Info

Name: Jacqui Unciano **Date:** March 27, 2024 **Assignment:** HW09

In [1]:
```
import pandas as pd
import numpy as np
```

In [2]:
```
import configparser
config = configparser.ConfigParser()
config.read("../../../env.ini")
data_home = config['DEFAULT']['data_home']
output_dir = config['DEFAULT']['output_dir']
local_lib = config['DEFAULT']['local_lib']
```

In [3]:
```
# Adjust this for a new corpus
data_prefix = 'austen-melville'
table_dir =  f'{data_home}/{data_prefix}'
OHCO = ['book_id', 'chap_id', 'para_num', 'sent_num', 'token_num']
CHAP = OHCO[:3]
PARA = OHCO[:4] # Paragraphs
SENT = OHCO[:5] # Sentences
```

In this week's code exercise, you will create a notebook that creates two word embedding models —one for the novels of Jane Austen and one for those of Melville — using our demonstration corpus from previous lessons (austen-melville-TOKENS.csv and austen-melville-LIB.csv). Using the word2vec notebook (M09_04_word2vec.ipynb) from class as your guide, do the following:

In [4]:
```
TOKENS = pd.read_csv(f'{output_dir}/{data_prefix}-CORPUS.csv').set_index(OHCO)
LIB = pd.read_csv(f'{output_dir}/{data_prefix}-LIB.csv').set_index(OHCO[:1])
```

In [5]:
```
TOKENS.head()
```

Out[5]:

book_id	chap_id	para_num	sent_num	token_num	pos_tuple	pos	token_str	term_str
105	1	1	0	0	('Sir', 'NNP')	NNP	Sir	sir
				1	('Walter', 'NNP')	NNP	Walter	walter
				2	('Elliot,', 'NNP')	NNP	Elliot,	elliot
				3	('of', 'IN')	IN	of	of
				4	('Kellynch', 'NNP')	NNP	Kellynch	kellynch

In [6]:
```
LIB.head()
```

Out[6]:

book_id	source_file_path	author	title	chap_regex
105	/Users/jacqu/OneDrive/Documents/MSDS-at-UVA-20...	AUSTEN, JANE	PERSUASION	^Chapter\s+\d+$
121	/Users/jacqu/OneDrive/Documents/MSDS-at-UVA-20...	AUSTEN, JANE	NORTHANGER ABBEY	^CHAPTER\s+\d+$
141	/Users/jacqu/OneDrive/Documents/MSDS-at-UVA-20...	AUSTEN, JANE	MANSFIELD PARK	^CHAPTER\s+[IVXLCM]+$
158	/Users/jacqu/OneDrive/Documents/MSDS-at-UVA-20...	AUSTEN, JANE	EMMA	^\s*CHAPTER\s+[IVXLCM]+\s*$
161	/Users/jacqu/OneDrive/Documents/MSDS-at-UVA-20...	AUSTEN, JANE	SENSE AND SENSIBILITY	^CHAPTER\s+\d+$

In [7]:
```
ab_dict = {key: group.index.tolist() for key, group in LIB.groupby('author')}
authors = []
for id in TOKENS.index.get_level_values(level=0):
    if id in ab_dict["AUSTEN, JANE"]:
        authors.append("AUSTEN, JANE")
    else:
        authors.append("MELVILLE, HERMAN")

TOKENS['author'] = authors
#TOKENS.drop(columns=['author'], inplace=True)
TOKENS = TOKENS.reset_index().set_index(['author']+OHCO)
austen_TOKENS = TOKENS.query('author=="AUSTEN, JANE"')
melville_TOKENS = TOKENS.query('author=="MELVILLE, HERMAN"')
```

```
aTOKENS = austen_TOKENS.reset_index().drop(columns=['author', 'token_str', 'pos_tup
mTOKENS = melville_TOKENS.reset_index().drop(columns=['author', 'token_str', 'pos_t
```

In [8]:
```
aTOKENS.head()
```

Out[8]:

					pos	term_str	pos_group
book_id	**chap_id**	**para_num**	**sent_num**	**token_num**			
105	**1**	**1**	**0**	**0**	NNP	sir	NN
				1	NNP	walter	NN
				2	NNP	elliot	NN
				3	IN	of	IN
				4	NNP	kellynch	NN

In [9]:
```
mTOKENS.head()
```

Out[9]:

					pos	term_str	pos_group
book_id	**chap_id**	**para_num**	**sent_num**	**token_num**			
1900	**1**	**0**	**0**	**0**	DT	the	DT
				1	NNP	sea	NN
				2	NNP	longings	NN
				3	NNP	for	NN
				4	NNP	shore	NN

Convert the TOKEN table into two Gensim corpora, one for each author.

