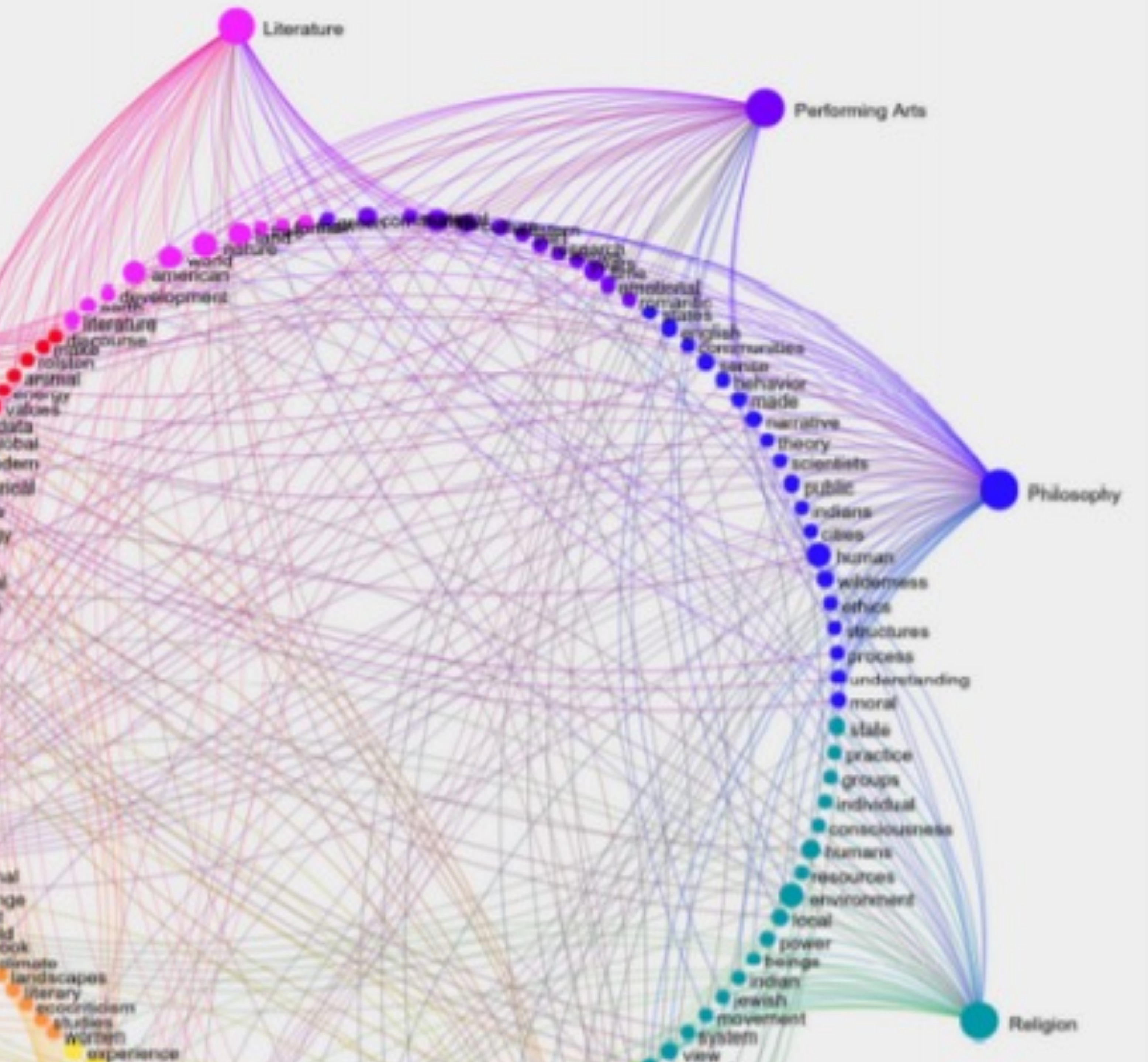


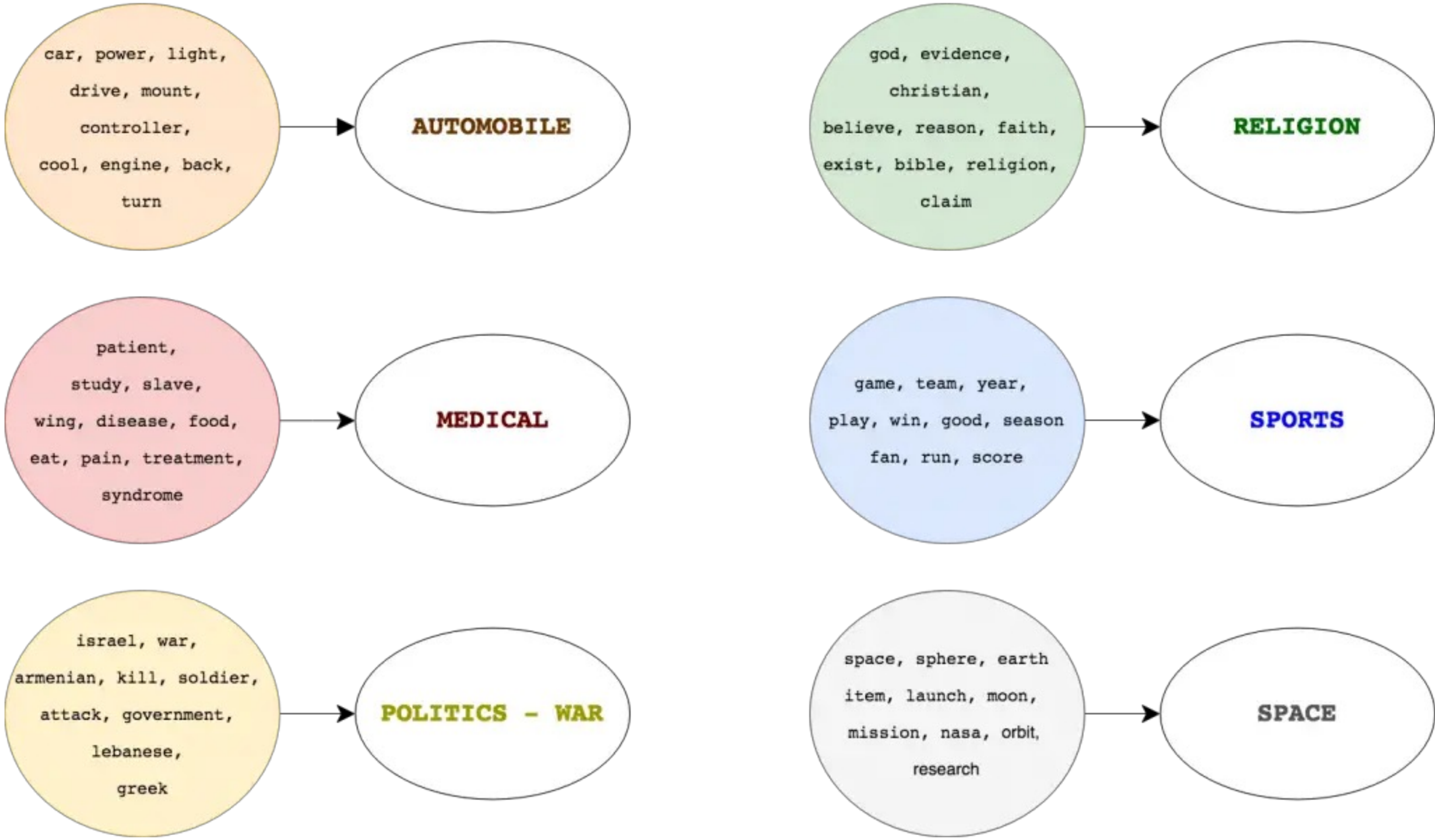
Topic 6
Topic Modeling

Unsupervise alg ct main
representing
topic

Topic Modeling

could be used for project

- Not the same as topic classification
- Using unsupervised learning to extract the main topics



https://www.machinelearningplus.com/nlp/topic-modeling-genism-python/#11createthedictionaryandcorpusneededfortopicmodeling

