Latent Dirichlet Allocation for Domain-Specific Language Modeling in Underwater Biology

In this study, I constructed and preprocessed a specialized text corpus to support the development of domain-specific word embeddings for underwater biology, with a focus on coral reef ecosystems. Wikipedia articles pertaining to key reef-building coral taxa—Acropora, Black Coral, Montipora, Pocillopora, Porites, and Scleractinia—were selected to ensure topical relevance and scientific depth. The goal of this corpus construction is to enhance the semantic precision of large language models in marine and ecological contexts.

Raw textual data were tokenized using a regular expression-based tokenizer to extract alphanumeric word units. Standard English stopwords were removed to reduce lexical noise and focus on semantically meaningful content. Tokens were filtered to include only alphabetic characters, and each cleaned article was stored as a processed text unit for downstream analysis. Throughout the preprocessing pipeline, I computed key corpus statistics, including the number of articles, paragraph counts, total token counts, and the vocabulary size (i.e., number of unique tokens).

The cleaned and tokenized corpus will serve as input for subsequent vectorization and topic modeling procedures, such as TF-IDF transformation and Latent Dirichlet Allocation (LDA), to uncover latent semantic structures within the data. These representations aim to support the training of fine-tuned language models for marine biology, ultimately contributing to improved performance in domain-specific information retrieval, question answering, and knowledge discovery applications.