1. Make sure you define a proper OHCO when importing your token data.
2. Filter the TOKEN table to include only nouns and verbs, excluding proper nouns.

For each corpus, extract a vocabulary table with the following features:

1. n: the raw count of the tokens for the term in the corpus
2. pos_group: the grouped maximum part-of-speech of the term (i.e. first get maximum POS, then group by first two characters)
3. Don't worry about computing DFIDF or anything else.

For each corpus, generate a table of word vectors using Gensim's word2vec with the following parameters:

1. window = 2
2. for Austen: min_count = 50
3. for Melville: min_count = 80

4. vector_size = 256

For each model, generate a table of tSNE coordinates using SciKit Learn's TSNE with the following paramters:

1. learning_rate = 200.
2. perplexity = 20
3. n_components = 2
4. init = 'random'
5. n_iter = 1000
6. random_state = 42 or any number, if you want your results to be reproducible in your notebook.

For convenience, join this table with your VOCAB table so you can use the latter's features when visualizing.

Visualize the tSNE coordinates using Plotly Express's scatter with the following paramters:

1. color = 'pos_group'
2. If you want larger data points, consider creating the log of VOCAB.n as a feature and using that as size in the visualization.

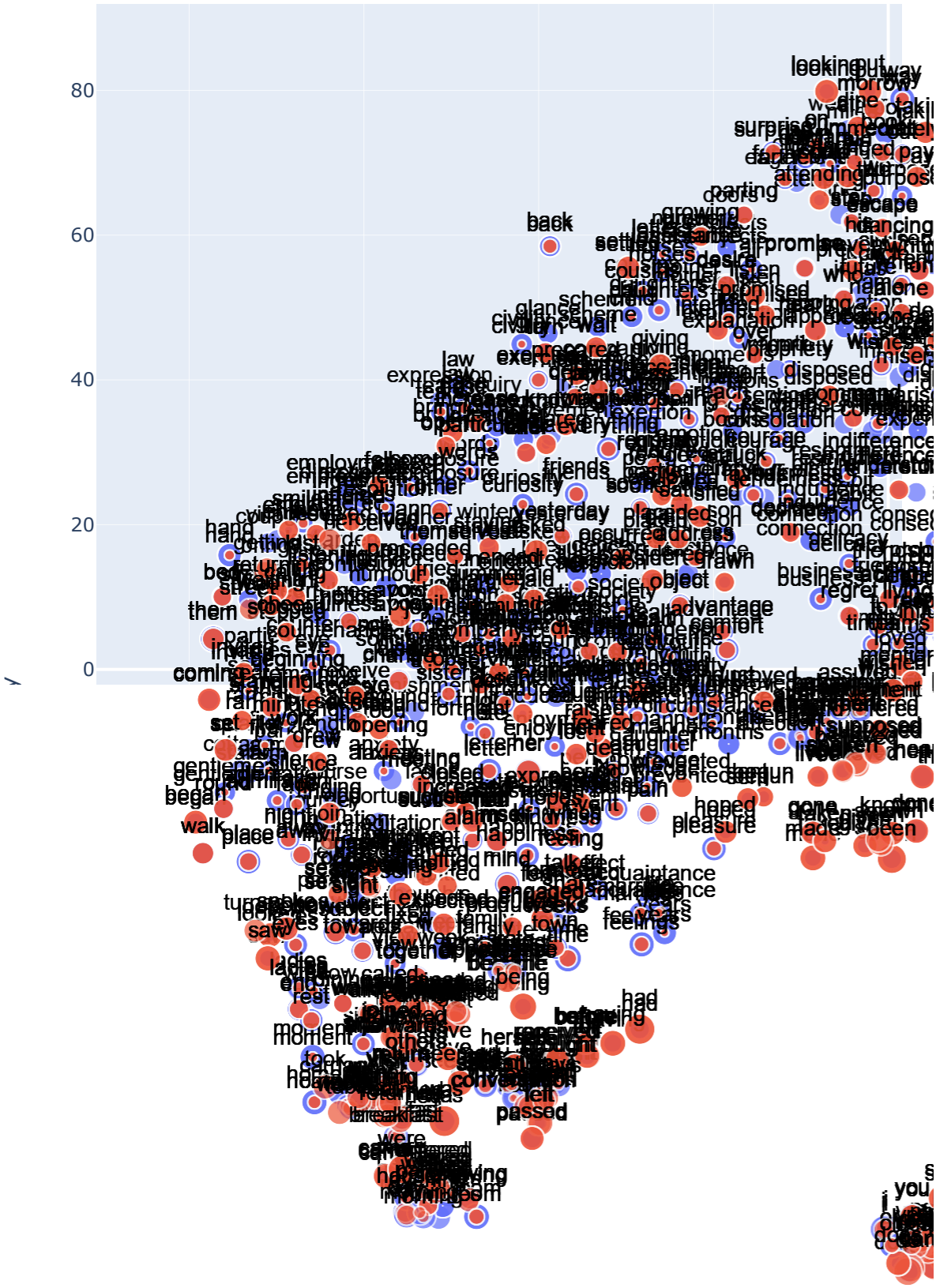
In [10]:
```
import pandas as pd
import numpy as np
import gensim
from gensim.models import word2vec
from gensim.corpora import Dictionary
from sklearn.manifold import TSNE
import plotly_express as px
import nltk
from nltk import pos_tag
from nltk.tokenize import word_tokenize
nltk.download('punkt')

class WordEmbed:
    TOKENS = pd.DataFrame()
    docs = []
    vocab = pd.DataFrame()
    wv_table = pd.DataFrame()
    tsne_table = pd.DataFrame()

    def __init__(self, TOKENS):
        self.TOKENS = TOKENS

    def filter_pos(self, include_pos = [], exclude_pos = []):