Industrial Use-cases

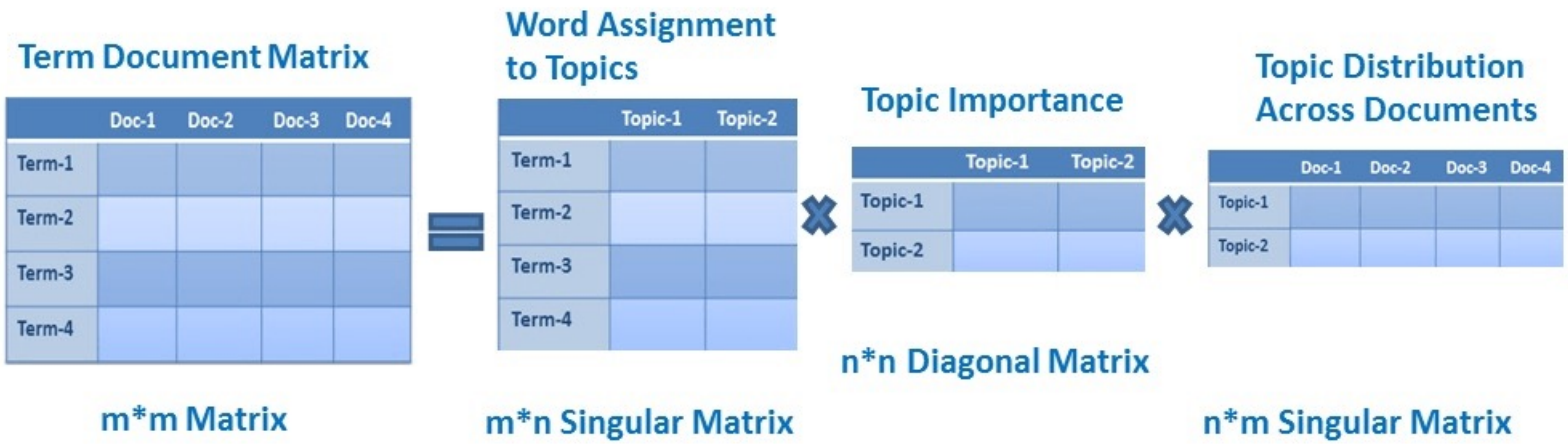
- Customer Service
  - Tagging customer support tickets
  - Routing conversations
  - Detecting the urgency
  - Act on customer feedback
  - Etc.

Latent Semantic Analysis (LSA)

- Assesses relationships between a set of documents and the terms it contains.
- Uses singular value decomposition (SVD)
- The word ‘latent’ = hidden topics in a document

Document term matrix

	W1	W2	W3	W4	W5	W6
D1	0	3	0	0	1	2
D2	1	0	0	1	1	1
D3	2	1	2	2	4	2
D4	1	1	1	4	0	0
D5	0	1	2	1	0	4




```
# Perform SVD on the document-term matrix
from sklearn.decomposition import TruncatedSVD
svd = TruncatedSVD(n_components=2)
doc_topic_matrix = svd.fit_transform(doc_term_matrix)
```

