```
!pip install pyLDAvis==3.4.1
!pip install --upgrade ipykernel
```

```
import nltk
import re
import os
import warnings
from gensim import corpora
from gensim.models.ldamodel import LdaModel
from gensim.models import CoherenceModel
import spacy
import pyLDAvis
import pyLDAvis.gensim
import matplotlib.pyplot as plt
import warnings
```

```
!python -m spacy download sv_core_news_sm
```

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

```
nltk.download('stopwords')
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
True
```

Let's create a simple example corpus

```
# coffee
doc_1 = "The aroma of freshly brewed coffee tantalized my senses as I entered the cozy cafe on a brisk morning. \
        With each sip of rich, dark espresso, I felt a surge of energy coursing through my veins, \
        awakening my mind and spirit. As the steam rose from my mug, I savored the robust flavor of \
        the carefully roasted beans, \
        relishing in the simple pleasure of a perfect cup of coffee."

# tea
doc_2 = "The aroma of freshly steeped tea leaves filled the air, creating a sense of tranquility in the serene tea room. \
        With each delicate sip of fragrant jasmine tea, I felt a soothing warmth spreading through my body, \
        calming my senses. As I held the porcelain teacup in my hands, I savored the gentle infusion of flavors, \
        embracing the peaceful moment offered by a perfect cup of tea."

# sleep
doc_3 = "As night falls, the world fades away into a realm of dreams where the mind finds solace in the \
        embrace of sleep. In the depths of slumber, the body enters a state of restorative rest, \
        replenishing its energy for the challenges of the coming day. \
        Each peaceful breath in the quiet of the night serves as a reminder of the profound importance \
        of a good night's sleep for overall well-being."

# morning routines
doc_4 = "Morning routines set the tone for the day, offering a structured start to ensure productivity and balance. \
        From the invigorating aroma of freshly brewed coffee to the calming ritual of meditation, \
        each activity contributes to a harmonious beginning. Whether it's a brisk jog in the crisp \
        morning air or a moment of reflection amidst the chaos, morning routines provide a sacred space \
        to align mind, body, and spirit for the day ahead."
```

```
documents = [doc_1, doc_2, doc_3, doc_4]
```

Step 1: Preprocessing

```
def to_lowercase(document):
    document = document.lower()
    return document
```

```
def remove_special_characters_english(document):
    document = re.sub(r'[^a-zA-Z\s]', '', document)
    return document
```

```
def remove_special_characters_swedish(document):
    document = re.sub(r'[^a-zA-ZåäöÅÄÖ\s]', '', document)
    return document
```

```
def tokenize(document):
    document = document.split()
    return document


stopwords_english = nltk.corpus.stopwords.words('english')


stopwords_swedish = nltk.corpus.stopwords.words('swedish')


def remove_stopwords_english(document):
    document = [token for token in document if token not in stopwords_english]
    return document


def remove_stopwords_swedish(document):
    document = [token for token in document if token not in stopwords_swedish]
    return document


def remove_letters(document):
    document = [token for token in document if len(token) > 1]
    return document


def lemmatize_english(tokenized_document):
    nlp = spacy.load("en_core_web_sm", disable=['ner', 'parser', 'textcat'])
    doc = nlp(" ".join(tokenized_document))
    document = [token.lemma_ for token in doc]
    return document


def lemmatize_swedish(tokenized_document):
    nlp = spacy.load("sv_core_news_sm", disable=['ner', 'parser', 'textcat'])
    doc = nlp(" ".join(tokenized_document))
    document = [token.lemma_ for token in doc]
    return document
```

