EE6405

Natural Language Processing

Dr. S. Supraja
NTU Electrical and Electronic Engineering

© 2023 Nanyang Technological University, Singapore. All Rights Reserved.

Term Weighting, Topic Modeling and Dimensionality Reduction

© 2023 Nanyang Technological University, Singapore. All Rights Reserved.

Term Weighting Schemes

Term Weighting Schemes involves calculating and assigning a numerical value to each term, aiming to assess its significance in differentiating a specific document from the rest.

- Term Weighting seeks to represent the importance of a chosen query term in the corpus.
- Common term weighting schemes include:

TF-IDF

BM25

© 2023 Nanyang Technological University, Singapore. All Rights Reserved.

TF- IDF

Term Frequency-Inverse Document Frequency (TF- IDF)

Term Frequency refers to how often a term appears across a particular document.

Inverse Document Frequency refers to the relative rarity of a term in the collection of documents.

- The final TF-IDF value is obtained by multiplying these values together.
- The higher the TF-IDF score, the more important or relevant the term is; as a term gets less relevant, its TF-IDF score will approach 0.

© 2023 Nanyang Technological University, Singapore. All Rights Reserved.

TF- IDF

Calculating Term Frequency:

Term Frequency can be calculated in multiple ways:

- The raw count of how often a term appears in the document. $\boldsymbol{tf_{t,d}}$
- Term frequency adjusted for document length. $\boldsymbol{\left(\frac{tf_{t,d}}{n.words}\right)}$
- Logarithmically Scaled. $\mathbf{1+\log}(\boldsymbol{tf_{t,d}})$
- Boolean Frequency (1 if term appears in document, 0 if term does not appear in document).

© 2023 Nanyang Technological University, Singapore. All Rights Reserved.

TF- IDF

Calculating Inverse Document Frequency:

Inverse document frequency can be defined as:

$$idf(t, D) = log(\frac{N}{count(d \in D: t \in d)})$$

- Where *N* is the total number of documents (*d*) in the corpus (*D*).
- The denominator is the number of documents where the term t appears in.
- If a term does not appear in the corpus, a divide by zero error may occur. This can be avoided using the following equation:

$$idf(t, D) = log\left(\frac{1 + N}{1 + count(d \in D: t \in d)}\right) + 1$$

© 2023 Nanyang Technological University, Singapore. All Rights Reserved.

TF- IDF

Calculating TF- IDF:

TF-IDF is simply defined as:

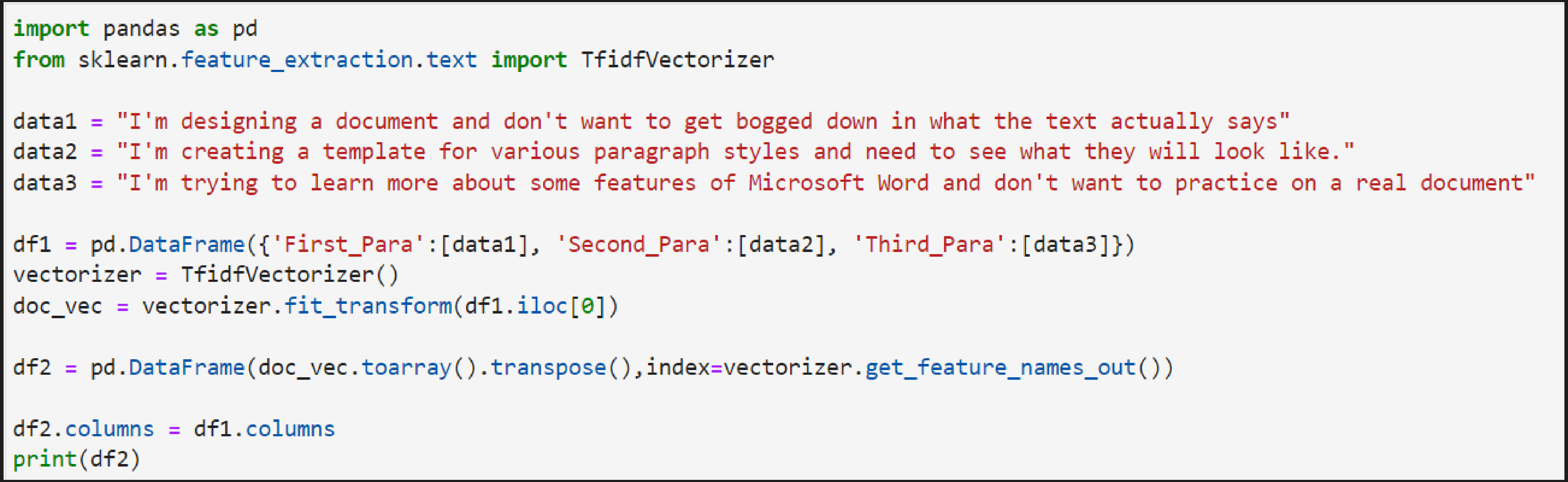
$$\boldsymbol{tf(t,d) \cdot idf(t,d)}$$

- The higher the TF-IDF score the more important or relevant the term is; as a term gets less relevant, its TF-IDF score will approach 0.
- TF-IDF vectorization involves calculating the TF-IDF score for every word in your corpus relative to that document and then putting that information into a vector.
- Each document in the corpus would have its own vector, and the vector would have a TF-IDF score for every single word in the entire collection of documents.
- The similarity between the documents can be derived from the cosine similarity between the two vectors.

© 2023 Nanyang Technological University, Singapore. All Rights Reserved.

TF- IDF

TF- IDF vectorization with SKLearn:



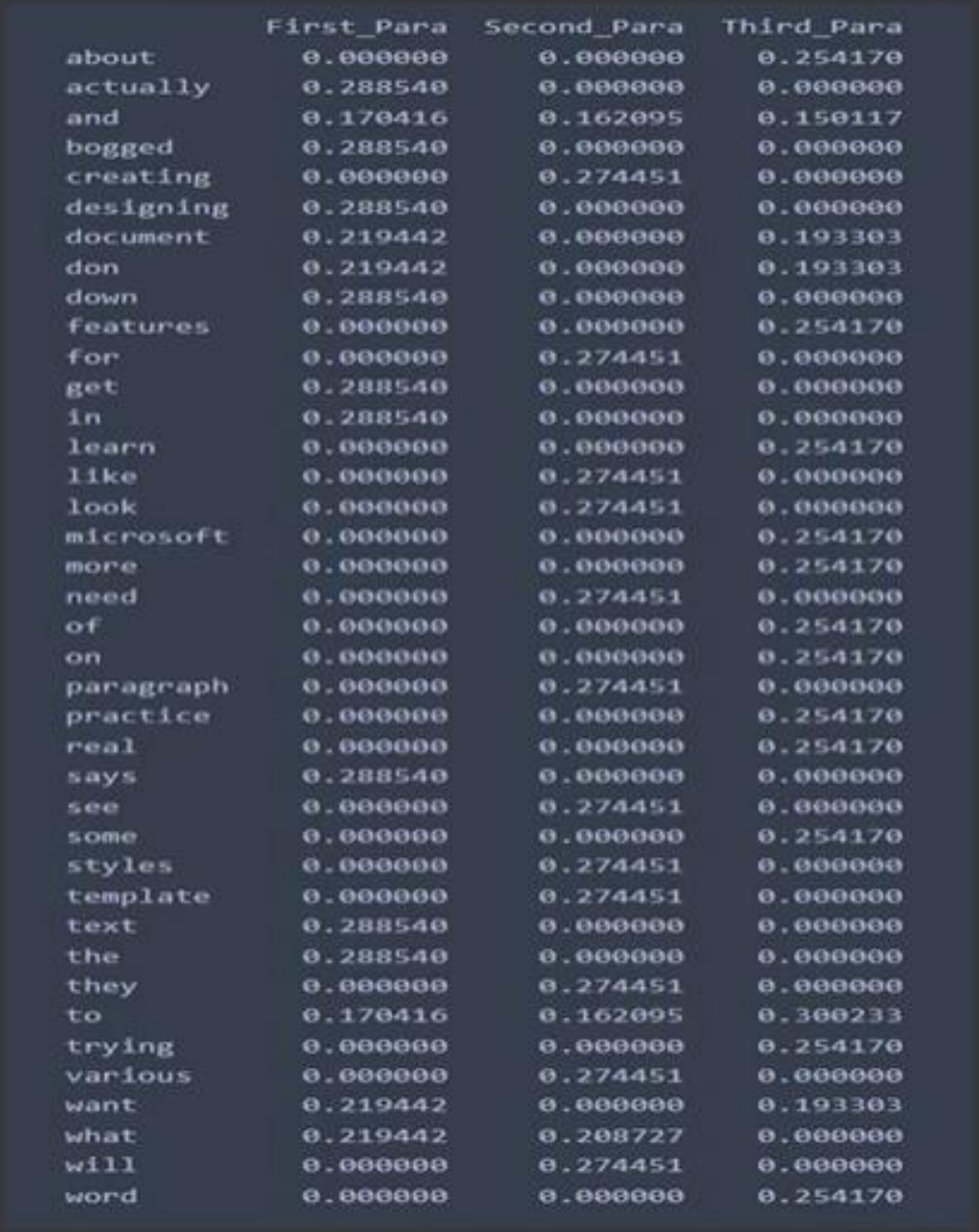
```python
import pandas as pd
from sklearn.feature_extraction.text import TfidfVectorizer

data1 = "I'm designing a document and don't want to get bogged down in what the text actually says"
data2 = "I'm creating a template for various paragraph styles and need to see what they will look like."
data3 = "I'm trying to learn more about some features of Microsoft Word and don't want to practice on a real document"

df1 = pd.DataFrame({'First_Para':[data1], 'Second_Para':[data2], 'Third_Para':[data3]})
vectorizer = TfidfVectorizer()
doc_vec = vectorizer.fit_transform(df1.iloc[0])

df2 = pd.DataFrame(doc_vec.toarray().transpose(),index=vectorizer.get_feature_names_out())

df2.columns = df1.columns
print(df2)
```

© 2023 Nanyang Technological University, Singapore. All Rights Reserved.