        self.TOKENS = self.TOKENS[self.TOKENS['pos'].str[0].isin(include_pos) & ~se

    def gensim_corpus(self, bag):
        docs = self.TOKENS.dropna(subset='term_str')\
        .groupby(bag)\
        .term_str.apply(lambda  x:  x.tolist())\
```

```
        .reset_index()['term_str'].tolist()

        self.docs = [doc for doc in docs if len(doc) > 1] # Lose single word docs
        return self.docs

    def get_vocab(self):
        self.vocab = self.TOKENS.groupby(['pos_group']).term_str.value_counts().to_
        self.vocab['log_n'] = np.log10(self.vocab.n)
        return self.vocab

    def vec_engine(self, w2v_params = dict(window=5, vector_size=200, min_count=10)
        self.model = word2vec.Word2Vec(self.docs, **w2v_params)

    def get_vector(self, row):
        w = row.name
        try:
            vec = self.model.wv[w]
        except KeyError as e:
            vec = None
        return vec

    def wv_table(self):
        self.wv_table = pd.DataFrame(self.vocab.apply(self.get_vector, axis=1).drop
        .apply(lambda x: pd.Series(x[0]), axis=1)
        return self.wv_table

    def get_tsne(self, tsne_params = dict(perplexity=40, n_components=2, init='pca'
        tsne_engine = TSNE(**tsne_params)
        tsne_model = tsne_engine.fit_transform(self.wv_table.to_numpy())
        self.tsne_table = pd.DataFrame(tsne_model, columns=['x','y'], index=self.wv
        return self.tsne_table

    def vis_tsne(self):
        X = self.tsne_table.join(self.vocab, how='left')
        my_plot = px.scatter(X.reset_index(), 'x', 'y',
                   text='term_str',
                   color='pos_group',
                   hover_name='term_str',
                   size='log_n',
                   height=1000).update_traces(
                        mode='markers+text',
                        textfont=dict(color='black', size=14, family='Arial'),
                        textposition='top center')
        return my_plot
```