Preprocessing example:

```
test_text = "This is 3!? somE ranDom.    Texts?!?! Please P helping me..."


test_text = to_lowercase(test_text)
test_text
```

```
'this is 3!? some random.    texts?!?! please p helping me...'
```

```
test_text = remove_special_characters_english(test_text)
test_text
```

```
'this is  some random    texts please p helping me'
```

```
test_text = tokenize(test_text)
test_text
```

```
['this', 'is', 'some', 'random', 'texts', 'please', 'p', 'helping', 'me']
```

```
test_text = remove_stopwords_english(test_text)
test_text
```

```
['random', 'texts', 'please', 'p', 'helping']
```

```
test_text = remove_letters(test_text)
test_text
```

```
['random', 'texts', 'please', 'helping']
```

```
test_text = lemmatize_english(test_text)
test_text
```

```
['random', 'text', 'please', 'help']
```

```
def preprocess_english(documents):
    preprocessed_documents = []

    for document in documents:
        document = to_lowercase(document)
        document = remove_special_characters_english(document)
        document = tokenize(document)
        document = remove_stopwords_english(document)
        document = lemmatize_english(document)
        document = remove_letters(document)
        preprocessed_documents.append(document)
    return preprocessed_documents


def preprocess_swedish(documents):
    preprocessed_documents = []

    for document in documents:
        document = to_lowercase(document)
        document = remove_special_characters_english(document)
        document = tokenize(document)
        document = remove_stopwords_swedish(document)
        document = lemmatize_swedish(document)
        document = remove_letters(document)
        preprocessed_documents.append(document)
    return preprocessed_documents


documents
```

```
['The aroma of freshly brewed coffee tantalized my senses as I entered the cozy cafe on a brisk morning.         With each sip of rich, dark espresso, I felt a surge of energy coursing through my veins,         awakening my mind and spirit. As the steam rose from my mug, I savored the robust flavor of         the carefully roasted beans, relishing in the simple pleasure of a perfect cup of coffee.',
 'The aroma of freshly steeped tea leaves filled the air, creating a sense of tranquility in the serene tea room. With each delicate sip of fragrant jasmine tea, I felt a soothing warmth spreading through my body,         calming my senses. As I held the porcelain teacup in my hands, I savored the gentle infusion of flavors,         embracing the peaceful moment offered by a perfect cup of tea.',
 "As night falls, the world fades away into a realm of dreams where the mind finds solace in the         embrace of sleep. In the depths of slumber, the body enters a state of restorative rest,         replenishing its energy for the challenges of the coming day.         Each peaceful breath in the quiet of the night serves as a reminder of the profound importance         of a good night's sleep for overall well-being.",
 "Morning routines set the tone for the day, offering a structured start to ensure productivity and balance. From the invigorating aroma of freshly brewed coffee to the calming ritual of meditation,         each activity contributes to a harmonious beginning. Whether it's a brisk jog in the crisp         morning air or a moment of reflection amidst the chaos, morning routines provide a sacred space         to align mind, body, and spirit for the day ahead."]
```

```
preprocessed_documents = preprocess_english(documents)
preprocessed_documents
```

```
 'begin',
'whether',
'brisk',
'jog',
'crisp',
'morning',
'air',
'moment',
'reflection',
'amidst',
'chaos',
'morning',
'routine',
'provide',
'sacred',
'space',
'align',
'mind',
'body',
'spirit',
'day',
'ahead']]
```

Gensim Topicmodelling with LDA

```
id2word = corpora.Dictionary(preprocessed_documents)
corpus = [id2word.doc2bow(text) for text in preprocessed_documents]
```

```
corpus
# mapping (word_id, word_frequency)
# (7, 2) implies that word with id 7 occurs 2 times in document 1
```

```
        (115, 1),
        (116, 1),
        (117, 1),
        (118, 1)]]
```

```
id2word[7]
```

```
'coffee'
```

Instantiate LDA Model

```
lda_model = LdaModel(corpus, num_topics=4, id2word=id2word, passes=15, random_state=1)
```

Print the resulting topics

```
topics = lda_model.print_topics(num_words=3)
for topic in topics:
    print(topic)
```

```
(0, '0.049*"night" + 0.034*"sleep" + 0.019*"body"')
(1, '0.061*"tea" + 0.032*"sense" + 0.018*"freshly"')
(2, '0.033*"coffee" + 0.018*"enter" + 0.018*"energy"')
(3, '0.044*"morning" + 0.031*"day" + 0.030*"routine"')
```