TF- IDF

TF- IDF vectorisation with SKLearn:

Words that appear more across each document have a higher TF-IDF score, while words that are unique to each document have a TF- IDF score of 0.



```
           First_Para  Second_Para  Third_Para
about        0.000000     0.000000    0.254170
actually     0.288540     0.000000    0.000000
and          0.170416     0.162095    0.150117
bogged       0.288540     0.000000    0.000000
creating     0.000000     0.274451    0.000000
designing    0.288540     0.000000    0.000000
document     0.219442     0.000000    0.193303
don          0.219442     0.000000    0.193303
down         0.288540     0.000000    0.000000
features     0.000000     0.000000    0.254170
for          0.000000     0.274451    0.000000
get          0.288540     0.000000    0.000000
in           0.288540     0.000000    0.000000
learn        0.000000     0.000000    0.254170
like         0.000000     0.274451    0.000000
look         0.000000     0.274451    0.000000
microsoft    0.000000     0.000000    0.254170
more         0.000000     0.000000    0.254170
need         0.000000     0.274451    0.000000
of           0.000000     0.000000    0.254170
on           0.000000     0.000000    0.254170
paragraph    0.000000     0.274451    0.000000
practice     0.000000     0.000000    0.254170
real         0.000000     0.000000    0.254170
says         0.288540     0.000000    0.000000
see          0.000000     0.274451    0.000000
some         0.000000     0.000000    0.254170
styles       0.000000     0.274451    0.000000
template     0.000000     0.274451    0.000000
text         0.288540     0.000000    0.000000
the          0.288540     0.000000    0.000000
they         0.000000     0.274451    0.000000
to           0.170416     0.162095    0.300233
trying       0.000000     0.000000    0.254170
various      0.000000     0.274451    0.000000
want         0.219442     0.000000    0.193303
what         0.219442     0.208727    0.000000
will         0.000000     0.274451    0.000000
word         0.000000     0.000000    0.254170
```

© 2023 Nanyang Technological University, Singapore. All Rights Reserved.

TF- IDF

Advantages

- **Efficiency:**
  It is computationally efficient to calculate TF-IDF scores for documents, especially compared to more complex natural language processing techniques.
- **Language Agnostic:**
  TF-IDF can be applied to documents in any language, making it versatile for multilingual text analysis.
- **No Supervised Training:**
  TF-IDF doesn't require a training phase with labeled data, making it readily applicable to various tasks without the need for extensive training data.

Limitations

- **No Semantics:**
  TF-IDF treats words as independent units and doesn't consider the semantics or relationships between words.
- **Term Frequency Bias:**
  Terms that appear more frequently tend to receive higher weights, which can lead to overemphasis on common words that are not very informative.
- **Vocabulary Size:**
  The size of the vocabulary (unique terms across the corpus) can become a computational challenge for large datasets, and rare terms might not receive accurate IDF values.

© 2023 Nanyang Technological University, Singapore. All Rights Reserved.

BM25

Best Match 25 (BM25)

BM25 is an algorithm that considers both term frequency (TF) and document length normalisation to determine the relevance of a document to a given query.

- It follows the probabilistic retrieval framework, which assumes that relevant and non-relevant documents follow different statistical distributions.

© 2023 Nanyang Technological University, Singapore. All Rights Reserved.

BM25

Best Match 25 (BM25)

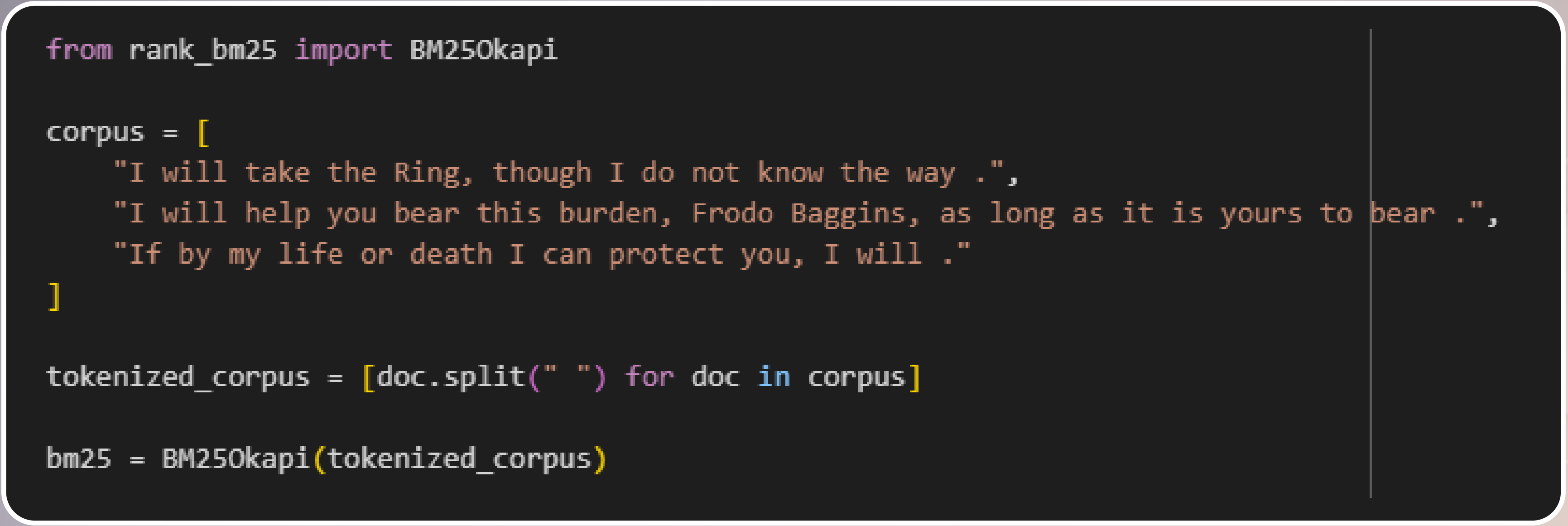
$$\boldsymbol{BM25(D,Q) = \sum_{i=1}^{n} IDF(q_i) \cdot \frac{TF(q_i, D) \cdot (k_1 + 1)}{TF(q_i, D) + k_1 \cdot \left(1 - b + b \cdot \frac{|D|}{avgdl}\right)}}$$

Where:

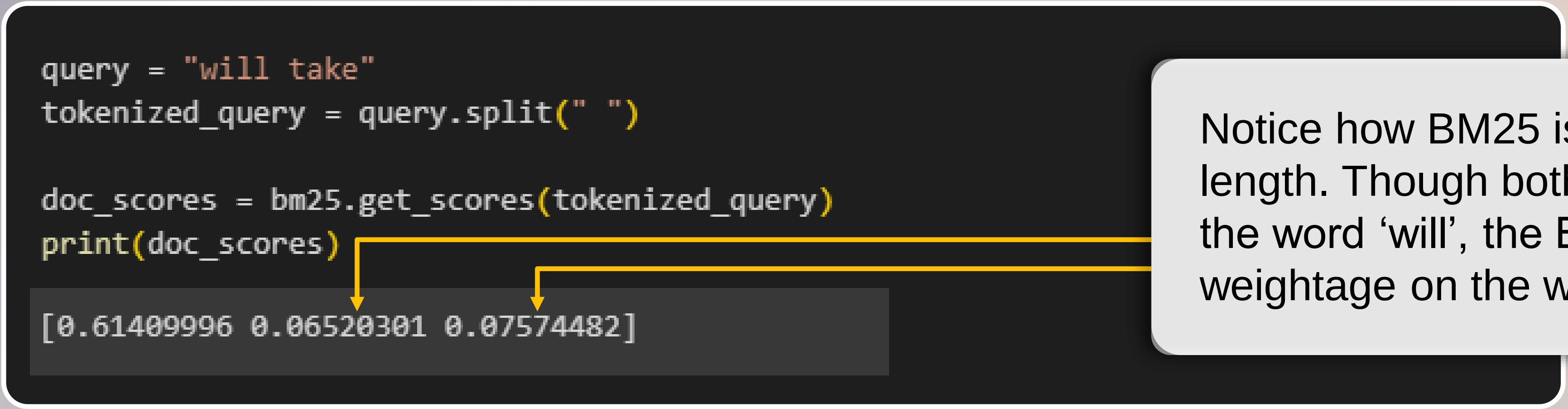
- IDF(q) represents the inverse document frequency of the query term q.
- TF(q, D) denotes the modified term frequency of term q in document D.
- |D| represents the length of document D.
- avgdl is the average document length in the corpus.
- Parameters k1 and b are tunable constants that control the impact of term frequency saturation and document length normalisation, respectively.

© 2023 Nanyang Technological University, Singapore. All Rights Reserved.