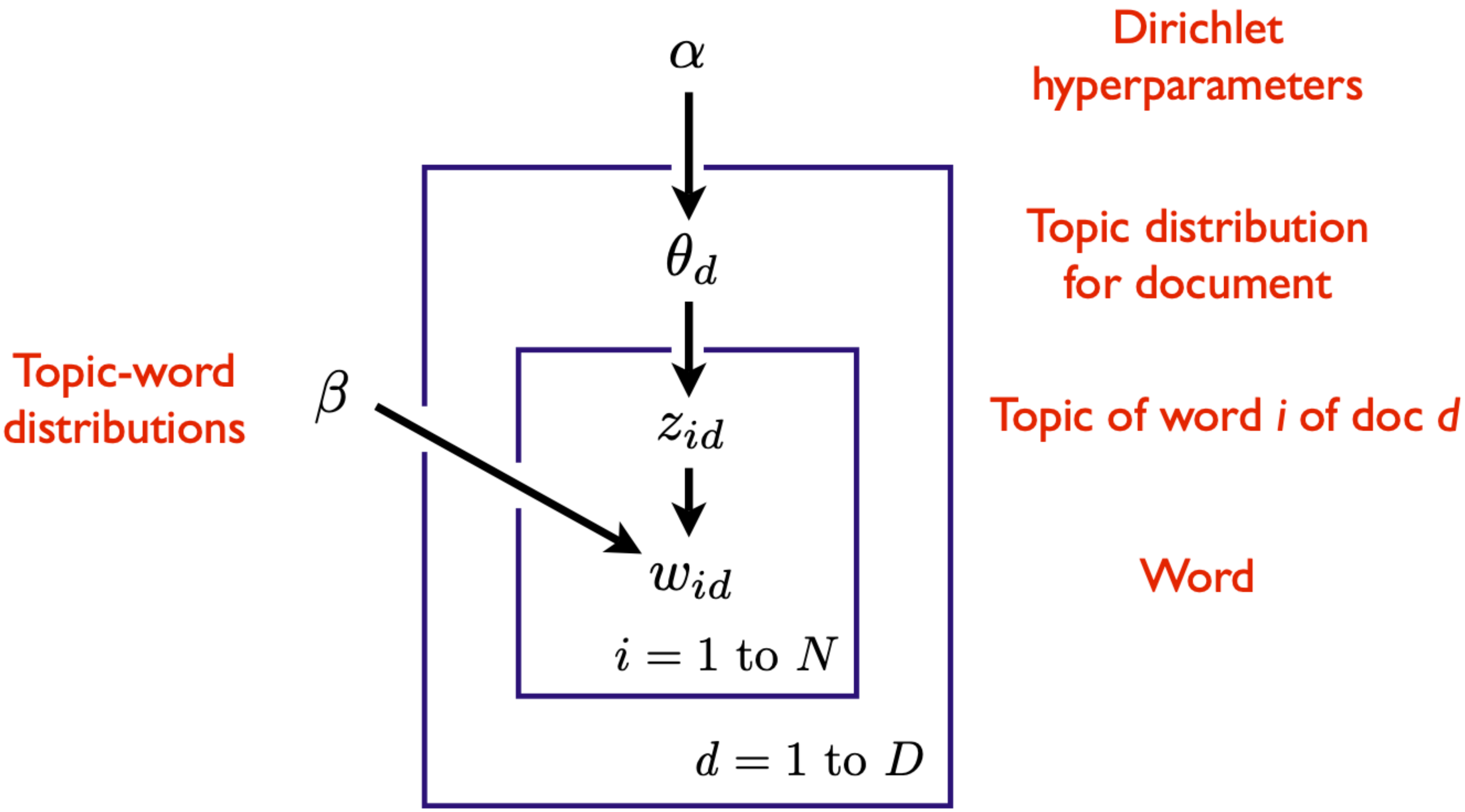
Latent Dirichlet Allocation (LDA)

- ***Every document is a mixture of topics***
- ***Every topic is a mixture of words***
- What it is trying to do is figure out what topics would create the original documents in the first place

Latent Dirichlet Allocation (LDA)

- An LDA model is defined by two parameters:
  - α—A prior estimate on topic probability (in other words, the average frequency that each topic within a given document occurs).
  - β—a collection of k topics where each topic is given a probability distribution over the vocabulary used in a document corpus, also called a "topic-word distribution."

Generative model



Toy example

	dog	cat	bird
Document 1	2	1	0
Document 2	0	0	3
Document 3	1	2	1

Document Term Matrix

Toy example

Step 1: Random Initialization of the document-topic matrix and the topic-term matrix

document-topic matrix

	Topic 1	Topic 2
D1	0.7	0.3
D2	0.4	0.6
D3	0.6	0.4

topic-term matrix

dog	0.6	0.4
cat	0.3	0.7
bird	0.4	0.6

Toy example

Step 1: Random Initialization of the document-topic matrix and the topic-term matrix

document-topic matrix

0.7	0.3
0.4	0.6
0.6	0.4

Document 1

topic-term matrix

0.6	0.4
0.3	0.7
0.4	0.6

Toy example

Step 1: Random Initialization of the document-topic matrix and the topic-term matrix

document-topic matrix

0.7	$\boxed{0.3}$
0.4	0.6
0.6	0.4

Probability that document 1 belongs to the 2nd topic

topic-term matrix

0.6	$\boxed{0.4}$
0.3	0.7
0.4	0.6

probability of each term given each topic

EM Algorithm: E-step

Calculate the posterior distribution of topic assignments for each word in each document

P(topic 1) = 0.6, P(topic 2) = 0.4

P(topic 1 | word "dog" in document D1) = (P(word "dog" | topic 1) * P(topic 1)) / (P(word "dog" | topic 1) * P(topic 1) + P(word "dog" | topic 2) * P(topic 2))

= (0.7) * (0.6) / (0.7 * 0.6 + 0.3 * 0.4) = 0.42/0.54

= 0.78

P(topic 2 | word "dog" in document D1) = (P(word "dog" | topic 2) * P(topic 2)) / (P(word "dog" | topic 2) * P(topic 2) + P(word "dog" | topic 1) * P(topic 1))

...