Print the topics for each document

```
for id, doc in enumerate(corpus):
    doc_topics = lda_model.get_document_topics(doc)

    # Sort topics by probability in descending order
    doc_topics = sorted(doc_topics, key=lambda x: x[1], reverse=True)

    # Print the document id and the dominant topics
    print(f"Document {id + 1}:")
    for topic, prob in doc_topics:
        print(f"Topic {topic}: Probability {prob}, \n {topics[topic][1]}")
        print("\n")
```

```
Document 1:
Topic 2: Probability 0.9805244207382202,
 0.033*"coffee" + 0.018*"enter" + 0.018*"energy"


Document 2:
Topic 1: Probability 0.9815220236778259,
 0.061*"tea" + 0.032*"sense" + 0.018*"freshly"


Document 3:
Topic 0: Probability 0.980141818523407,
 0.049*"night" + 0.034*"sleep" + 0.019*"body"


Document 4:
Topic 3: Probability 0.9831551313400269,
 0.044*"morning" + 0.031*"day" + 0.030*"routine"
```

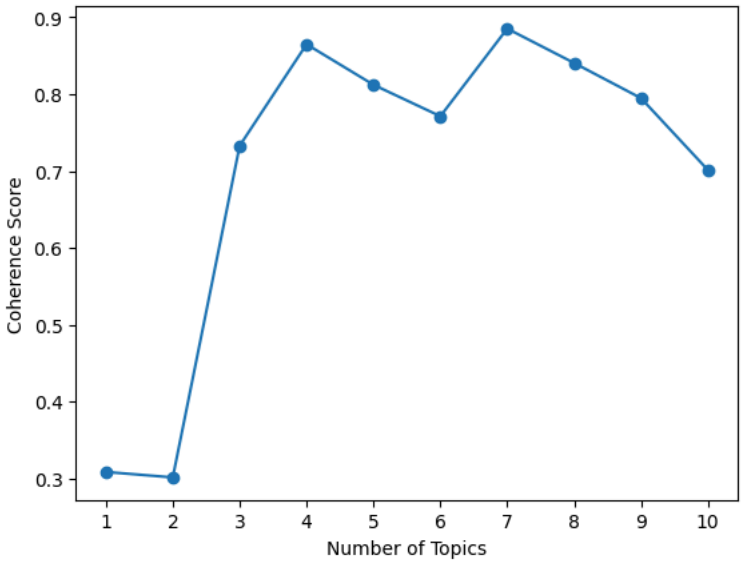
How can we find the optimal number of topics?

- We use the coherence score!
- We want to find a model that leads to a high coherence score
- A higher coherence score indicates that the words within the topic tend to co-occur together more frequently, topic is more coherent and interpretable
- Use the "Elbow Method" to determine optimal number of topics

```
def calculate_coherence_scores(corpus, id2word, preprocessed_documents, max_topics=10):
    coherence_scores = []
    for k in range(max_topics):
        lda_model = LdaModel(corpus, num_topics=1+k, id2word=id2word, passes=15, random_state=1)
        coherence_model_lda = CoherenceModel(model=lda_model, texts=preprocessed_documents, dictionary=id2word, coherence='
        coherence_score = coherence_model_lda.get_coherence()
        coherence_scores.append(coherence_score)
    return coherence_scores


coherence_scores = calculate_coherence_scores(corpus, id2word, preprocessed_documents, max_topics=10)
coherence_scores
```

```
[0.3082028799523205,
 0.3011128087185201,
 0.7329060846428858,
 0.8649793510627883,
 0.8122628278859892,
 0.7712635741077962,
 0.8853488946158085,
 0.8405937405784534,
 0.7948177640447704,
 0.7012926133475059]
```