BM25



```
from rank_bm25 import BM25Okapi

corpus = [
    "I will take the Ring, though I do not know the way .",
    "I will help you bear this burden, Frodo Baggins, as long as it is yours to bear .",
    "If by my life or death I can protect you, I will ."
]

tokenized_corpus = [doc.split(" ") for doc in corpus]

bm25 = BM25Okapi(tokenized_corpus)
```



```
query = "will take"
tokenized_query = query.split(" ")

doc_scores = bm25.get_scores(tokenized_query)
print(doc_scores)
```

```
[0.61409996 0.06520301 0.07574482]
```

Notice how BM25 is adjusted for document length. Though both sentences 2 & 3 contain the word ‘will’, the BM25 algorithm puts higher weightage on the word in a shorter document.

© 2023 Nanyang Technological University, Singapore. All Rights Reserved.

BM25

Advantages

- **Information Retrieval:**
  BM25 is known to be effective in information retrieval tasks. It often outperforms traditional TF-IDF weighting, especially when dealing with large and diverse text corpora.
- **Balanced Term Frequency:**
  BM25 addresses the issue of term frequency saturation by introducing the parameter 'k.' This makes it more robust in handling documents with varying term frequencies, preventing common terms from dominating the score.

Limitations

- **No Semantics:**
  Similarly to TF-IDF, BM25 does not take into account the semantics of a document.
- **Sparse Data:**
  BM25 may not perform as well when dealing with very sparse data or extremely short documents because it relies on term frequencies and document lengths to calculate scores.

© 2023 Nanyang Technological University, Singapore. All Rights Reserved.

Topic Modeling

Topic Modeling Schemes:

Topic Modeling is an unsupervised method for document classification, aiming to discover and extract hidden thematic structures within large collections of text data.

- This is achieved by analysing word co-occurrence patterns within unstructured text data.
- The hidden thematic structures can then be used to classify and summarise the available data.
- Common Topic modelling techniques include:

LDA **LSA** **NMF**

© 2023 Nanyang Technological University, Singapore. All Rights Reserved.

LDA

Latent Dirichlet Allocation (LDA)

LDA aims to find the representative words for a topic and classify a document based on the occurrence of these words

- LDA consists of 2 parts:
  - Words belonging to a document.
  - Words belonging to a topic.
- LDA works off a Bag-Of-Words approach – the order and semantics of a word is not considered
- LDA requires the user to set a predetermined number of topics, *k*.

© 2023 Nanyang Technological University, Singapore. All Rights Reserved.

LDA

Algorithm

Go through each document in the corpus and randomly assign each word in the document to one of *k* topics.

- For each document *d*, go through each word *w* and find:
  - $\boldsymbol{p(topic\ t|document\ d)}$**:** The proportion of words in document *d* that are assigned to topic *t*. Captures how many words belong to topic *t* in document *d*.
  - $\boldsymbol{p(word\ w|topic\ t)}$**:** The proportion of assignments to topic *t* from all documents that come from word *w*. Captures how many documents are in topic *t* due to document *w*.
  - Update the probability of word *w* belonging to topic *t* using:

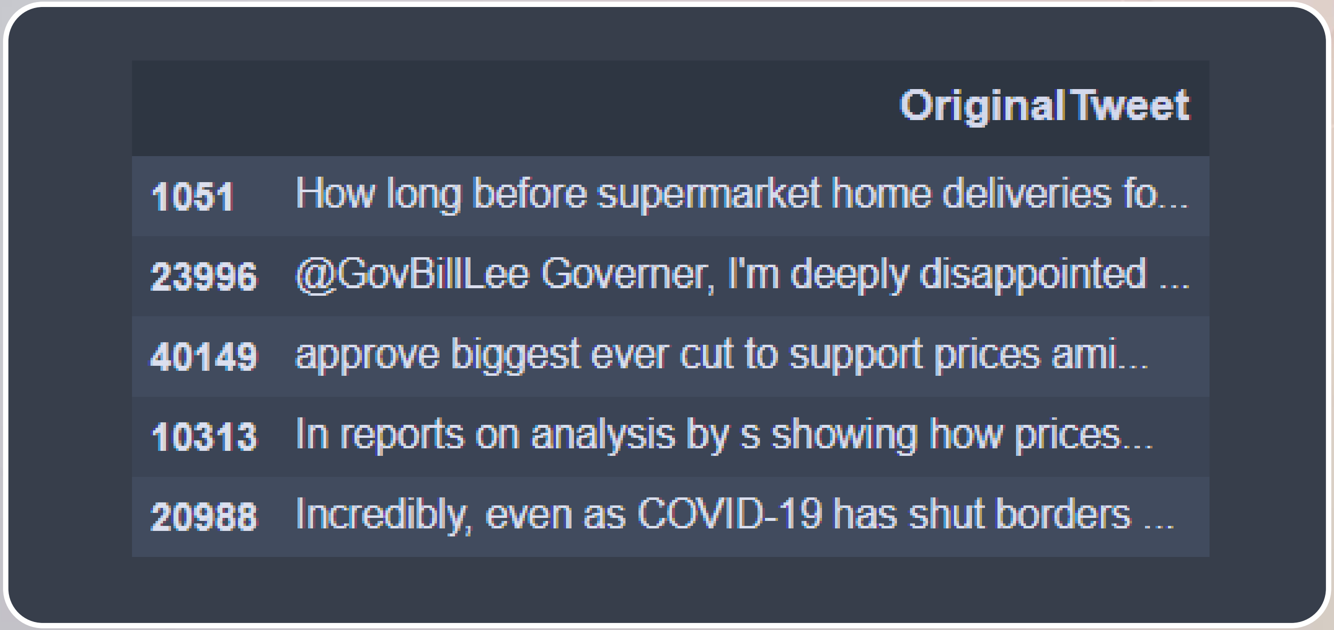
$$\boldsymbol{p(word\ w\ belongs\ to\ topic\ t) = p(topic\ t|document\ d) \cdot p(word\ w|topic\ t)}$$

© 2023 Nanyang Technological University, Singapore. All Rights Reserved.

LDA

LDA in gensim:

- For this demonstration, we will use a compilation of COVID tweets found at: https://www.kaggle.com/datasets/datatattle/covid-19-nlp-text-classification
- We first filter out only the tweets:



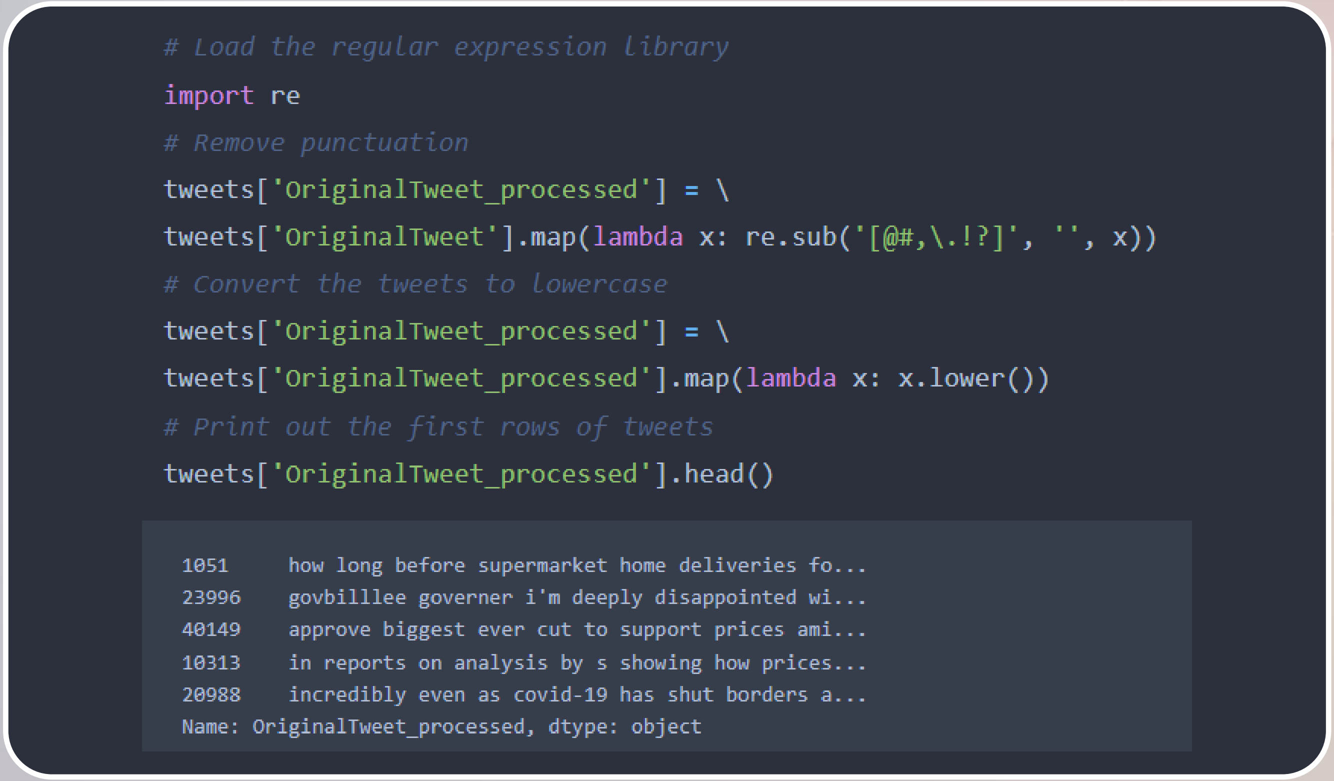
	OriginalTweet
1051	How long before supermarket home deliveries fo...
23996	@GovBillLee Governer, I'm deeply disappointed ...
40149	approve biggest ever cut to support prices ami...
10313	In reports on analysis by s showing how prices...
20988	Incredibly, even as COVID-19 has shut borders ...

