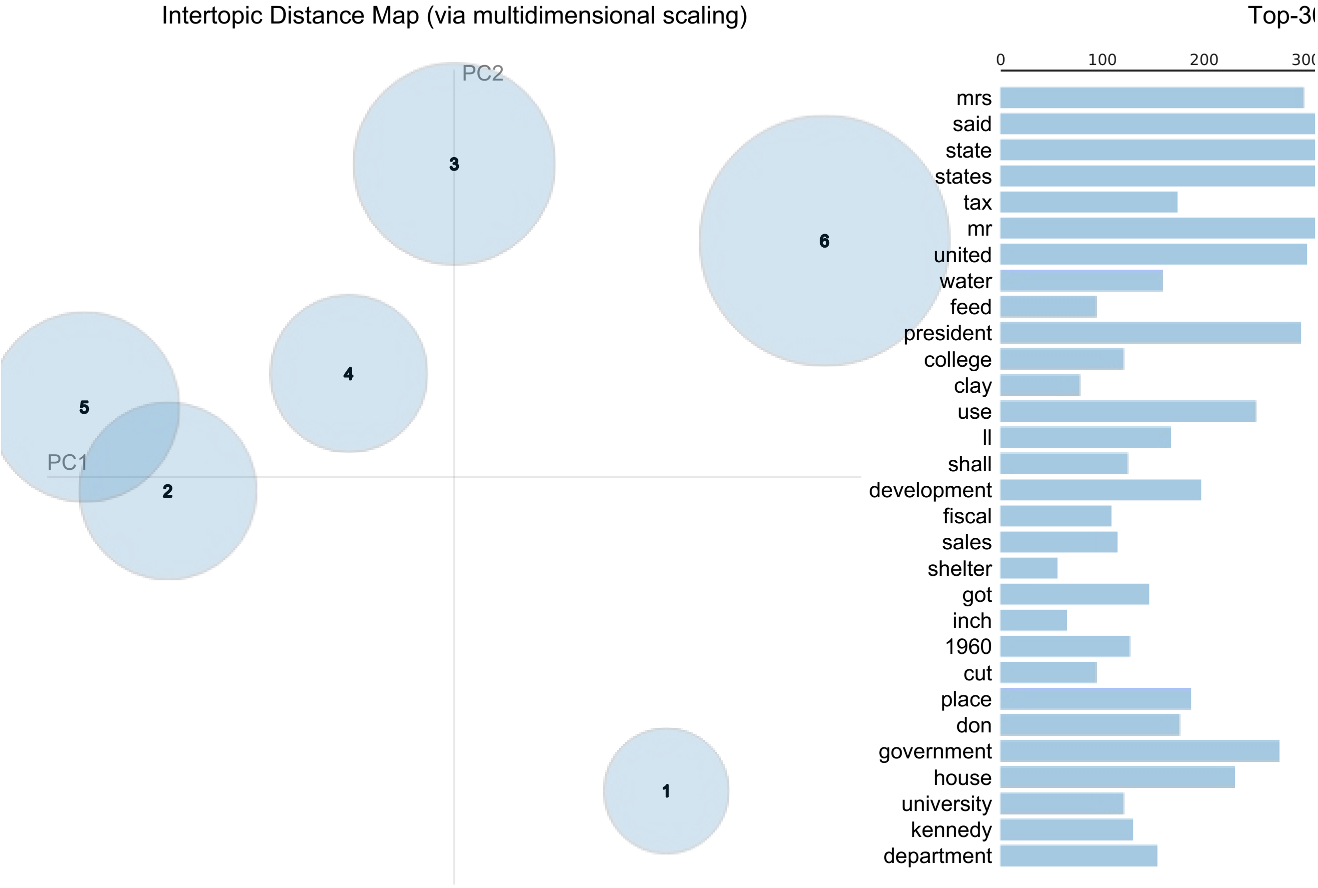
A: The five-topic LDA model reveals a mixture of categories, with some that could be merged. Topic 4 is strongly associated with government and news. Topic 2 is dominated by editorial and news content, but it is hard to distinguish a clear category. Topic 5 has a strong romance presence but also contains news, making it difficult to classify. Topic 3 is primarily hobbies-focused. Topic 1 aligns closely with government, and Topic 0 is mainly hobbies-driven. While some topics align with Brown categories, there is significant overlap, indicating that the model would benefit from better preprocessing or fine-tuning.

In [100…
```
lda_display = pyLDAvis.lda_model.prepare(lda_text_model, count_text_vectors, count_text_vectorizer, sort_topics=
```

In [101…
```
pyLDAvis.display(lda_display)
```

Out[101…





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

A: From the visualization, it is evident that the model requires better cleaning, preprocessing, or fine-tuning, as many of the words do not provide meaningful insights. The principal component scatterplot shows some overlap, particularly between Topics 5 and 2, suggesting that these topics are not well-separated. Additionally, Topics 1 and 6 are positioned farther from the rest, with Topic 6 being the largest. Notably, Topic 6 has "said" as the most frequently occurring word, significantly more than any other term, which may indicate that it is not contributing to meaningful topic differentiation. Similarly, in Topic 1, "mrs" appears with the highest frequency, but it does not provide much interpretive value, and its occurrence is disproportionately higher than other words in that topic. Topic 3 contains "000," which also lacks meaningful context. The overlap between Topics 5 and 2 suggests a connection to business-related themes. Overall, the visualization highlights the need for improved preprocessing to refine the topics and ensure they capture more meaningful distinctions.

```
In [ ]:
```