Repeat this calculation for all words in all document.

EM Algorithm: M-step

Update the estimates of the topic-word and topic prior probabilities, in order to maximize the likelihood of the observed document-topic assignments.

Repeat the E-step and M-step until convergence.

Performing LDA with Gensim

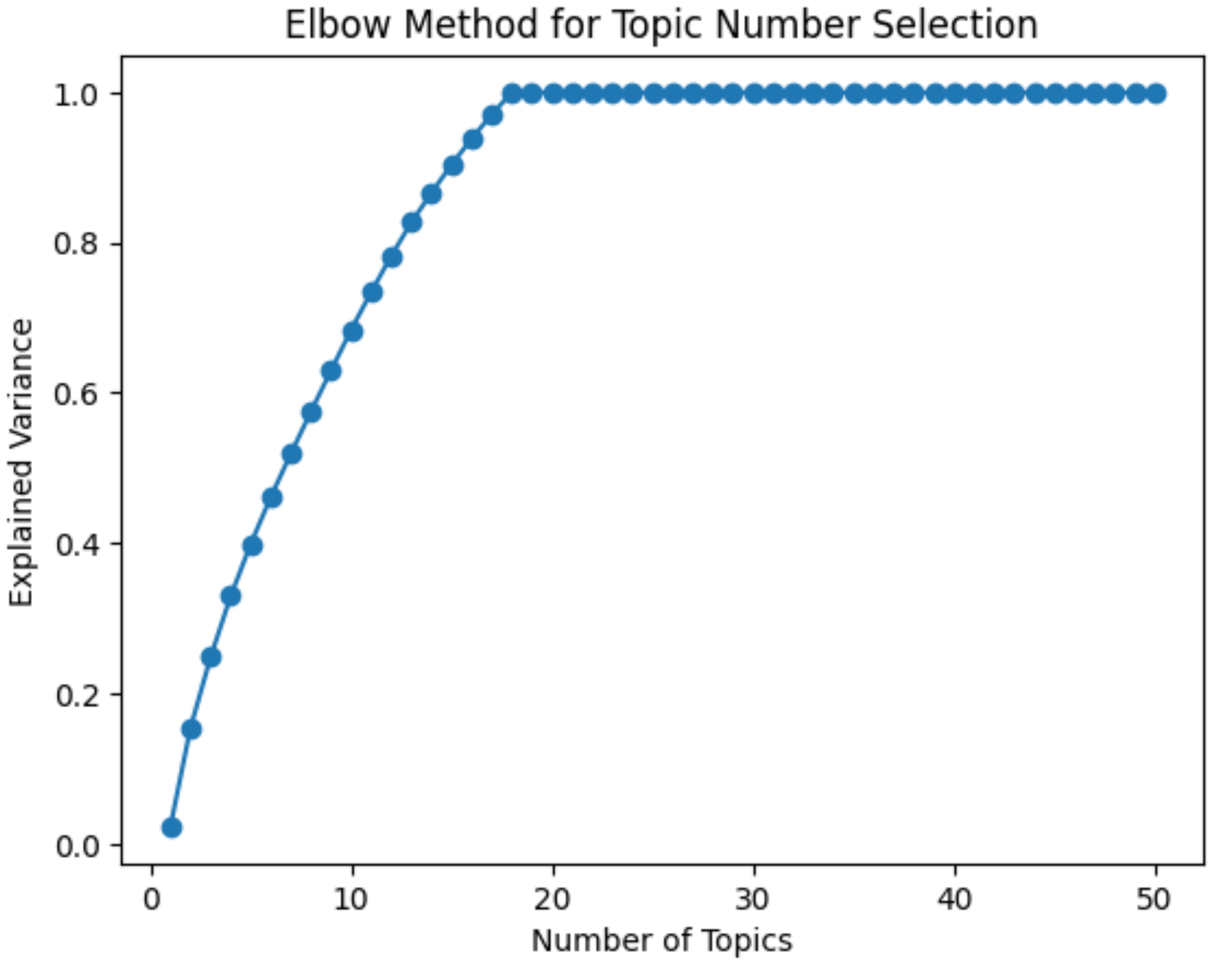
```
from gensim.test.utils import common_texts
from gensim.corpora.dictionary import Dictionary
# Create a corpus from a list of texts
common_dictionary = Dictionary(common_texts
common_corpus = [common_dictionary.doc2bow(text) for text in
common_texts]
# Train the model on the corpus
lda = LdaModel(common_corpus, num_topics=10)
```