© 2023 Nanyang Technological University, Singapore. All Rights Reserved.

LDA

LDA in gensim:

- To make the data easier to process, we remove all punctuation and convert all words to lowercase.



```
# Load the regular expression library
import re
# Remove punctuation
tweets['OriginalTweet_processed'] = \
tweets['OriginalTweet'].map(lambda x: re.sub('[@#,\.!?]', '', x))
# Convert the tweets to lowercase
tweets['OriginalTweet_processed'] = \
tweets['OriginalTweet_processed'].map(lambda x: x.lower())
# Print out the first rows of tweets
tweets['OriginalTweet_processed'].head()
```

```
1051     how long before supermarket home deliveries fo...
23996    govbilllee governer i'm deeply disappointed wi...
40149    approve biggest ever cut to support prices ami...
10313    in reports on analysis by s showing how prices...
20988    incredibly even as covid-19 has shut borders a...
Name: OriginalTweet_processed, dtype: object
```

© 2023 Nanyang Technological University, Singapore. All Rights Reserved.

LDA

LDA in gensim:

- As LDA is a BOW model, we want to remove stop words that don’t carry much semantic meaning.

```
import gensim
from gensim.utils import simple_preprocess
import nltk
nltk.download('stopwords')
from nltk.corpus import stopwords
stop_words = stopwords.words('english')
stop_words.extend(['https', 'tco'])
def sent_to_words(sentences):
    for sentence in sentences:
        # deacc=True removes punctuations
        yield(gensim.utils.simple_preprocess(str(sentence), deacc=True))
def remove_stopwords(texts):
    return [[word for word in simple_preprocess(str(doc))
             if word not in stop_words] for doc in texts]
data = tweets.OriginalTweet_processed.values.tolist()
data_words = list(sent_to_words(data))
# remove stop words
data_words = remove_stopwords(data_words)
print(data_words[:1][0][:30])
```

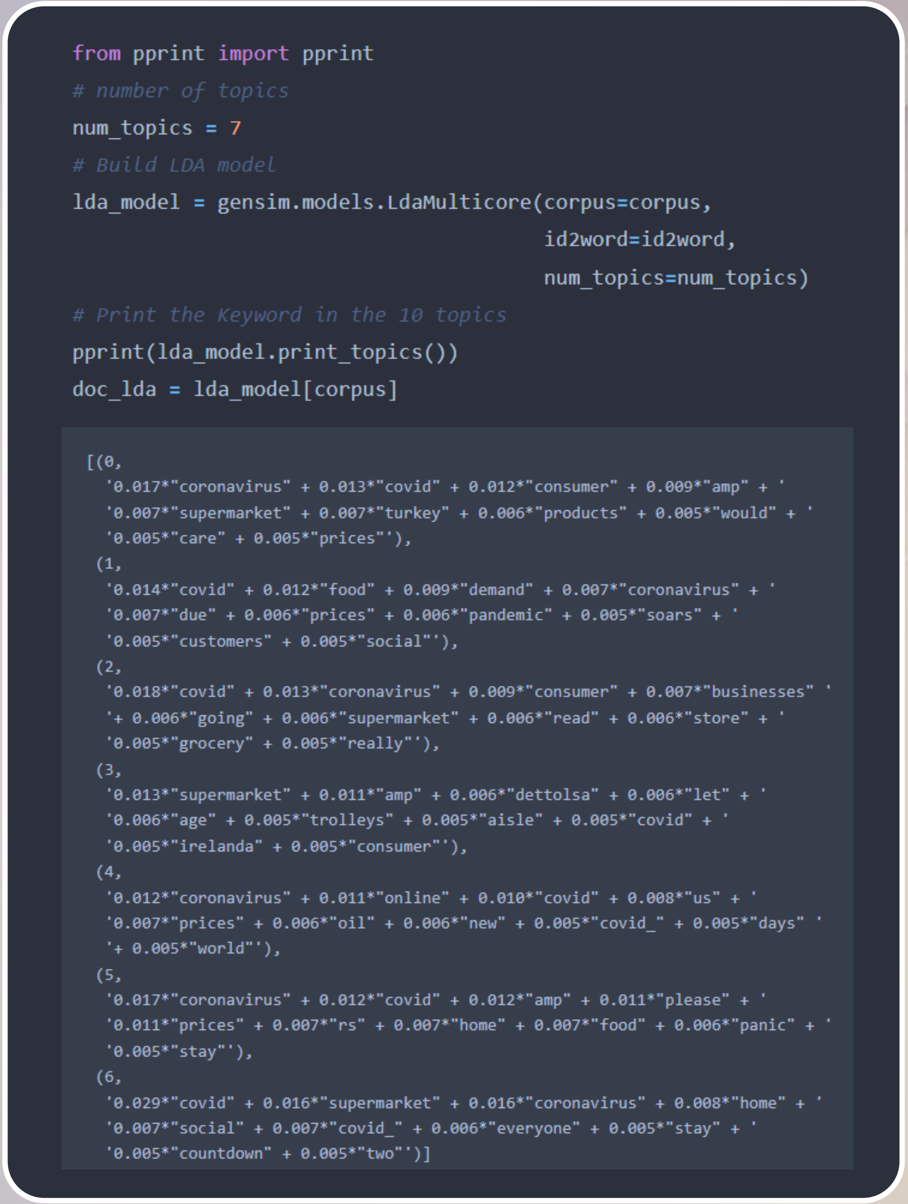
Note: ‘https’ and ‘tco’ are parts of shared hyperlinks within tweets. They will be removed for this example.

© 2023 Nanyang Technological University, Singapore. All Rights Reserved.

LDA

LDA in gensim:

- With k =7, we can generate the LDA scores for each word in a tweet and classify them into a topic.



```
from pprint import pprint
# number of topics
num_topics = 7
# Build LDA model
lda_model = gensim.models.LdaMulticore(corpus=corpus,
                                       id2word=id2word,
                                       num_topics=num_topics)
# Print the Keyword in the 10 topics
pprint(lda_model.print_topics())
doc_lda = lda_model[corpus]
```

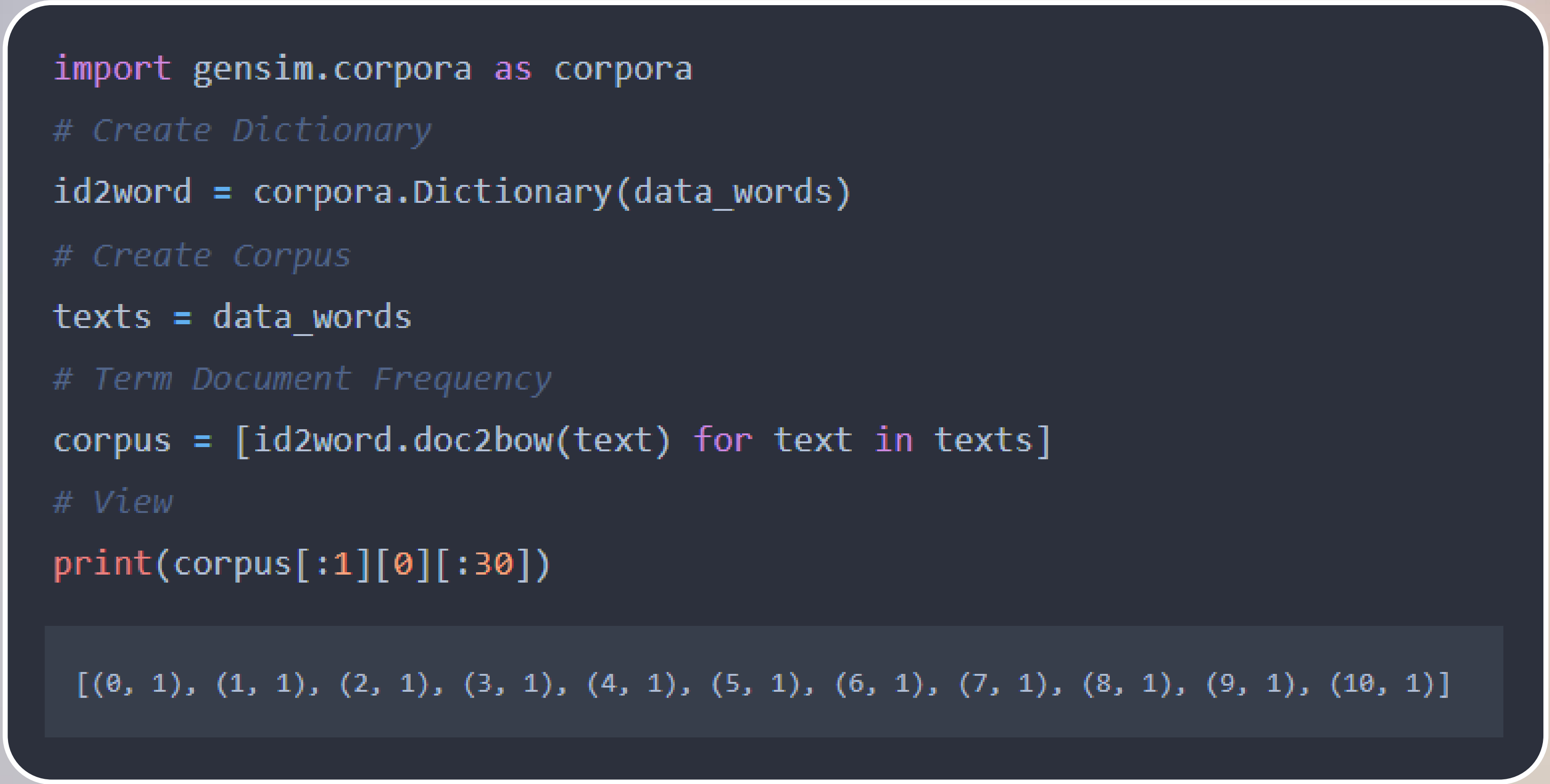
```
[(0,
  '0.017*"coronavirus" + 0.013*"covid" + 0.012*"consumer" + 0.009*"amp" + '
  '0.007*"supermarket" + 0.007*"turkey" + 0.006*"products" + 0.005*"would" + '
  '0.005*"care" + 0.005*"prices"'),
 (1,
  '0.014*"covid" + 0.012*"food" + 0.009*"demand" + 0.007*"coronavirus" + '
  '0.007*"due" + 0.006*"prices" + 0.006*"pandemic" + 0.005*"soars" + '
  '0.005*"customers" + 0.005*"social"'),
 (2,
  '0.018*"covid" + 0.013*"coronavirus" + 0.009*"consumer" + 0.007*"businesses" '
  '+ 0.006*"going" + 0.006*"supermarket" + 0.006*"read" + 0.006*"store" + '
  '0.005*"grocery" + 0.005*"really"'),
 (3,
  '0.013*"supermarket" + 0.011*"amp" + 0.006*"dettolsa" + 0.006*"let" + '
  '0.006*"age" + 0.005*"trolleys" + 0.005*"aisle" + 0.005*"covid" + '
  '0.005*"irelanda" + 0.005*"consumer"'),
 (4,
  '0.012*"coronavirus" + 0.011*"online" + 0.010*"covid" + 0.008*"us" + '
  '0.007*"prices" + 0.006*"oil" + 0.006*"new" + 0.005*"covid_" + 0.005*"days" '
  '+ 0.005*"world"'),
 (5,
  '0.017*"coronavirus" + 0.012*"covid" + 0.012*"amp" + 0.011*"please" + '
  '0.011*"prices" + 0.007*"rs" + 0.007*"home" + 0.007*"food" + 0.006*"panic" + '
  '0.005*"stay"'),
 (6,
  '0.029*"covid" + 0.016*"supermarket" + 0.016*"coronavirus" + 0.008*"home" + '
  '0.007*"social" + 0.007*"covid_" + 0.006*"everyone" + 0.005*"stay" + '
  '0.005*"countdown" + 0.005*"two"')]
```