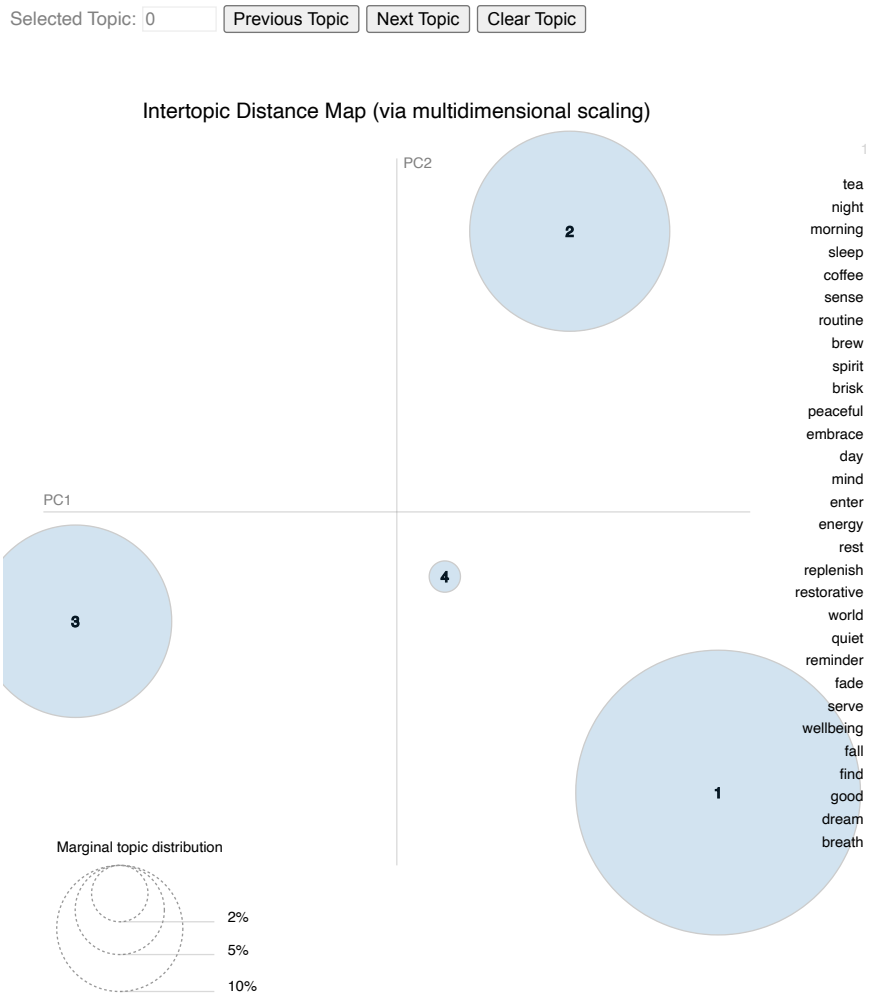
```
# plot coherence scores

num_topics = range(1, len(coherence_scores) + 1)
plt.plot(num_topics, coherence_scores, marker='o')

# Adding labels and title
plt.xlabel('Number of Topics')
plt.ylabel('Coherence Score')

plt.xticks(num_topics)

plt.show()
```



Visualize the Topics

```
lda_model = LdaModel(corpus, num_topics=4, id2word=id2word, passes=15)
```

```
pyLDAvis.enable_notebook()
vis = pyLDAvis.gensim.prepare(lda_model, corpus, id2word)
vis
```



- Each bubble represents a topic
- The larger the bubble, the higher percentage of the number of documents in the corpus is about that topic
- Blue bars represent the frequency of each word in the corpus
- Red bars give the estimated number of times a term was generated by a given topic
- The further the bubbles (topics) are away from each other, the more different they are

Your Turn!

Upload the data I sent you, or upload your own data (can be done on the left side under "Files")

```
folder_path = 'data/'

documents = []
for filename in os.listdir(folder_path):
    file_path = os.path.join(folder_path, filename)
    with open(file_path, 'r') as file:
        doc = file.read()
        documents.append(doc)
```