How can we evaluate out topic modeling results?

- Human interpretation
- Kullback Leibler (KL) Divergence Score
- Perplexity
- Coherence
- Topic-Keyword Contingency Table

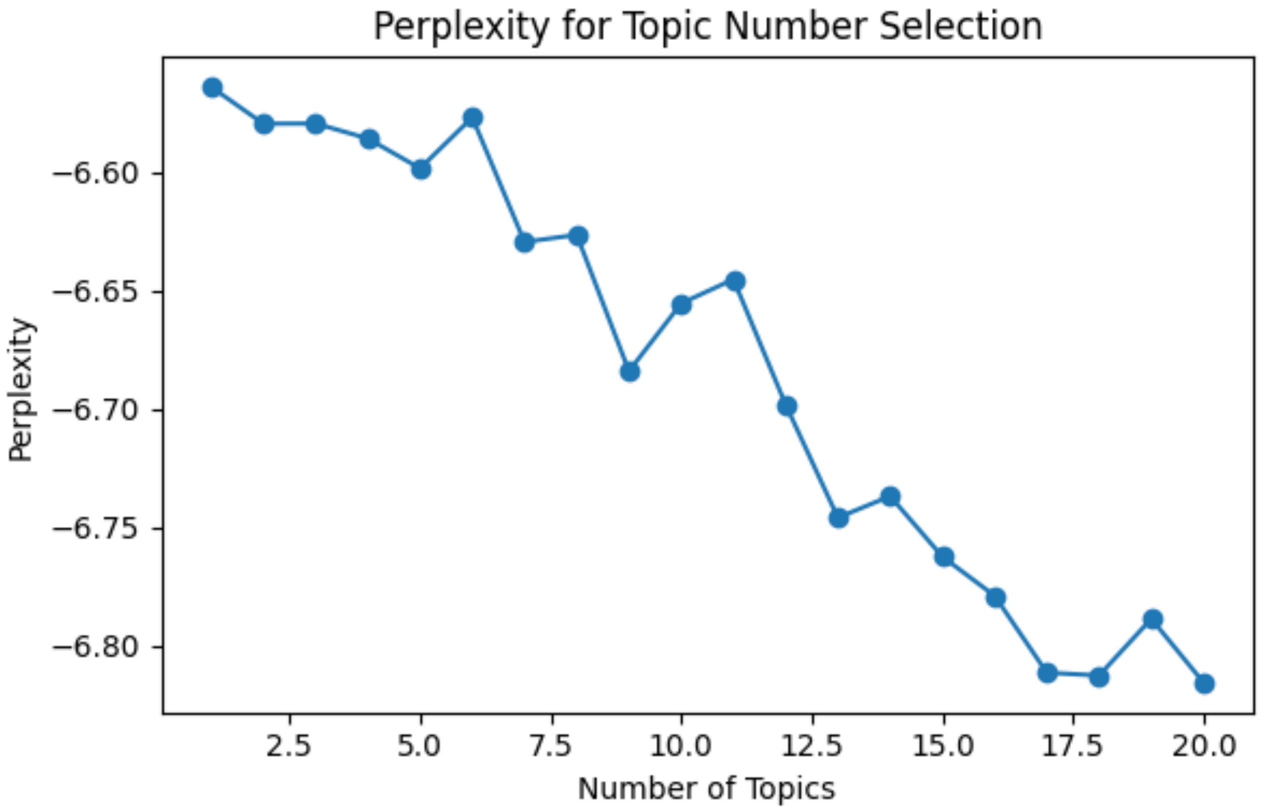
How can we decide how many topics to keep?