© 2023 Nanyang Technological University, Singapore. All Rights Reserved.

LDA

LDA in gensim:

- Create a dictionary of all words in the corpus



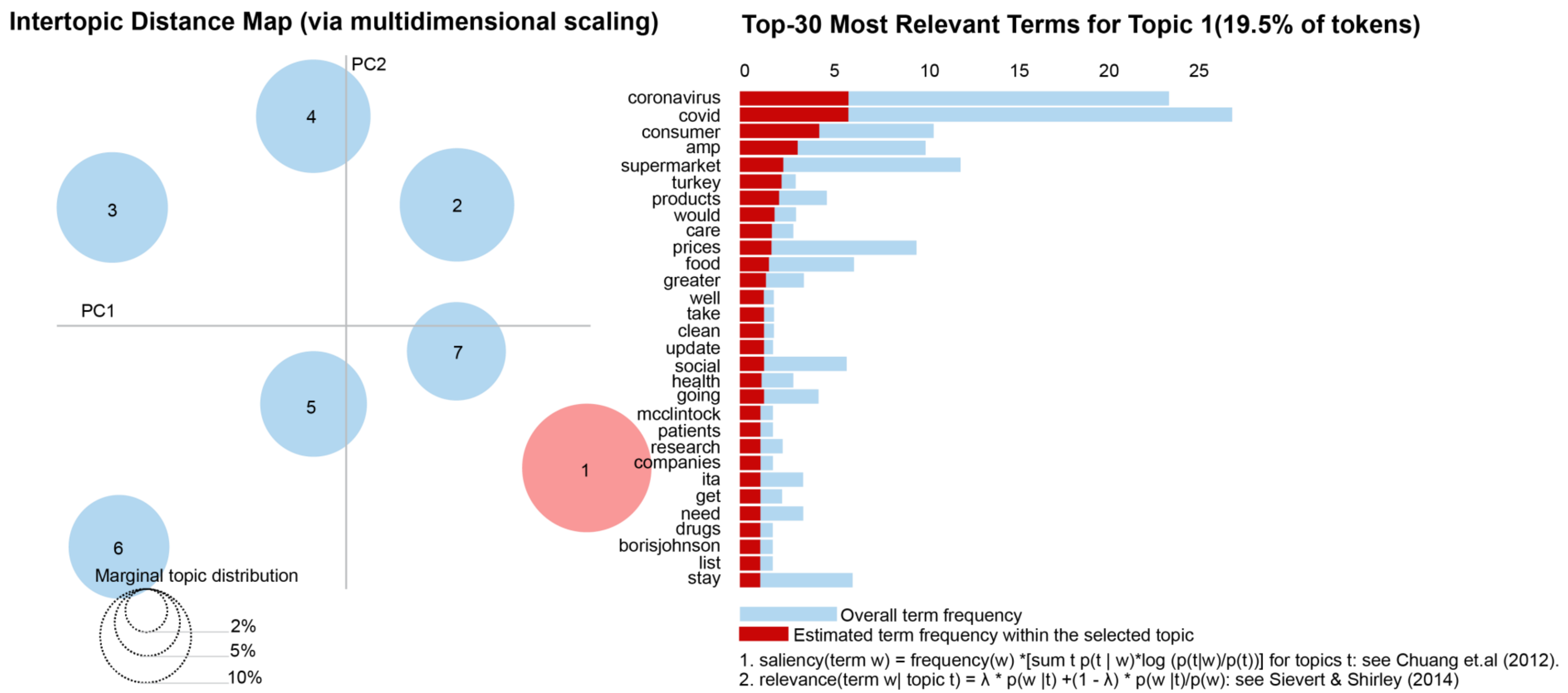
```python
import gensim.corpora as corpora
# Create Dictionary
id2word = corpora.Dictionary(data_words)
# Create Corpus
texts = data_words
# Term Document Frequency
corpus = [id2word.doc2bow(text) for text in texts]
# View
print(corpus[:1][0][:30])
```

```
[(0, 1), (1, 1), (2, 1), (3, 1), (4, 1), (5, 1), (6, 1), (7, 1), (8, 1), (9, 1), (10, 1)]
```

© 2023 Nanyang Technological University, Singapore. All Rights Reserved.

LDA

Visualisation:



1. saliency(term w) = frequency(w) *[sum t p(t | w)*log (p(t|w)/p(t))] for topics t: see Chuang et.al (2012).
2. relevance(term w| topic t) = λ * p(w |t) +(1 - λ) * p(w |t)/p(w): see Sievert & Shirley (2014)

© 2023 Nanyang Technological University, Singapore. All Rights Reserved.

LDA

Advantages

- **Dimensionality Reduction:**
  LDA reduces the dimensionality of the data by representing documents as mixtures of topics. This simplifies data representation.
- **Interpretability:**
  The topics generated by LDA are typically represented as lists of words, making them interpretable.

Limitations

- **No Semantics:**
  LDA treats words independently of their context in a document. Hence, the semantics of a word are not captured with LDA.
- **Preprocessing Sensitivity:**
  LDA is highly sensitive to the preprocessing used on data, such as stop word removal and stemming.

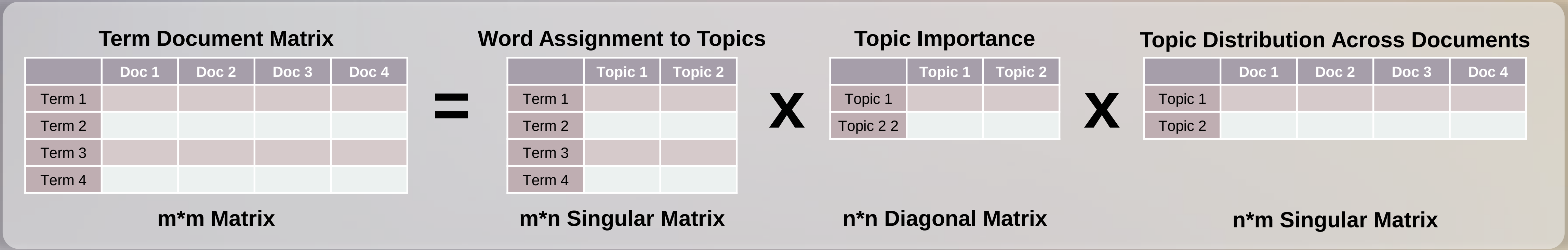
© 2023 Nanyang Technological University, Singapore. All Rights Reserved.

LSA

Latent Semantic Analysis (LSA):

LSA aims to uncover the underlying semantic structure of text data and transform it into a lower-dimensional space.

- LSA works by generating a document-term matrix.
- The matrix is typically populated with the TF-IDF scores for each word.
- The matrix is then decomposed (typically with SVD) into sub-matrixes.