1) Preprocessing

```
preprocessed_documents = preprocess_english(documents)
```

2) LDA Model

```
id2word = corpora.Dictionary(preprocessed_documents)
corpus = [id2word.doc2bow(text) for text in preprocessed_documents]


lda_model = LdaModel(corpus, num_topics=5, id2word=id2word, passes=15, random_state=1)


topics = lda_model.print_topics(num_words=5)
for topic in topics:
    print(topic)
```

```
(0, '0.061*"language" + 0.014*"human" + 0.013*"use" + 0.010*"social" + 0.010*"system"')
(1, '0.017*"ai" + 0.013*"humanity" + 0.013*"microsoft" + 0.010*"language" + 0.010*"study"')
(2, '0.047*"digital" + 0.035*"humanity" + 0.013*"field" + 0.013*"study" + 0.010*"dh"')
(3, '0.031*"language" + 0.021*"linguistic" + 0.014*"study" + 0.011*"knowledge" + 0.010*"ai"')
(4, '0.061*"intelligence" + 0.020*"theory" + 0.015*"iq" + 0.013*"test" + 0.013*"individual"')
```

```
for id, doc in enumerate(corpus):
    doc_topics = lda_model.get_document_topics(doc)

    # Sort topics by probability in descending order
    doc_topics = sorted(doc_topics, key=lambda x: x[1], reverse=True)

    # Print the document id and the dominant topics
    print("\n")
    print(f"Document {id + 1}:")
    print("\n")
    for topic, prob in doc_topics:
        print(f"Topic {topic}: Probability {prob}, \n {topics[topic][1]}")
        #print("\n")
```

```
Document 1:


Topic 2: Probability 0.9981434345245361,
 0.047*"digital" + 0.035*"humanity" + 0.013*"field" + 0.013*"study" + 0.010*"dh"


Document 2:


Topic 3: Probability 0.9992031455039978,
 0.031*"language" + 0.021*"linguistic" + 0.014*"study" + 0.011*"knowledge" + 0.010*"ai"


Document 3:


Topic 0: Probability 0.9978558421134949,
 0.061*"language" + 0.014*"human" + 0.013*"use" + 0.010*"social" + 0.010*"system"


Document 4:


Topic 1: Probability 0.9972884654998779,
 0.017*"ai" + 0.013*"humanity" + 0.013*"microsoft" + 0.010*"language" + 0.010*"study"


Document 5:


Topic 1: Probability 0.9964922666549683,
 0.017*"ai" + 0.013*"humanity" + 0.013*"microsoft" + 0.010*"language" + 0.010*"study"


Document 6:


Topic 3: Probability 0.998895525932312,
 0.031*"language" + 0.021*"linguistic" + 0.014*"study" + 0.011*"knowledge" + 0.010*"ai"


Document 7:


Topic 4: Probability 0.9983711838722229,
 0.061*"intelligence" + 0.020*"theory" + 0.015*"iq" + 0.013*"test" + 0.013*"individual"
```

```
# Specify the number of top topics to display
num_top_topics = 3  # Adjust this value as needed

for id, doc in enumerate(corpus):
    doc_topics = lda_model.get_document_topics(doc)

    # Sort topics by probability in descending order
    doc_topics = sorted(doc_topics, key=lambda x: x[1], reverse=True)

    # Print the document id and the top topics
    print(f"Document {id + 1}:")
    for i, (topic, prob) in enumerate(doc_topics[:num_top_topics]):
        print(f"Topic {i + 1}: Topic {topic}, Probability {prob}")
        print(topics[topic][1])  # Print topic words or description
        print("\n")
```

```
Document 1:
Topic 1: Topic 2, Probability 0.9981434345245361
0.047*"digital" + 0.035*"humanity" + 0.013*"field" + 0.013*"study" + 0.010*"dh"


Document 2:
Topic 1: Topic 3, Probability 0.9992032647132874
0.031*"language" + 0.021*"linguistic" + 0.014*"study" + 0.011*"knowledge" + 0.010*"ai"


Document 3:
Topic 1: Topic 0, Probability 0.9978557825088501
0.061*"language" + 0.014*"human" + 0.013*"use" + 0.010*"social" + 0.010*"system"


Document 4:
Topic 1: Topic 1, Probability 0.9972884654998779
0.017*"ai" + 0.013*"humanity" + 0.013*"microsoft" + 0.010*"language" + 0.010*"study"


Document 5:
Topic 1: Topic 1, Probability 0.996491551399231
0.017*"ai" + 0.013*"humanity" + 0.013*"microsoft" + 0.010*"language" + 0.010*"study"


Document 6:
Topic 1: Topic 3, Probability 0.998895525932312
0.031*"language" + 0.021*"linguistic" + 0.014*"study" + 0.011*"knowledge" + 0.010*"ai"


Document 7:
Topic 1: Topic 4, Probability 0.9983711838722229
0.061*"intelligence" + 0.020*"theory" + 0.015*"iq" + 0.013*"test" + 0.013*"individual"
```

```
documents[0]
```