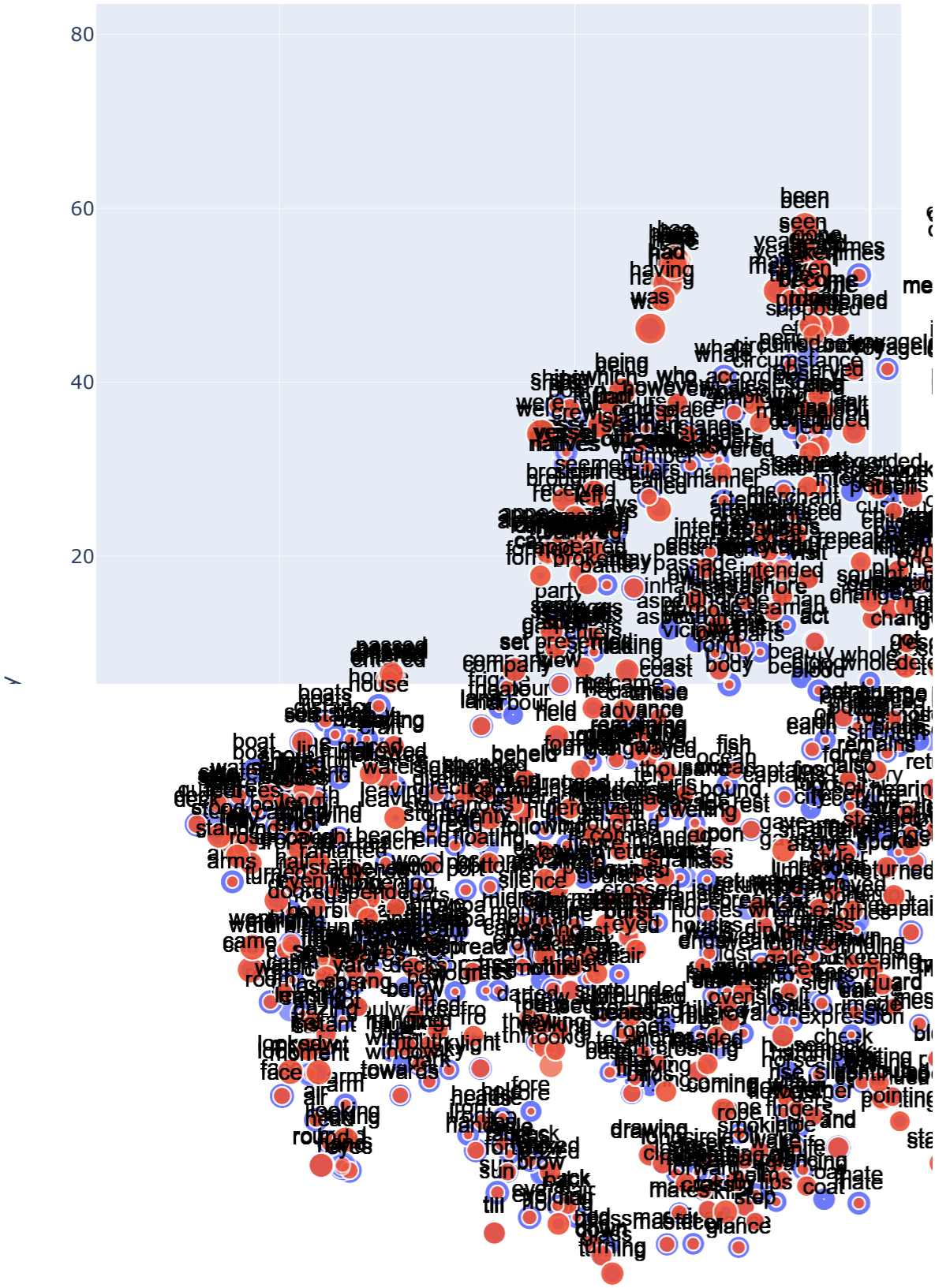
```
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\jacqu\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!
```

In [11]:
```
# Filter the TOKEN table to include only nouns and verbs, excluding proper nouns.
include_pos = ['N', 'V']
exclude_pos = ['NNP', 'NNPS']
BAG = CHAP
# word2vec parameters
w2v_params = dict(
    window = 2,
```

```
    vector_size = 256,
    min_count = 50, # THIS LIMITS OUR VOCAB
)
tsne_params = dict(
    learning_rate = 200.,
    perplexity=20,
    n_components=2,
    init='random',
    n_iter=1000,
    random_state=42
)

aWE = WordEmbed(aTOKENS)
aWE.filter_pos(include_pos = include_pos, exclude_pos = exclude_pos)
acorpus = aWE.gensim_corpus(bag=BAG)
avocab = aWE.get_vocab()
aWE.vec_engine(w2v_params = w2v_params)
awv_table = aWE.wv_table()
atsne_table = aWE.get_tsne(tsne_params = tsne_params)
aWE.vis_tsne()
```