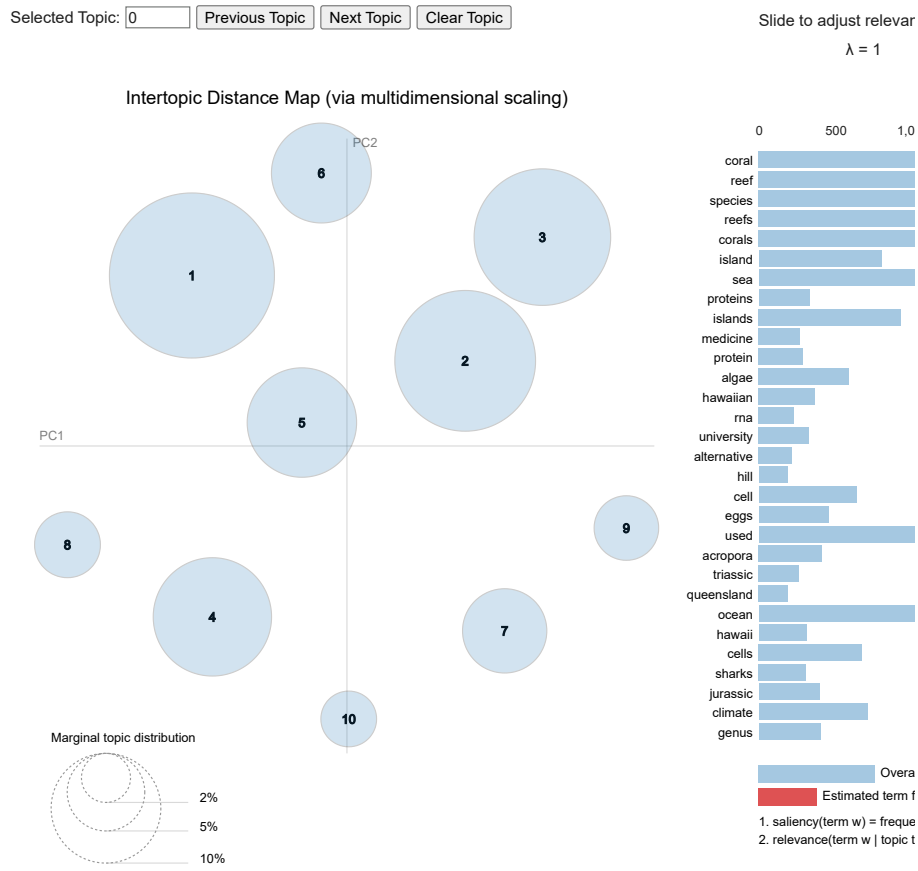
```
import os
from nltk import ne_chunk, pos_tag, word_tokenize
from sklearn.feature_extraction.text import TfidfVectorizer, CountVectorizer
from nltk.tokenize import RegexpTokenizer
from sklearn.decomposition import LatentDirichletAllocation
from nltk.corpus import stopwords
import pyLDAvis
from pyLDAvis.lda_model import prepare as lda_prepare
```

```
cleaned_text=[]
tokenizer = RegexpTokenizer(r"\w+")
# create English stop words list
englishfile = open("englishvocabulary.txt", encoding="utf8")
engstop = englishfile.read()
engstop = set(stopwords.words("english"))
files=os.listdir('wiki_corpus')
token_set=set()
file_count=0
paragraph_count=0
word_count=0
center_articles="[Acropora, Black Coral, Montipora, Pocillopora, Porites, Scleractinia]"
for file in files:
    with open(f'wiki_corpus/{file}','r') as f:
        raw=f.read().lower()
    tokens=tokenizer.tokenize(raw)
    tokens = [i for i in tokens if not i in engstop and i.isalpha()]
    token_set.update(tokens)
    file_count+=1
    paragraph_count+=raw.count('\n')
    word_count+=len(tokens)
    newfile=' '.join(tokens)
    cleaned_text.append(newfile)
print("Articles: ", file_count)
print("Paragraphs:", paragraph_count)
print("Tokens:", word_count)
print("Unique tokens:", len(token_set))
print("Central Articles:", center_articles)
```

