Lab 8: Topic Modeling

By Sawit Koseeyaumporn

65070507238

In this Lab, we demonstrate to students how to apply topic modeling to real-world data. Students will gain hands-on experience through this example.

- Download the datasets

Supplementary: Topic Modeling

Objectives:

- To demonstrate students how to apply topic modeling to real-world data.
- Students will gain hands-on experience through this example.

Create a data directory and store the downloaded data (state-of-the-union.csv) in it. The data is about State of the Union addresses from 1970 to 2012.

```
[1] !mkdir -p data
    !wget -nc https://nyc3.digitaloceanspaces.com/ml-files-distro/v1/text-analysis/data/state-of-the-union.csv -P data
```

```
--2024-11-05 14:13:28--  https://nyc3.digitaloceanspaces.com/ml-files-distro/v1/text-analysis/data/state-of-the-union.csv
Resolving nyc3.digitaloceanspaces.com (nyc3.digitaloceanspaces.com)... 162.243.189.2
Connecting to nyc3.digitaloceanspaces.com (nyc3.digitaloceanspaces.com)|162.243.189.2|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 10501219 (10M) [text/csv]
Saving to: 'data/state-of-the-union.csv'

state-of-the-union. 100%[===================>]  10.01M  5.06MB/s    in 2.0s

2024-11-05 14:13:31 (5.06 MB/s) - 'data/state-of-the-union.csv' saved [10501219/10501219]
```

- Read the dataset and convert it dataFrame shape

Read data

```
import pandas as pd

df = pd.read_csv("data/state-of-the-union.csv")

# Clean it up a little bit, removing non-word characters (numbers and ___ etc)
df.content = df.content.str.replace("[^A-Za-z ]", " ")

df.head()
```

	year	content
0	1790	George Washington\nJanuary 8, 1790\n\nFellow-C...
1	1790	\nState of the Union Address\nGeorge Washingto...
2	1791	\nState of the Union Address\nGeorge Washingto...
3	1792	\nState of the Union Address\nGeorge Washingto...
4	1793	\nState of the Union Address\nGeorge Washingto...

```
[3] df.shape
```

```
(226, 2)
```

- Uses Gensim Library to convert a document into a list of tokens.

Using Gensim to perform topic modeling

```
# Run this cell if gensim has not been installed yet.
!pip install gensim
```

Apply `simple_process` to convert a document into a list of tokens. The input will be lowercased, tokenized, and de-accented (optional).

```
[4] from gensim.utils import simple_preprocess

    texts = df.content.apply(simple_preprocess)
```

```
[5] texts
```

	content
0	[george, washington, january, fellow, citizens...
1	[state, of, the, union, address, george, washi...
2	[state, of, the, union, address, george, washi...
3	[state, of, the, union, address, george, washi...
4	[state, of, the, union, address, george, washi...
...	...
221	[state, of, the, union, address, george, bush,...
222	[address, to, joint, session, of, congress, ba...
223	[state, of, the, union, address, barack, obama...
224	[state, of, the, union, address, barack, obama...
225	[state, of, the, union, address, barack, obama...

226 rows × 1 columns

dtype: object

Task 1: ID-to-word mapping

Task 1 : ID-to-word mapping:

In the current notebook, after calling the doc2bow method, all words are represented by their IDs. Consequently, when you use the print_topics method, only these IDs are displayed, making the output challenging to interpret and less meaningful. Therefore, Task #1 is to incorporate an ID-to-word mapping to resolve this issue.

Create a dictionary, using the texts that have already been preprocessed.

The method `doc2bow` is for converting document (a list of words) into the bag-of-words format.

```
from gensim import corpora

dictionary = corpora.Dictionary(texts)
dictionary.filter_extremes(no_below=5, no_above=0.5, keep_n=2000)

# สร้าง Bag-of-word เป็นรูปแบบ Corpus

corpus = [dictionary.doc2bow(text) for text in texts]
```

```
[12] print(corpus[:10])
```

```
[[(0, 1), (1, 1), (2, 1), (3, 1), (4, 1), (5, 1), (6, 1), (7, 1), (8, 1), (9, 1), (10, 2), (11, 1), (12, 1), (13, 1), (14, 1), (15, 1),
```

- Trying the different n_topics from the LDA models and found out what is the words in each topic.