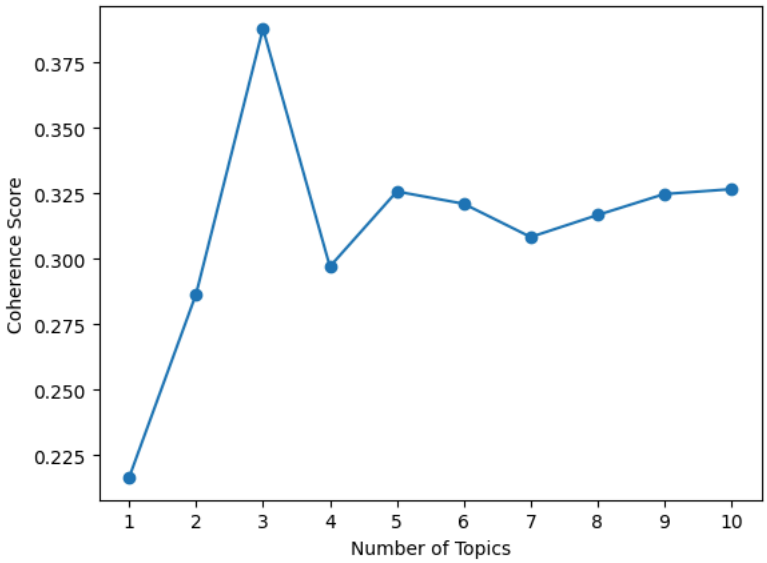
```
'Digital humanities (DH) is an area of scholarly activity at the intersectio
n of computing or digital technologies \nand the disciplines of the humaniti
es. It includes the systematic use of digital resources in the humanities,
\nas well as the analysis of their application.[1][2] DH can be defined as n
ew ways of doing scholarship that \ninvolve collaborative, transdisciplinar
y, and computationally engaged research, teaching, and publishing.[3] \nIt b
rings digital tools and methods to the study of the humanities with the reco
gnition that the printed word \nis no longer the main medium for knowledge p
roduction and distribution.[3]\nBy producing and using new applications and
```

3) Plot Coherence Scores

```
coherence_scores = calculate_coherence_scores(corpus, id2word, preprocessed_documents, max_topics=10)
coherence_scores
```

```
[0.21658607989415665,
 0.28657359762373663,
 0.38790070274201144,
 0.297075024110046,
 0.32573475676338015,
 0.3210381867077738,
 0.30834887787359555,
 0.316741830500208,
 0.3247695639275221,
 0.3266140789640716]
```

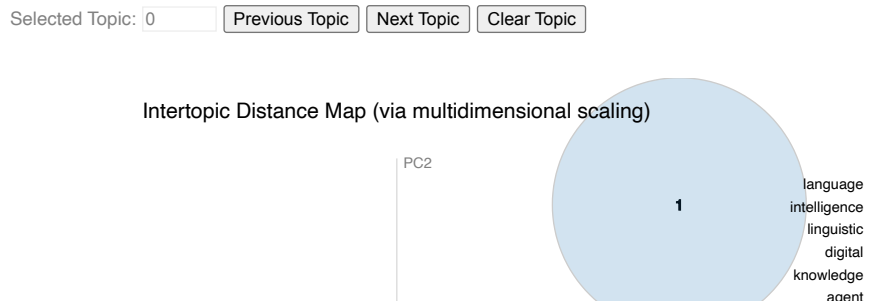
```
# plot coherence scores

num_topics = range(1, len(coherence_scores) + 1)
plt.plot(num_topics, coherence_scores, marker='o')

# Adding labels and title
plt.xlabel('Number of Topics')
plt.ylabel('Coherence Score')

plt.xticks(num_topics)

plt.show()
```