```
Articles:  492
Paragraphs: 12924
Tokens: 500297
Unique tokens: 41163
Central Articles: [Acropora, Black Coral, Montipora, Pocillopora, Porites, Scleractinia]
```

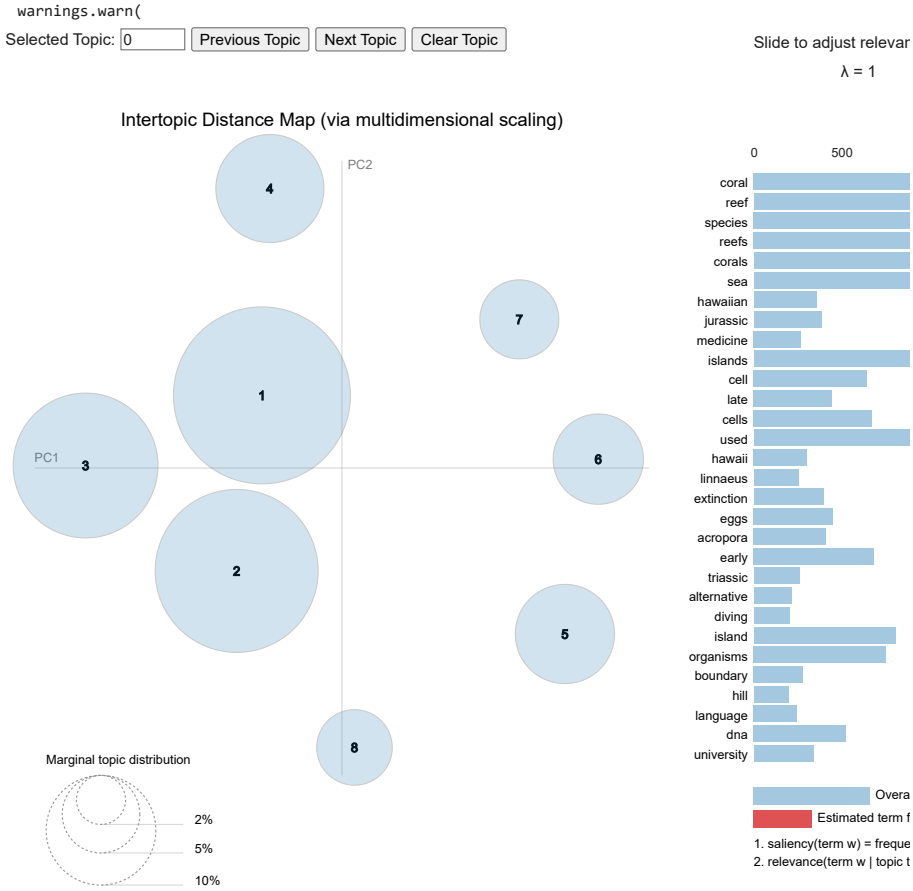
```
n,w =10, 10
count=CountVectorizer(lowercase=True,stop_words='english', token_pattern=r'[A-Za-z][A-Za-z][A-Za-z]+')
docmatrix=count.fit_transform(cleaned_text)
lda=LatentDirichletAllocation(n_components=n,random_state=42)
lda.fit(docmatrix)
features=count.get_feature_names_out()
topics=[]
for index,topic in enumerate(lda.components_):
    words=[features[i] for i in topic.argsort()[:-w - 1:-1]]
    topics.append(words)
pyLDAvis.enable_notebook()
lda_prepare(lda, docmatrix, count, mds='mmds')
```

```
c:\Users\McKay Shields\anaconda3\lib\site-packages\sklearn\manifold\_mds.py:299: FutureWarning: The de
  warnings.warn(
```



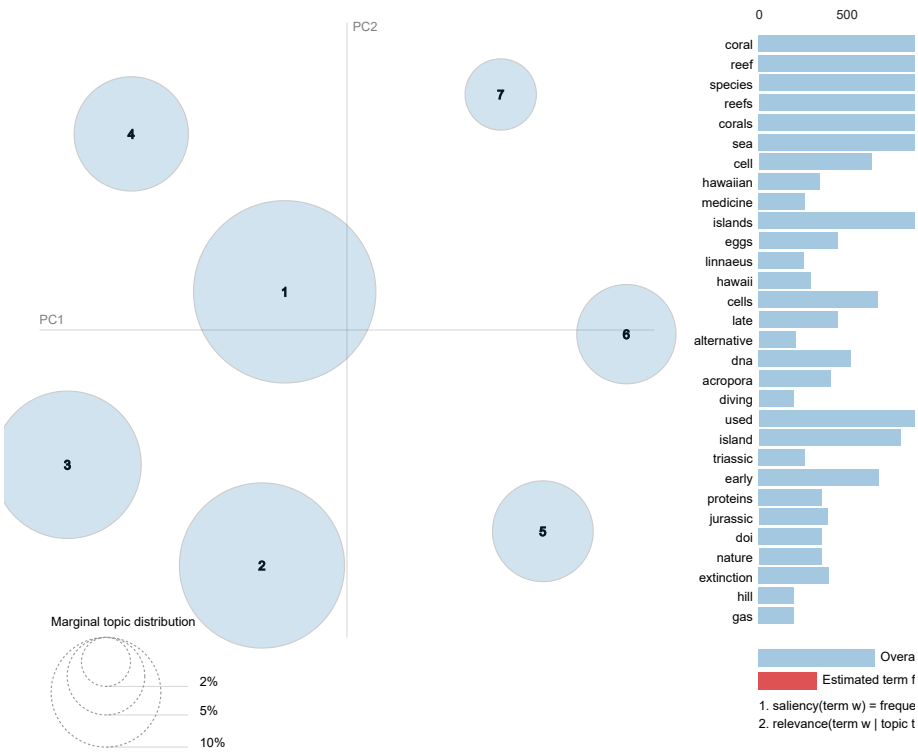

```
n,w =8, 10
count=CountVectorizer(lowercase=True,stop_words='english', token_pattern=r'[A-Za-z][A-Za-z][A-Za-z]+')
docmatrix=count.fit_transform(cleaned_text)
lda=LatentDirichletAllocation(n_components=n,random_state=42)
lda.fit(docmatrix)
features=count.get_feature_names_out()
topics=[]
for index,topic in enumerate(lda.components_):
    words=[features[i] for i in topic.argsort()[:-w - 1:-1]]
    topics.append(words)
pyLDAvis.enable_notebook()
lda_prepare(lda, docmatrix, count, mds='mmds')
```

```
c:\Users\McKay Shields\anaconda3\lib\site-packages\sklearn\manifold\_mds.py:299: FutureWarning: The de
  warnings.warn(