Finding the topics with the LDA Models and with the different number of max_topics

```
from gensim import models

max_topics = 10

for n_topics in range(1, max_topics + 1):
    lda_model = models.LdaModel(corpus=corpus, num_topics=n_topics, id2word=dictionary)
    print(f"Topics for {n_topics} topics:")
    topics = lda_model.print_topics()
    for topic in topics:
        print(topic)
    print("\n")
```

```
Topics for 6 topics:
(0, '0.005*"mexico" + 0.003*"americans" + 0.003*"help" + 0.003*"per" + 0.002*"budget" + 0.002*"jobs" + 0.002*"court" + 0.002*"gold" + 0.002*"million" + 0.002*"tariff"')
(1, '0.004*"mexico" + 0.003*"program" + 0.003*"americans" + 0.003*"help" + 0.003*"minister" + 0.003*"today" + 0.003*"spain" + 0.002*"convention" + 0.002*"million" + 0.002*"intercourse"')
(2, '0.004*"mexico" + 0.003*"americans" + 0.003*"convention" + 0.002*"spain" + 0.002*"banks" + 0.002*"budget" + 0.002*"court" + 0.002*"bill" + 0.002*"per" + 0.002*"program"')
(3, '0.004*"program" + 0.004*"help" + 0.003*"americans" + 0.002*"million" + 0.002*"mexico" + 0.002*"budget" + 0.002*"tariff" + 0.002*"convention" + 0.002*"billion" + 0.002*"per"')
(4, '0.003*"help" + 0.003*"mexico" + 0.002*"per" + 0.002*"americans" + 0.002*"today" + 0.002*"convention" + 0.002*"programs" + 0.002*"estimated" + 0.002*"cannot" + 0.002*"tariff"')
(5, '0.004*"help" + 0.004*"program" + 0.003*"million" + 0.003*"programs" + 0.003*"billion" + 0.003*"americans" + 0.003*"budget" + 0.003*"per" + 0.003*"problems" + 0.002*"tonight"')


WARNING:gensim.models.ldamodel:too few updates, training might not converge; consider increasing the number of passes or iterations to improve accuracy
Topics for 7 topics:
(0, '0.003*"mexico" + 0.003*"help" + 0.003*"americans" + 0.003*"budget" + 0.002*"billion" + 0.002*"program" + 0.002*"islands" + 0.002*"today" + 0.002*"intercourse" + 0.002*"programs"')
(1, '0.003*"per" + 0.003*"americans" + 0.003*"program" + 0.002*"help" + 0.002*"minister" + 0.002*"mexico" + 0.002*"court" + 0.002*"cent" + 0.002*"intercourse" + 0.002*"ships"')
(2, '0.004*"mexico" + 0.003*"help" + 0.003*"gold" + 0.003*"per" + 0.003*"americans" + 0.002*"minister" + 0.002*"convention" + 0.002*"bank" + 0.002*"banks" + 0.002*"notes"')
(3, '0.003*"spain" + 0.003*"estimated" + 0.003*"help" + 0.003*"convention" + 0.002*"program" + 0.002*"mexico" + 0.002*"americans" + 0.002*"budget" + 0.002*"per" + 0.002*"gold"')
(4, '0.004*"mexico" + 0.003*"program" + 0.003*"help" + 0.003*"jobs" + 0.003*"million" + 0.003*"americans" + 0.003*"convention" + 0.002*"today" + 0.002*"billion" + 0.002*"tonight"')
(5, '0.004*"million" + 0.004*"help" + 0.003*"program" + 0.003*"americans" + 0.003*"budget" + 0.003*"mexico" + 0.002*"per" + 0.002*"tonight" + 0.002*"programs" + 0.002*"problems"')
(6, '0.004*"program" + 0.004*"americans" + 0.003*"help" + 0.003*"mexico" + 0.002*"million" + 0.002*"tonight" + 0.002*"budget" + 0.002*"per" + 0.002*"banks" + 0.002*"bank"')


WARNING:gensim.models.ldamodel:too few updates, training might not converge; consider increasing the number of passes or iterations to improve accuracy
Topics for 8 topics:
(0, '0.005*"mexico" + 0.004*"americans" + 0.003*"help" + 0.003*"budget" + 0.003*"programs" + 0.003*"million" + 0.002*"program" + 0.002*"jobs" + 0.002*"today" + 0.002*"minister"')
(1, '0.003*"program" + 0.003*"help" + 0.003*"mexico" + 0.003*"americans" + 0.003*"budget" + 0.003*"million" + 0.002*"convention" + 0.002*"gold" + 0.002*"billion" + 0.002*"court"')
(2, '0.003*"mexico" + 0.003*"americans" + 0.003*"program" + 0.003*"million" + 0.003*"help" + 0.002*"budget" + 0.002*"banks" + 0.002*"billion" + 0.002*"programs" + 0.002*"working"')
(3, '0.003*"mexico" + 0.003*"help" + 0.003*"program" + 0.003*"americans" + 0.002*"per" + 0.002*"tariff" + 0.002*"programs" + 0.002*"tonight" + 0.002*"convention" + 0.002*"cent"')
(4, '0.003*"mexico" + 0.003*"americans" + 0.002*"program" + 0.002*"convention" + 0.002*"spain" + 0.002*"bank" + 0.002*"silver" + 0.002*"payment" + 0.002*"bill" + 0.002*"per"')
(5, '0.004*"mexico" + 0.004*"program" + 0.003*"per" + 0.002*"spain" + 0.002*"minister" + 0.002*"indians" + 0.002*"convention" + 0.002*"americans" + 0.002*"tariff" + 0.002*"help"')
(6, '0.003*"million" + 0.003*"program" + 0.003*"court" + 0.003*"americans" + 0.003*"help" + 0.002*"today" + 0.002*"problems" + 0.002*"per" + 0.002*"billion" + 0.002*"mexico"')
(7, '0.005*"help" + 0.003*"americans" + 0.003*"per" + 0.003*"mexico" + 0.002*"tonight" + 0.002*"budget" + 0.002*"reform" + 0.002*"convention" + 0.002*"jobs" + 0.002*"get"')
```