Term Document Matrix

	Doc 1	Doc 2	Doc 3	Doc 4
Term 1				
Term 2				
Term 3				
Term 4				

m*m Matrix

=

Word Assignment to Topics

	Topic 1	Topic 2
Term 1		
Term 2		
Term 3		
Term 4		

m*n Singular Matrix

X

Topic Importance

	Topic 1	Topic 2
Topic 1		
Topic 2 2		

n*n Diagonal Matrix

X

Topic Distribution Across Documents

	Doc 1	Doc 2	Doc 3	Doc 4
Topic 1				
Topic 2				

n*m Singular Matrix

© 2023 Nanyang Technological University, Singapore. All Rights Reserved.

LSA

Steps

Generate a document-term matrix of shape mxn:

The column entries can be filled by a raw count of the number of times a word appears in the document, but the TF-IDF score is usually used as it gives a better representation of the significance of the word.

Terms

Documents

	T1	**T2**	**T3**	**…**	**Tn**
D1	0.2	0.1	0.3	…	0.5
D2	0.1	0.3	0.2	…	0.4
…	…	…	…	…	…
Dn	0.3	0.5	0.3	…	0.3

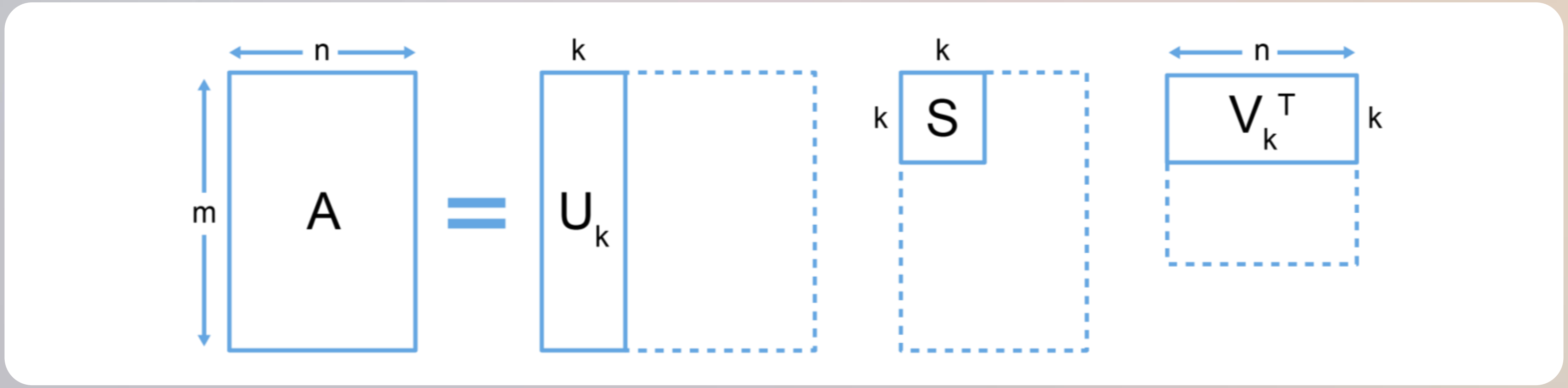
© 2023 Nanyang Technological University, Singapore. All Rights Reserved.

LSA

Steps

By applying SVD (explained in detail later), break the matrix down into

- Document-Topic Matrix $\boldsymbol{U_k}$
  - Document vectors expressed in terms of topics
- Term-Topic Matrix $\boldsymbol{V_k}^{\boldsymbol{T}}$
  - Term vectors in terms of topics



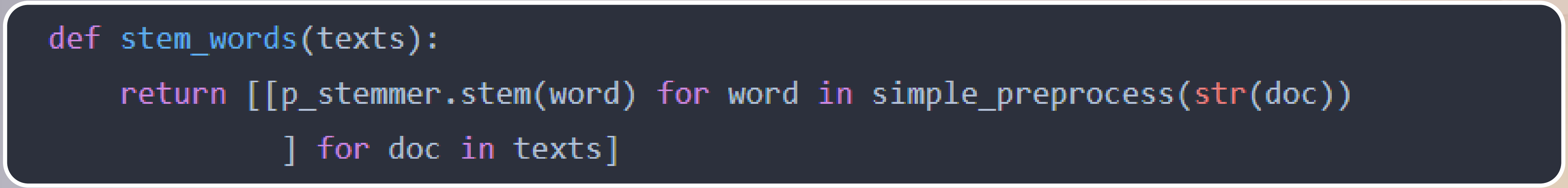
© 2023 Nanyang Technological University, Singapore. All Rights Reserved.

LSA

LSA in gensim

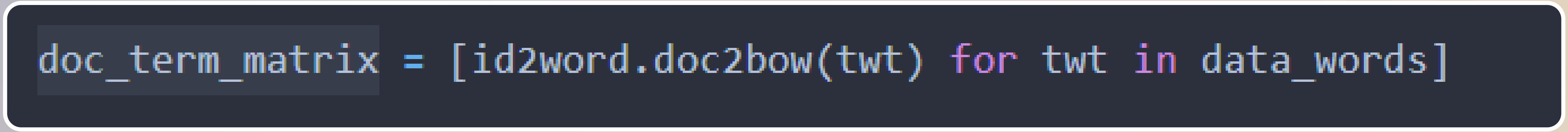
Using the same COVID dataset

- Stem words to reduce noise



```
def stem_words(texts):
    return [[p_stemmer.stem(word) for word in simple_preprocess(str(doc))
             ] for doc in texts]
```

- Stem words to reduce noise



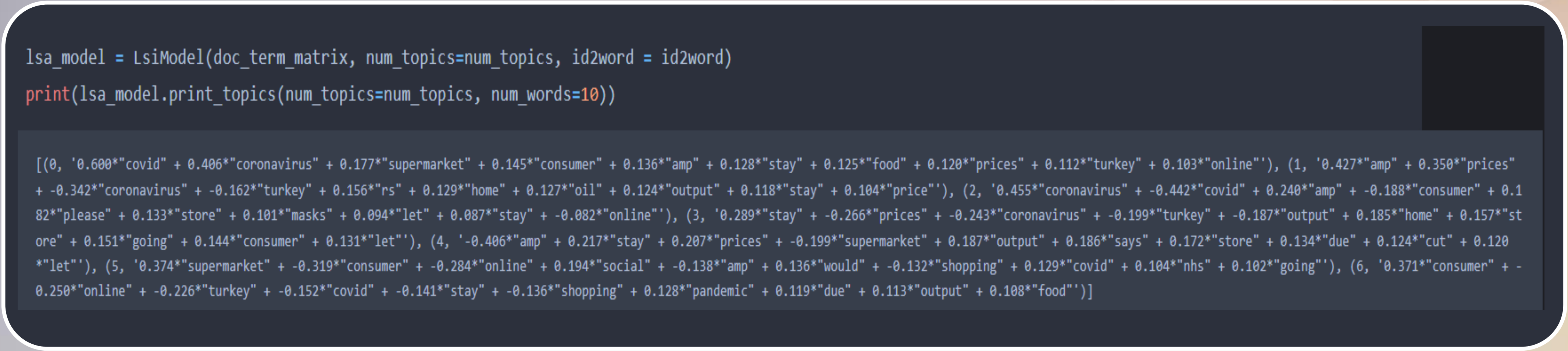
```
doc_term_matrix = [id2word.doc2bow(twt) for twt in data_words]
```

© 2023 Nanyang Technological University, Singapore. All Rights Reserved.

LSA

LSA in gensim

We can then build the LSA model and extract the top 10 relevant keywords.



```
lsa_model = LsiModel(doc_term_matrix, num_topics=num_topics, id2word = id2word)
print(lsa_model.print_topics(num_topics=num_topics, num_words=10))
```

```
[(0, '0.600*"covid" + 0.406*"coronavirus" + 0.177*"supermarket" + 0.145*"consumer" + 0.136*"amp" + 0.128*"stay" + 0.125*"food" + 0.120*"prices" + 0.112*"turkey" + 0.103*"online"'), (1, '0.427*"amp" + 0.350*"prices"
+ -0.342*"coronavirus" + -0.162*"turkey" + 0.156*"rs" + 0.129*"home" + 0.127*"oil" + 0.124*"output" + 0.118*"stay" + 0.104*"price"'), (2, '0.455*"coronavirus" + -0.442*"covid" + 0.240*"amp" + -0.188*"consumer" + 0.1
82*"please" + 0.133*"store" + 0.101*"masks" + 0.094*"let" + 0.087*"stay" + -0.082*"online"'), (3, '0.289*"stay" + -0.266*"prices" + -0.243*"coronavirus" + -0.199*"turkey" + -0.187*"output" + 0.185*"home" + 0.157*"st
ore" + 0.151*"going" + 0.144*"consumer" + 0.131*"let"'), (4, '-0.406*"amp" + 0.217*"stay" + 0.207*"prices" + -0.199*"supermarket" + 0.187*"output" + 0.186*"says" + 0.172*"store" + 0.134*"due" + 0.124*"cut" + 0.120
*"let"'), (5, '0.374*"supermarket" + -0.319*"consumer" + -0.284*"online" + 0.194*"social" + -0.138*"amp" + 0.136*"would" + -0.132*"shopping" + 0.129*"covid" + 0.104*"nhs" + 0.102*"going"'), (6, '0.371*"consumer" + -
0.250*"online" + -0.226*"turkey" + -0.152*"covid" + -0.141*"stay" + -0.136*"shopping" + 0.128*"pandemic" + 0.119*"due" + 0.113*"output" + 0.108*"food"')]
```

© 2023 Nanyang Technological University, Singapore. All Rights Reserved.