```



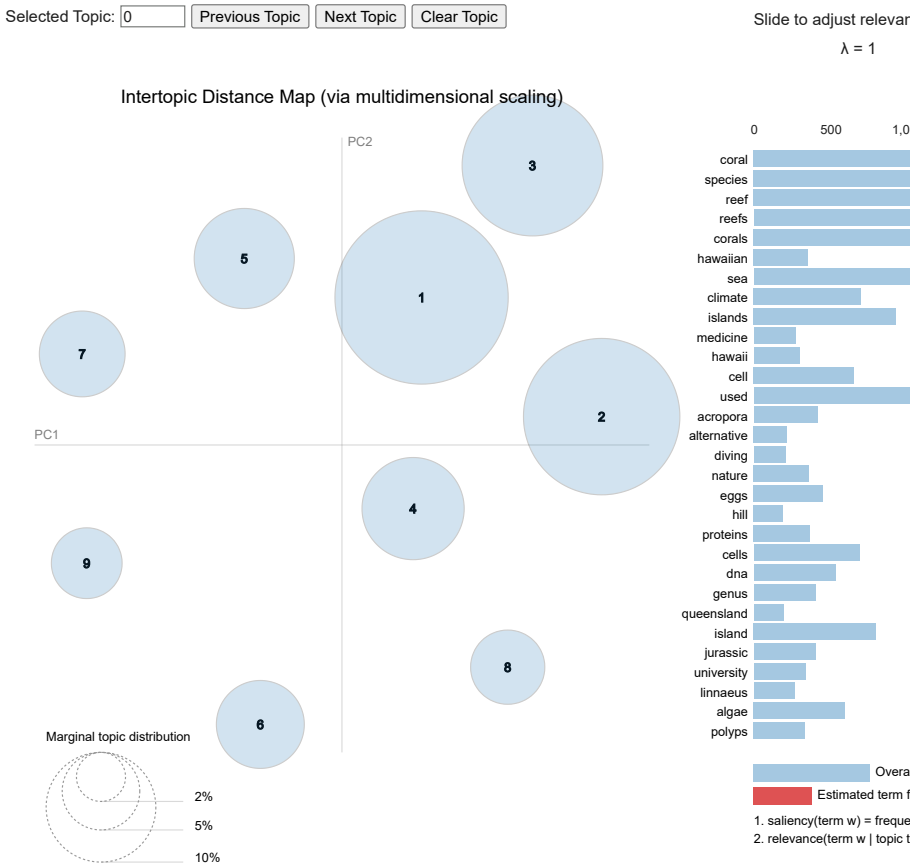

```
n,w =7, 10
count=CountVectorizer(lowercase=True,stop_words='english', token_pattern=r'[A-Za-z][A-Za-z][A-Za-z]+')
docmatrix=count.fit_transform(cleaned_text)
lda=LatentDirichletAllocation(n_components=n,random_state=42)
lda.fit(docmatrix)
features=count.get_feature_names_out()
topics=[]
for index,topic in enumerate(lda.components_):
    words=[features[i] for i in topic.argsort()[:-w - 1:-1]]
    topics.append(words)
pyLDAvis.enable_notebook()
lda_prepare(lda, docmatrix, count, mds='mmds')
```

```
c:\Users\McKay Shields\anaconda3\lib\site-packages\sklearn\manifold\_mds.py:299: FutureWarning: The de
  warnings.warn(
```



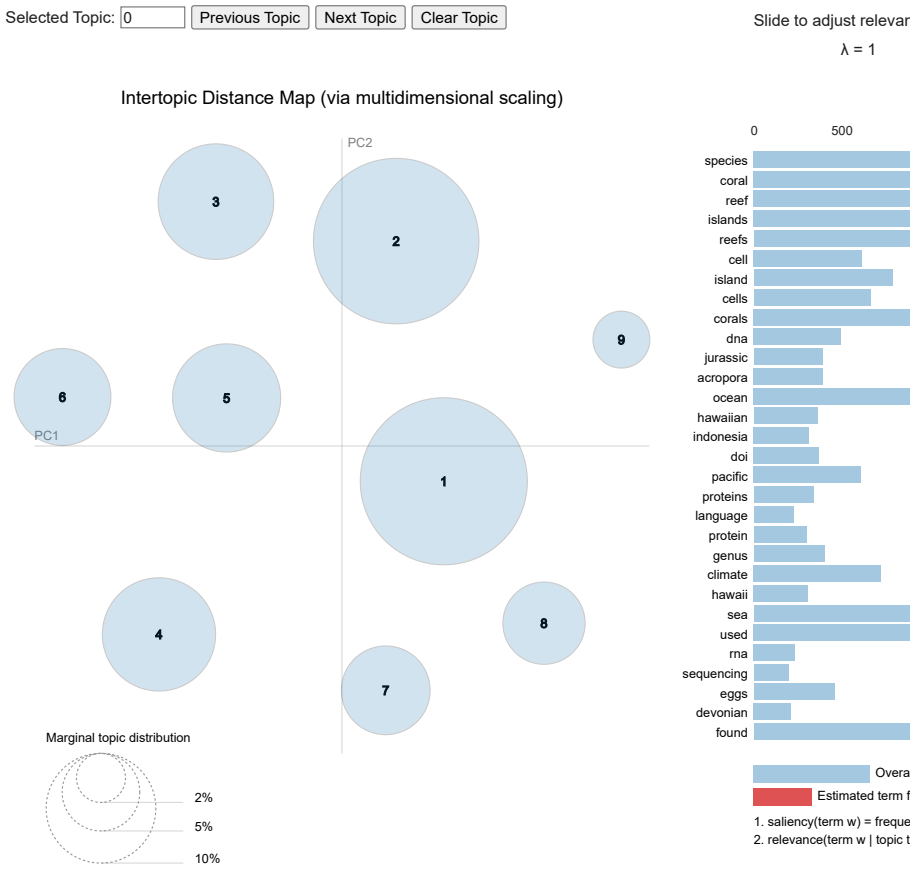

```
n,w =9, 10
count=CountVectorizer(lowercase=True,stop_words='english', token_pattern=r'[A-Za-z][A-Za-z][A-Za-z]+')
docmatrix=count.fit_transform(cleaned_text)
lda=LatentDirichletAllocation(n_components=n,random_state=42)
lda.fit(docmatrix)
features=count.get_feature_names_out()
topics=[]
for index,topic in enumerate(lda.components_):
    words=[features[i] for i in topic.argsort()[:-w - 1:-1]]
    topics.append(words)
pyLDAvis.enable_notebook()
lda_prepare(lda, docmatrix, count, mds='mmds')
```

```
c:\Users\McKay Shields\anaconda3\lib\site-packages\sklearn\manifold\_mds.py:299: FutureWarning: The de
  warnings.warn(
```




```
n,w =9, 10
count=CountVectorizer(lowercase=True, token_pattern=r'[A-Za-z][A-Za-z][A-Za-z]+')
docmatrix=count.fit_transform(cleaned_text)