80
60
40
20
0
y

In [12]:
```
# Filter the TOKEN table to include only nouns and verbs, excluding proper nouns.
include_pos = ['N', 'V']
exclude_pos = ['NNP', 'NNPS']
BAG = CHAP
# word2vec parameters
w2v_params = dict(
    window = 2,
    vector_size = 256,
    min_count = 80, # THIS LIMITS OUR VOCAB
)
tsne_params = dict(
    learning_rate = 200.,
    perplexity=20,
    n_components=2,
    init='random',
    n_iter=1000,
    random_state=42
)

mWE = WordEmbed(mTOKENS)
mWE.filter_pos(include_pos = include_pos, exclude_pos = exclude_pos)
mcorpus = mWE.gensim_corpus(bag=BAG)
mvocab = mWE.get_vocab()
mWE.vec_engine(w2v_params = w2v_params)
mwv_table = mWE.wv_table()
mtsne_table = mWE.get_tsne(tsne_params = tsne_params)
mWE.vis_tsne()
```


80
60
40
20

Question 1

Identify two regions of word clusters in the Austen plot that clearly contain words with associated meanings. Give a gloss of what you think these clusters "mean."

Answer 1 For one region, I have found 'solicitude', 'possibility', 'relief', 'suffering'; words that I associate with heavy emotions or relationships. Another cluster has "consequence," "fear," "money," "mistaken," and "dislike". I'm thinking of the chaotic misunderstandings that revolve around the complexities of the relationships in Austen's novels.

Question 2

Identify two regions of word clusters in the Melville plot that clearly contain words with associated meanings. Give a gloss of what you think these clusters "mean."

Answer 2 In one region, there are words like 'sailor', 'whaling', 'power', 'knowledge', 'nature' in a cluster. I think this might mean something about adventure or seeking adventure. In another region, I see 'food', 'tobacco', 'steward', 'master' in a cluster. I'm leaning towards something around sea-life with food and tobacco being common commodities on ships around that time period and master and steward being roles.

Question 3

Based on your inspection of the results, come up with two analogies from the Austen model using 'man' and 'woman' as the A and C terms of the analogy. Describe your results.

Answer 3 Man is to father as woman is to daughter OR man is to gentlemen as woman is to lady. In Austen's novels, the father and daughter, gentlemen and lady combos are really prelevent. Her novels are about navigating relationships in the regency(?) era- both familial and romantic relationships.

Question 4

Do the same thing with Melville.

Answer 4 Man is to seamen as woman is to ship. In the context that both are ocean related but the people on the ship are referred to using masculine nouns/diminutives and ships are referred to using feminine pronouns/diminutives. Another one is man is to captain as woman is to navigator. This one is a bit of a reach, but thinking about the time period Melville is in, the captain is someone in charge of leading or giving direction while a

navigator is in charge of safety and relaying important information, similar to how the partiarch and the martriarch of a family in that time period typically work.

Question 5

Consider the information that topic models provide in comparison to what word embeddings provide. How would you describe their differences? Might they complementary to each other?

Answer 5 For topic models, the basic purpose is to , aim tfindnt topics within a corpus by identifying groups of words that frequently co-occur in documentsThese topics represent themes that capture the main ideas present in the text. Whereas word embeddings turn words into vectors based on their distributional properties in a corpus. And these vectors capture semantic relationships between words, enabling operations like similarity measurement, analogy completion, and language understanding.

tI would use topic modeling ford in text mining, information retrieval, and content analyin order sis to uncover hidden patterns, trends, and themes in large text datas For wcation: Word e, I would use it for such as sentiment gnition, machine translatsometimes ion, and information in order to capture thet require understanding of word semantics and relati

I think they can work together really well.ent Analysis: Topic models can help uncover the main themes and topics in a docume, and thenection, while word embeddings n in-depthfiner-grainedthe analysis of semantic relationships between individual words within ematic overview.