LSA

Advantages

- **Dimensionality Reduction:**
  LSA reduces the dimensionality of the data by representing documents as mixtures of topics. This simplifies data representation.
- **Semantic Understanding:**
  LSA captures the underlying semantic structure of text, allowing it to identify and measure semantic similarities between words and documents, even when they don't share exact word overlap.

Limitations

- **Lack of Interpretability:**
  The reduced semantic space created by LSA lacks human interpretability. While it captures semantic relationships, it may not provide direct insights into the meaning of specific dimensions.
- **Preprocessing Sensitivity:**
  LSA is highly sensitive to the preprocessing used on data, such as stop word removal and stemming.

© 2023 Nanyang Technological University, Singapore. All Rights Reserved.

pLSA

Probabilistic Latent Semantic Analysis (pLSA):

pLSA is an extension of LSA that operates on a probabilistic framework.

- Assumes documents are generated from a mixture of latent topics.
- Each topic is associated with a distribution over words, and each document is assumed to be generated by selecting a topic from this mixture and then choosing words from the selected topic's word distribution.

© 2023 Nanyang Technological University, Singapore. All Rights Reserved.

pLSA

Probabilistic Latent Semantic Analysis (pLSA):

The goal of pLSA is to find a model that can generate the data observed within a term-document matrix.

- This means we want a function P(*w,d*) that will generate the corresponding entry in the word-document matrix given the word w and document *d*.

$$\boldsymbol{P(w,d) = P(d)\sum_t P(w|t)P(t|d)}$$

- Intuitively, this means the probability of seeing a particular word in a document should be given by the product of probability of the document P(*d*) and sum over all topics for the conditional distribution of words given topic P*(w*|*t*) and P(*t*|*d*).

© 2023 Nanyang Technological University, Singapore. All Rights Reserved.

LSA vs pLSA

LSA

- **SVD based decomposition:**
  LSA is a relatively straightforward technique that relies on singular value decomposition (SVD) for dimensionality reduction. As a result, LSA is easily scalable.
- **Interpretability:**
  While LSA can reveal semantic relationships between words and documents, it may not provide human-interpretable topics, as it doesn't explicitly model topics in a probabilistic way.

pLSA

- **Probabilistic modelling:**
  pLSA views the generation of a document as a probabilistic mixture of topics. This means pLSA is less scalable than LSA due to its higher complexity.
- **Interpretability:**
  pLSA is more interpretable as it explicitly defines topics as probability distributions over words, allowing users to interpret topics based on their word distributions.

© 2023 Nanyang Technological University, Singapore. All Rights Reserved.

NMF

Non-negative Matrix Factorisation (NMF):

NMF aims to find latent topics within a document by factorising a term-document matrix.

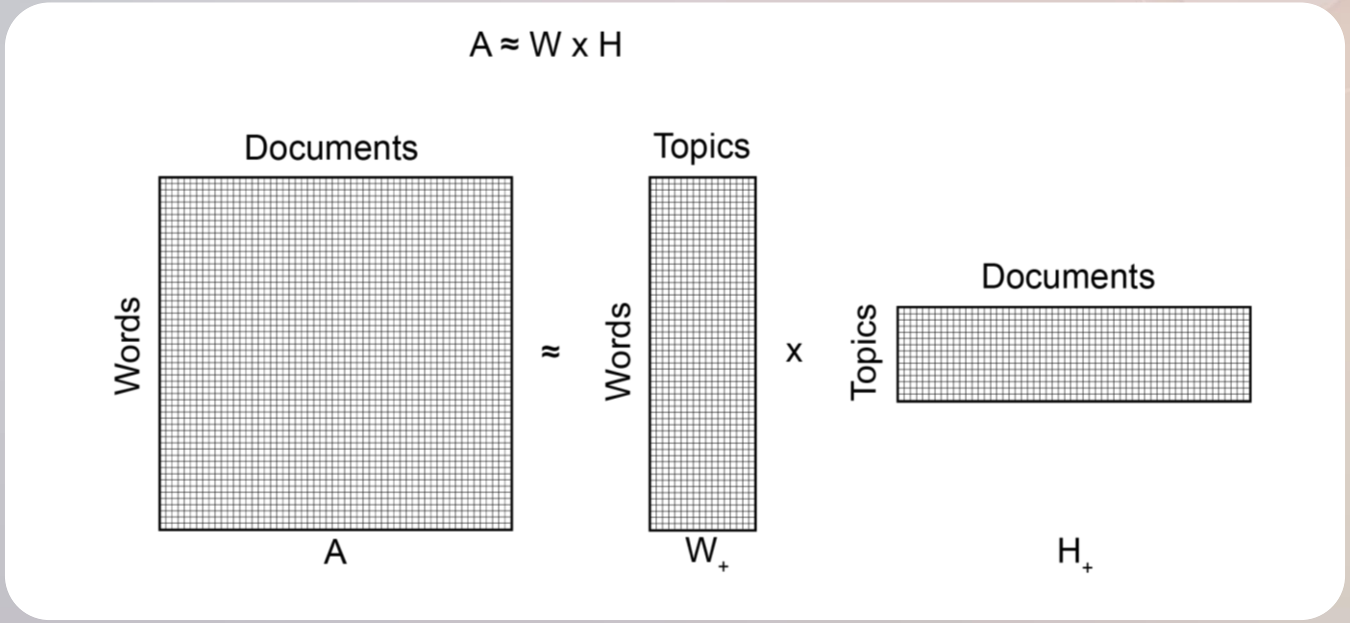
- The term-document matrix is factorised into 2 parts:
  - A word-topic matrix
  - A topic document matrix
- These matrices can then be used to infer the semantic relations of words with each sentence.

© 2023 Nanyang Technological University, Singapore. All Rights Reserved.

NMF

Steps:

- Similar to LSA, a term-document matrix of size $\boldsymbol{n \times m}$ normalised using TF-IDF needs to be generated.
- The matrix is then factorised into two matrixes, one of n-words by k-topics and the other of k-topics by m-original documents.



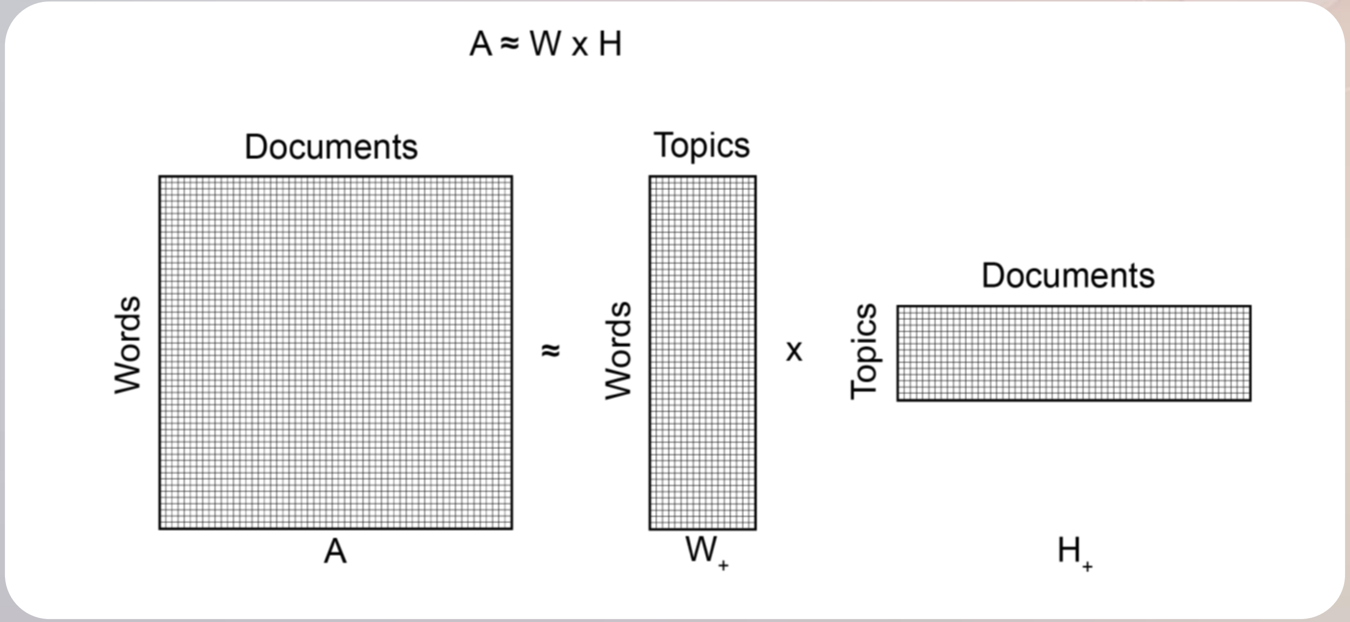
© 2023 Nanyang Technological University, Singapore. All Rights Reserved.

NMF

Steps:

- Intuitively, $\boldsymbol{W}_+$ represents every topic and the terms found within the topic.
- $\boldsymbol{H}_+$ represents every document and the topics found within the document.

It is assumed that all elements of matrices **W** and **H** are positive given the elements of **A** are positive.



© 2023 Nanyang Technological University, Singapore. All Rights Reserved.

NMF

Steps:

When factorising the matrix, the two main cost functions that can be used are:

- Generalised Kullback–Leibler Divergence

$$kl_{div(x,y)} = \begin{cases} x\log\left(\frac{x}{y}\right) - x + y & x > 0, y > 0 \\ y & x = 0, y \geq 0 \\ \infty & otherwise \end{cases}$$

As the value of the KL-divergence reaches zero, the closeness of corresponding words increases.