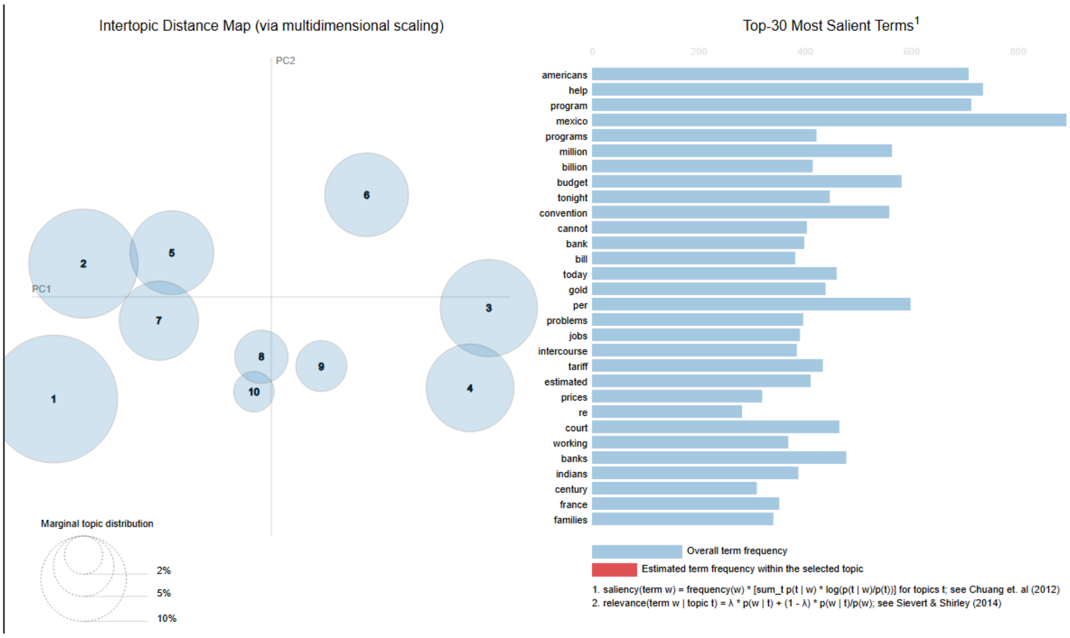
- Installing the pyLDAvis for visualization

```
[14] # Run this cell if pyLDAvis has never been installed
!pip install pyLDAvis
```

```
Collecting pyLDAvis
  Downloading pyLDAvis-3.4.1-py3-none-any.whl.metadata (4.2 kB)
Requirement already satisfied: numpy>=1.24.2 in /usr/local/lib/python3.10/dist-packages (from pyLDAvis) (1.26.4)
Requirement already satisfied: scipy in /usr/local/lib/python3.10/dist-packages (from pyLDAvis) (1.13.1)
Requirement already satisfied: pandas>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from pyLDAvis) (2.2.2)
Requirement already satisfied: joblib>=1.2.0 in /usr/local/lib/python3.10/dist-packages (from pyLDAvis) (1.4.2)
Requirement already satisfied: jinja2 in /usr/local/lib/python3.10/dist-packages (from pyLDAvis) (3.1.4)
Requirement already satisfied: numexpr in /usr/local/lib/python3.10/dist-packages (from pyLDAvis) (2.10.1)
Collecting funcy (from pyLDAvis)
  Downloading funcy-2.0-py2.py3-none-any.whl.metadata (5.9 kB)
Requirement already satisfied: scikit-learn>=1.0.0 in /usr/local/lib/python3.10/dist-packages (from pyLDAvis) (1.5.2)
Requirement already satisfied: gensim in /usr/local/lib/python3.10/dist-packages (from pyLDAvis) (4.3.3)
Requirement already satisfied: setuptools in /usr/local/lib/python3.10/dist-packages (from pyLDAvis) (75.1.0)
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.10/dist-packages (from pandas>=2.0.0->pyLDAvis) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas>=2.0.0->pyLDAvis) (2024.2)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.10/dist-packages (from pandas>=2.0.0->pyLDAvis) (2024.2)
Requirement already satisfied: threadpoolctl>=3.1.0 in /usr/local/lib/python3.10/dist-packages (from scikit-learn>=1.0.0->pyLDAvis) (3.5.0)
Requirement already satisfied: smart-open>=1.8.1 in /usr/local/lib/python3.10/dist-packages (from gensim->pyLDAvis) (7.0.5)
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.10/dist-packages (from jinja2->pyLDAvis) (3.0.2)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.8.2->pandas>=2.0.0->pyLDAvis) (1.16.0)
Requirement already satisfied: wrapt in /usr/local/lib/python3.10/dist-packages (from smart-open>=1.8.1->gensim->pyLDAvis) (1.16.0)
Downloading pyLDAvis-3.4.1-py3-none-any.whl (2.6 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 2.6/2.6 MB 23.2 MB/s eta 0:00:00
Downloading funcy-2.0-py2.py3-none-any.whl (30 kB)
Installing collected packages: funcy, pyLDAvis
Successfully installed funcy-2.0 pyLDAvis-3.4.1
```