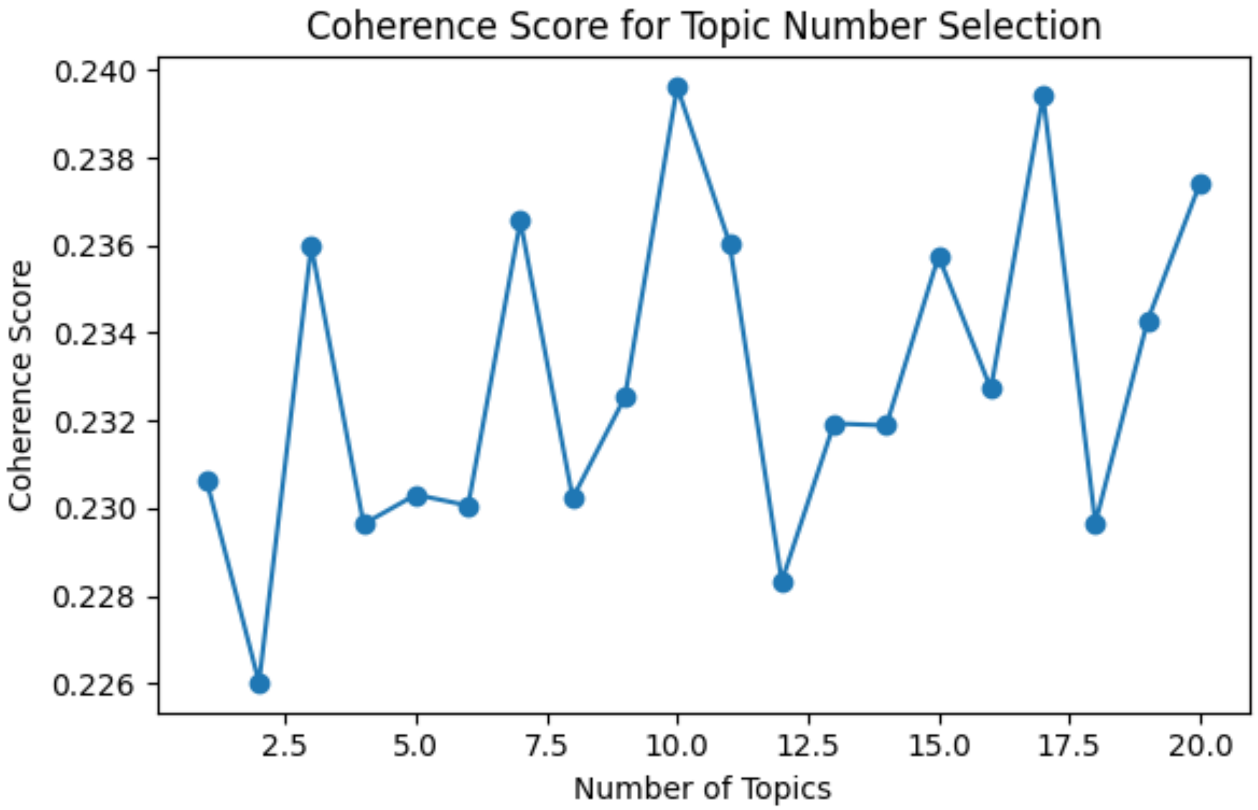
- The elbow method



How can we decide how many topics to keep?

- Perplexity and Coherence





In this example what kind of preprocessing steps should we do?

- Tokenization
- Lowercasing or any other normalization
- Stemming or lemmatization
- Remove numbers
- Stop words removal
- Removing rare words

- Basically, anything that can help to reduce the size of the document term matrix and get rid of things that will not add value to determining possible topics

Filtering words or tags for improving topic models

- POS tagging
  - POS tag IN contain– “within”, “upon”, “except”. “CD” contains – “one”, “two”, “hundred” etc. “MD” contains “may”, “must” etc.
  - These terms are not good as topics, nor do they tell us anything about the topics in the documents
  - Solution: Remove words labeled with these tags

Next time

- Using pretrained networks
  - Show some of the pretrained models we have already used + some new ones
  - How to use them for a specific task/dataset