lda=LatentDirichletAllocation(n_components=n,random_state=42)
lda.fit(docmatrix)
features=count.get_feature_names_out()
topics=[]
for index,topic in enumerate(lda.components_):
    words=[features[i] for i in topic.argsort()[:-w - 1:-1]]
    topics.append(words)
pyLDAvis.enable_notebook()
lda_prepare(lda, docmatrix, count, mds='mmds')
```

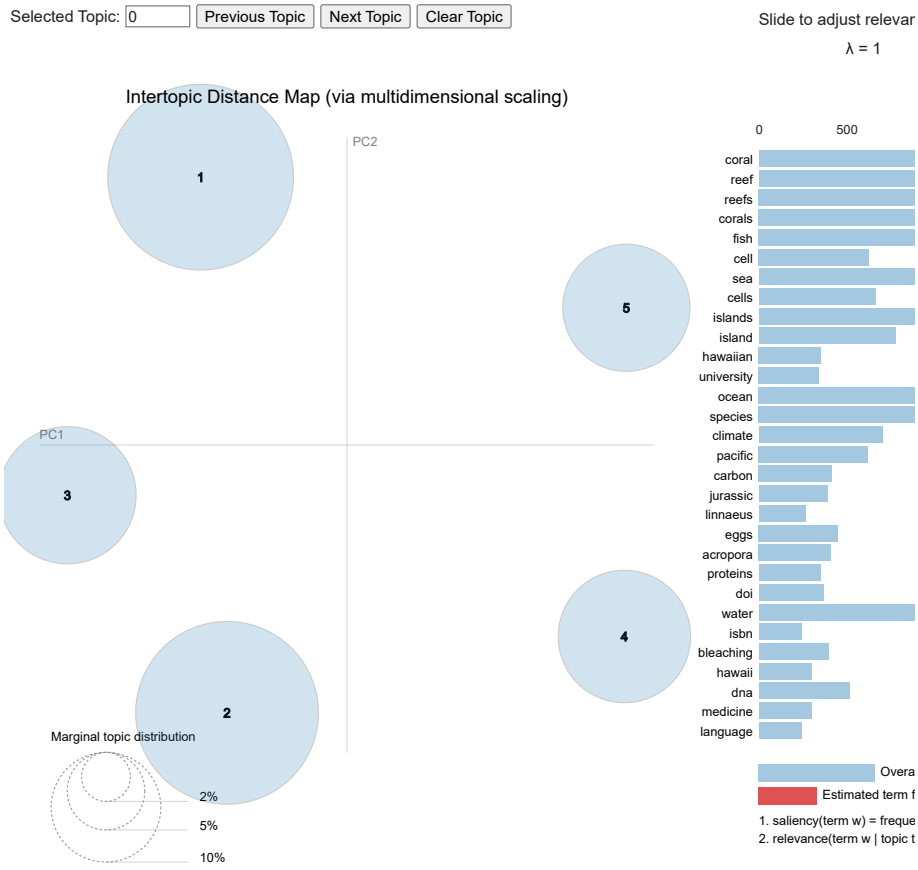
```
c:\Users\McKay Shields\anaconda3\lib\site-packages\sklearn\manifold\_mds.py:299: FutureWarning: The de
  warnings.warn(
```




```
cleaned_text2=[]
tokenizer = RegexpTokenizer(r"\w+")
# create English stop words list
englishfile = open("englishvocabulary.txt", encoding="utf8")
engstop = englishfile.read()
engstop = set(stopwords.words("english"))
files=os.listdir('wiki_corpus')
for file in files:
    with open(f'wiki_corpus/{file}','r') as f:
        lines=f.readlines()
    for line in lines:
        raw=line.lower()
        tokens=tokenizer.tokenize(raw)
        tokens = [i for i in tokens if not i in engstop and i.isalpha()]
        newfile=' '.join(tokens)
        cleaned_text2.append(newfile)
```

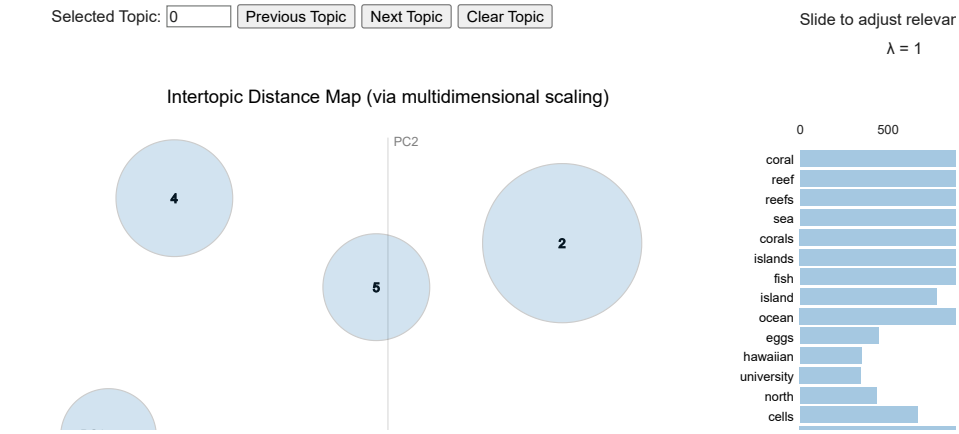
```
n,w =5, 10
count=CountVectorizer(lowercase=True,stop_words='english', token_pattern=r'[A-Za-z][A-Za-z][A-Za-z]+')
docmatrix=count.fit_transform(cleaned_text2)
lda=LatentDirichletAllocation(n_components=n,random_state=42)
lda.fit(docmatrix)
features=count.get_feature_names_out()
topics=[]
for index,topic in enumerate(lda.components_):
    words=[features[i] for i in topic.argsort()[:-w - 1:-1]]