- Using lda_model, corpus and dictionary to create the visualization

```
import pyLDAvis
import pyLDAvis.gensim

pyLDAvis.enable_notebook()
vis = pyLDAvis.gensim.prepare(lda_model, corpus, dictionary)
vis
```



Task 2: Try tuning the LDA parameters or incorporate additional text preprocessing

- First, we will add additional text preprocessing

```python
import nltk
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer
import string
import re

nltk.download('stopwords')
nltk.download('wordnet')

lemmatizer = WordNetLemmatizer()


def clean_punctuation(text):
    # Define a regular expression pattern to match punctuation and special characters
    # Matches any character that is not a word character (\w), space (\s), or underscore (_)
    punctuation_pattern = re.compile(r'[^\w\s]|_')

    # Replace punctuation and special characters with an empty string
    cleaned_text = re.sub(punctuation_pattern, '', text)

    return cleaned_text

def clean_text(text):
    # Convert to lowercase
    text = text.lower()
    # Lemmatizing
    text = " ".join([lemmatizer.lemmatize(word) for word in text.split()])
    # Remove punctuation
    text = clean_punctuation(text)
    # Remove numbers
    text = re.sub(r'\d+', '', text)
    # Remove extra whitespaces
    text = re.sub(r'\s+', ' ', text).strip()
    # Remove stopwords
    stop_words = set(stopwords.words("english"))
    text = " ".join([word for word in text.split() if word not in stop_words])
    return text
```

```
df.content = df.content.str.replace("[^A-Za-z ]", " ")
df.content = df.content.apply(clean_text)

df.head()
```

	year	content
0	1790	george washington january fellowcitizens senat...
1	1790	state union address george washington december...
2	1791	state union address george washington october ...
3	1792	state union address george washington november...
4	1793	state union address george washington december...