4) Visualize Topics

```
lda_model = LdaModel(corpus, num_topics=3, id2word=id2word, passes=15, random_state=1)
```

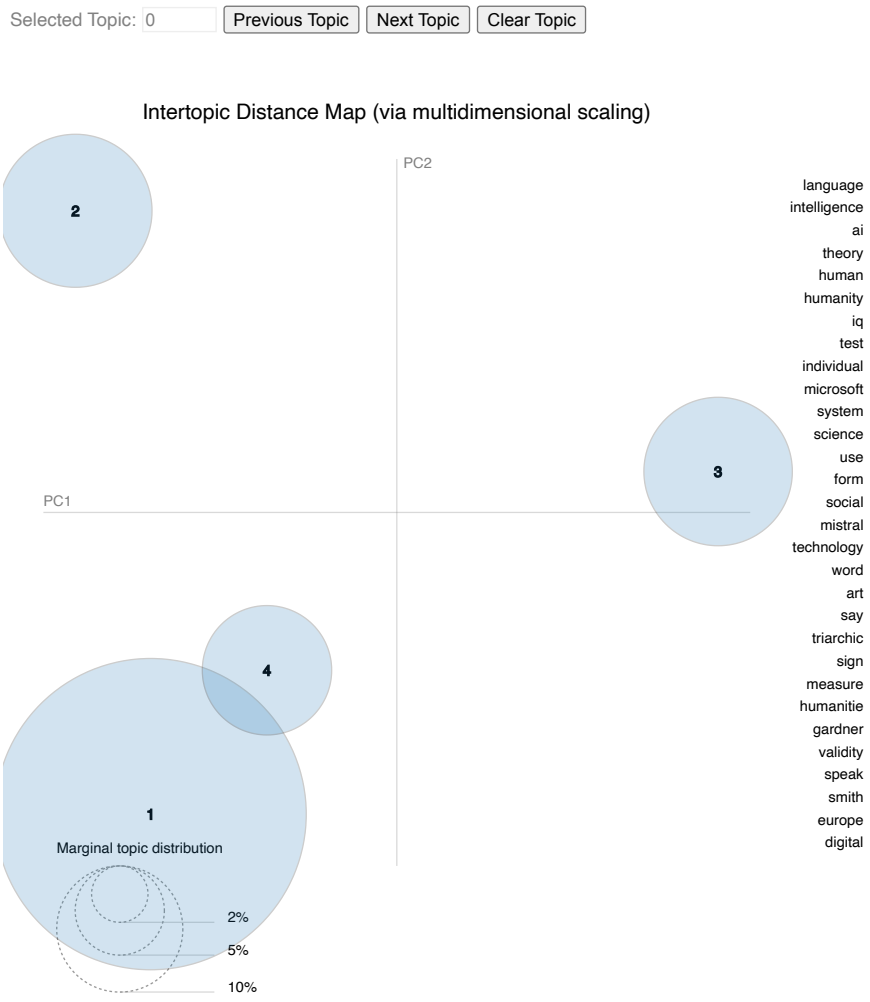
```
pyLDAvis.enable_notebook()
vis = pyLDAvis.gensim.prepare(lda_model, corpus, id2word)
vis
```



Let's see how the plot differs if we search for 4 topics

```
lda_model = LdaModel(corpus, num_topics=4, id2word=id2word, passes=15, random_state=1)
```

```
pyLDAvis.enable_notebook()
vis = pyLDAvis.gensim.prepare(lda_model, corpus, id2word)
vis
```



You can see that there is some overlap between topic 1 and 4 now! This means that the two topics are not perfectly separated

...Already finished? Download some additional texts from the internet and apply topic modelling on them.