    topics.append(words)
pyLDAvis.enable_notebook()
lda_prepare(lda, docmatrix, count, mds='mmds')
```

```
c:\Users\McKay Shields\anaconda3\lib\site-packages\sklearn\manifold\_mds.py:299: FutureWarning: The de
  warnings.warn(
```




```
n,w =7, 10
count=CountVectorizer(lowercase=True,stop_words='english', token_pattern=r'[A-Za-z][A-Za-z][A-Za-z]+')
docmatrix=count.fit_transform(cleaned_text2)
lda=LatentDirichletAllocation(n_components=n,random_state=42)
lda.fit(docmatrix)
features=count.get_feature_names_out()
topics=[]
for index,topic in enumerate(lda.components_):
    words=[features[i] for i in topic.argsort()[:-w - 1:-1]]
    topics.append(words)
pyLDAvis.enable_notebook()
lda_prepare(lda, docmatrix, count, mds='mmds')
```

```
c:\Users\McKay Shields\anaconda3\lib\site-packages\sklearn\manifold\_mds.py:299: FutureWarning: The de
  warnings.warn(
```




```
n,w =10, 10
count=CountVectorizer(lowercase=True,stop_words='english', token_pattern=r'[A-Za-z][A-Za-z][A-Za-z]+')
docmatrix=count.fit_transform(cleaned_text2)
lda=LatentDirichletAllocation(n_components=n,random_state=42)
lda.fit(docmatrix)
features=count.get_feature_names_out()
topics=[]
for index,topic in enumerate(lda.components_):
    words=[features[i] for i in topic.argsort()[:-w - 1:-1]]
    topics.append(words)
pyLDAvis.enable_notebook()
lda_prepare(lda, docmatrix, count, mds='mmds')
```

```
c:\Users\McKay Shields\anaconda3\lib\site-packages\sklearn\manifold\_mds.py:299: FutureWarning: The de
  warnings.warn(
```