- Create the iteration to find the LDA parameter

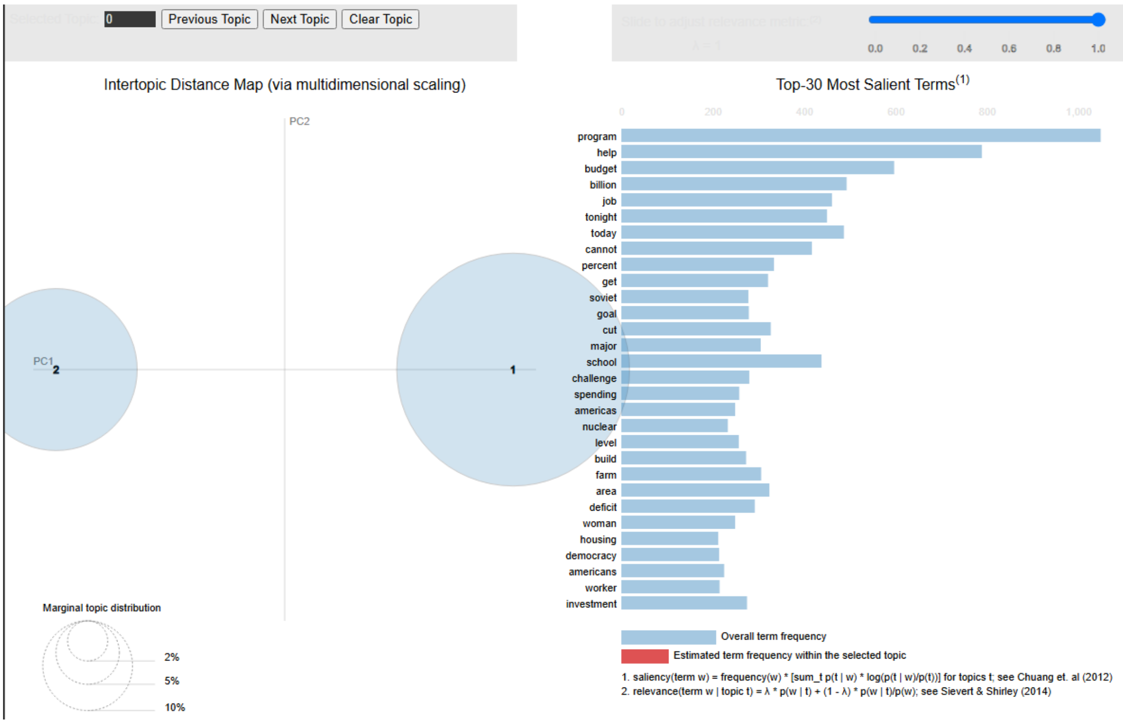
```
from gensim import models
from gensim.corpora import Dictionary
from gensim.utils import simple_preprocess
import pandas as pd

# ... (your previous code for data loading and cleaning) ...

# Assuming 'cleaned_content' column contains the cleaned text
texts = df['content'].apply(simple_preprocess)
dictionary = Dictionary(texts)
dictionary.filter_extremes(no_below=5, no_above=0.5, keep_n=2000)
corpus = [dictionary.doc2bow(text) for text in texts]

# Define the parameter grid
param_grid = {
    'num_topics': [2, 5, 10, 15],  # Number of topics
    'passes': [10]
}

# Initialize variables to store the best model and its coherence score
best_lda_model = None
best_coherence_score = -1
```

```python
# Iterate through the parameter grid (excluding alpha and eta)
for num_topics in param_grid['num_topics']:
    for passes in param_grid['passes']:
        # Train the LDA model
        lda_model = models.LdaModel(
            corpus=corpus,
            id2word=dictionary,
            num_topics=num_topics,
            passes=passes,
            random_state=42
        )

        # ... (rest of your code for coherence calculation and model selection) ...

                # Calculate coherence score
        coherence_model_lda = models.CoherenceModel(
            model=lda_model, texts=texts, dictionary=dictionary, coherence='c_v'
        )
        coherence_score = coherence_model_lda.get_coherence()

        # Update best model if coherence score is improved
        if coherence_score > best_coherence_score:
            best_coherence_score = coherence_score
            best_lda_model = lda_model

# Print the best model and its coherence score
print(f"Best LDA Model: {best_lda_model}")
print(f"Best Coherence Score: {best_coherence_score}")
```

```
/usr/local/lib/python3.10/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarni
  and should_run_async(code)
Best LDA Model: LdaModel<num_terms=2000, num_topics=2, decay=0.5, chunksize=2000>
Best Coherence Score: 0.5241149774801065
```

สรุปว่า Num_topics=2 และ num_terms = 2000 เหมาะสมที่สุดสำหรับ Data นี้

- Visualize again



Sawit Koseeyaumporn 65070507238