- Frobenius Norm

$$\|A\|_F \equiv \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n}|a_{ij}|^2}$$

Defined as the square root of the sum of the absolute squares of its elements, the Frobenius Norm is a method of measuring how good an approximation is.

© 2023 Nanyang Technological University, Singapore. All Rights Reserved.

NMF

Implementation with Scikit-Learn:

```
# use tfidf by removing tokens that don't appear in at least 50 documents
vect = TfidfVectorizer(min_df=10, stop_words=stop_words )

# Fit and transform
X = vect.fit_transform(tweets.OriginalTweet)
```

Filter out words that don’t appear in at least 10 tweets

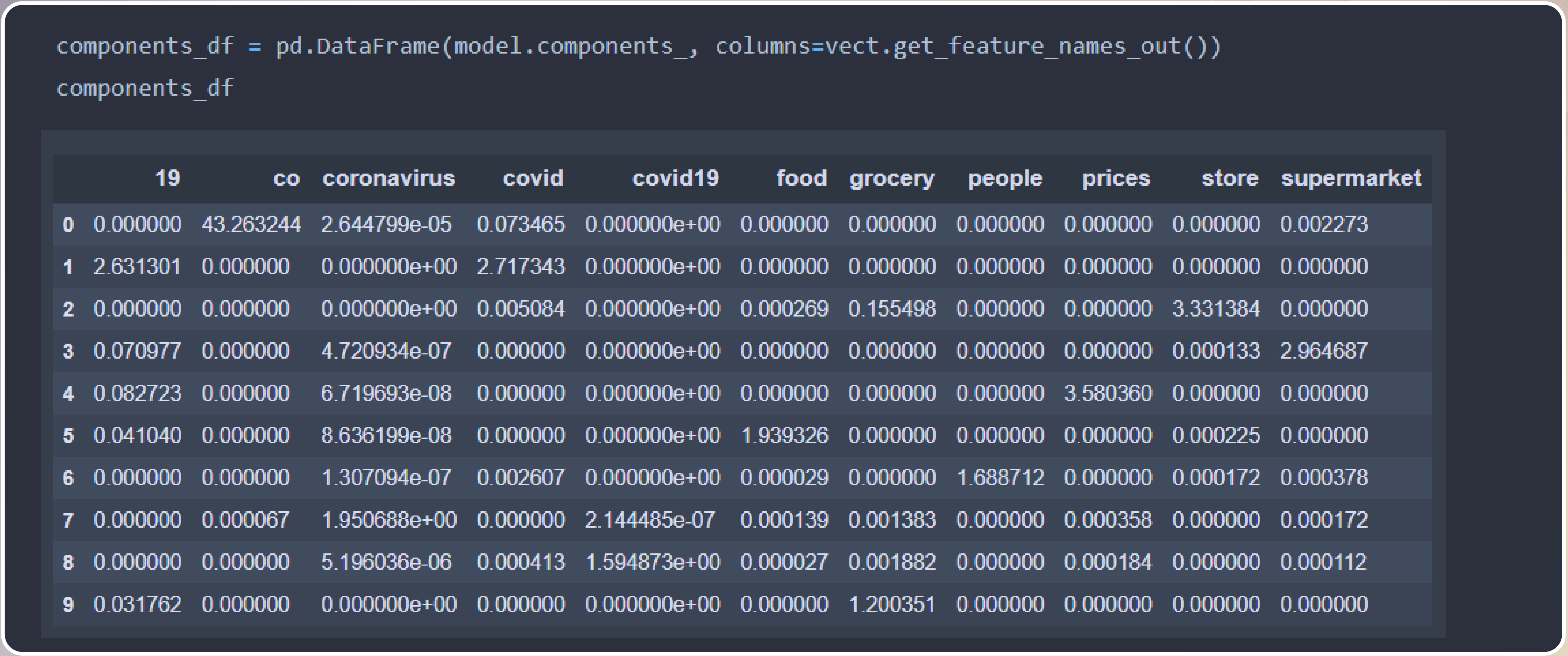
```
# Create an NMF instance: model
# the 10 components will be the topics
model = NMF(n_components=10, random_state=5)

# Fit the model to TF-IDF
model.fit(X)

# Transform the TF-IDF: nmf_features
nmf_features = model.transform(X)
```

N-components here refers to the number of topics.

© 2023 Nanyang Technological University, Singapore. All Rights Reserved.

NMF

Implementation with Scikit-Learn:



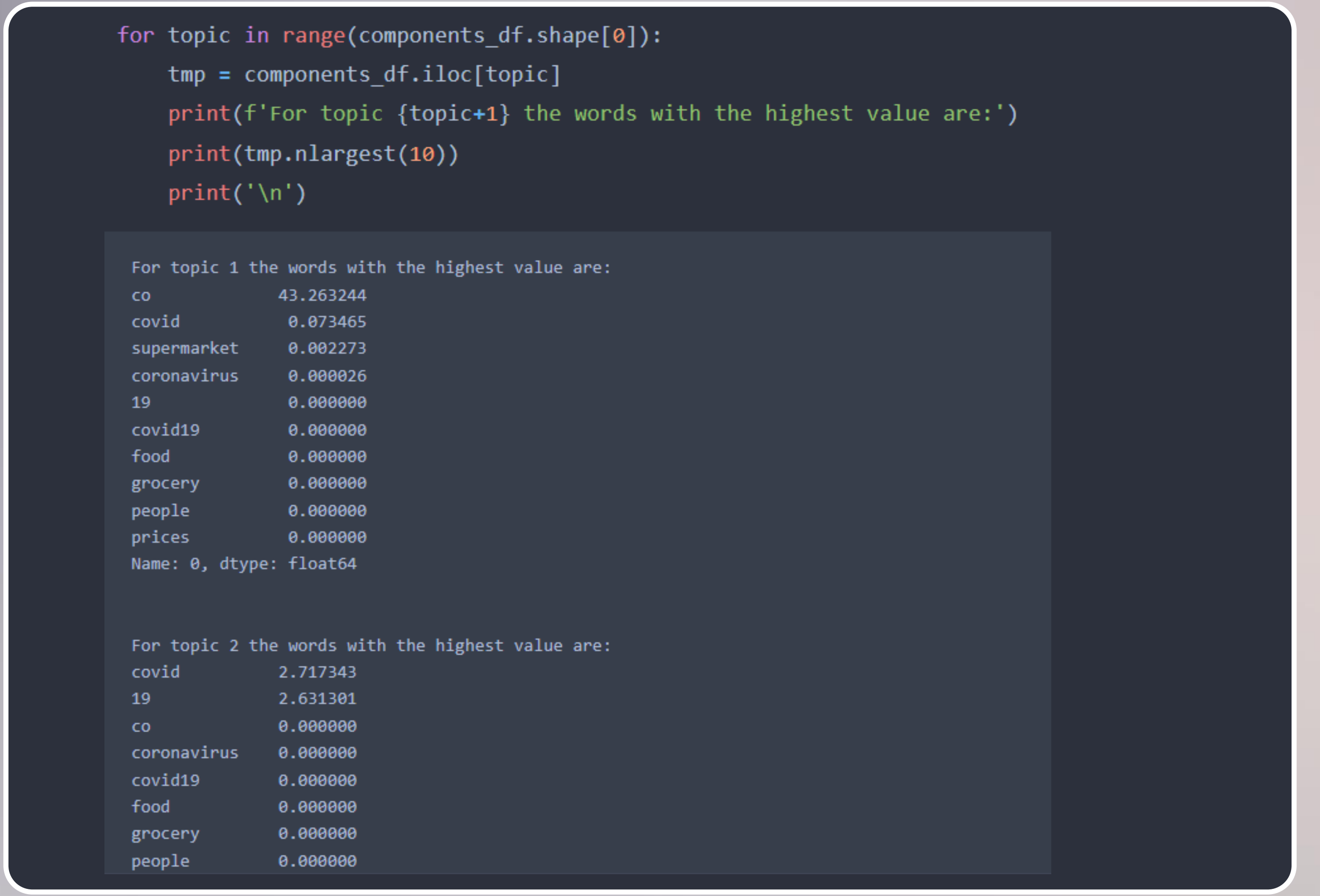
```python
components_df = pd.DataFrame(model.components_, columns=vect.get_feature_names_out())
components_df
```

	19	co	coronavirus	covid	covid19	food	grocery	people	prices	store	supermarket
0	0.000000	43.263244	2.644799e-05	0.073465	0.000000e+00	0.000000	0.000000	0.000000	0.000000	0.000000	0.002273
1	2.631301	0.000000	0.000000e+00	2.717343	0.000000e+00	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
2	0.000000	0.000000	0.000000e+00	0.005084	0.000000e+00	0.000269	0.155498	0.000000	0.000000	3.331384	0.000000
3	0.070977	0.000000	4.720934e-07	0.000000	0.000000e+00	0.000000	0.000000	0.000000	0.000000	0.000133	2.964687
4	0.082723	0.000000	6.719693e-08	0.000000	0.000000e+00	0.000000	0.000000	0.000000	3.580360	0.000000	0.000000
5	0.041040	0.000000	8.636199e-08	0.000000	0.000000e+00	1.939326	0.000000	0.000000	0.000000	0.000225	0.000000
6	0.000000	0.000000	1.307094e-07	0.002607	0.000000e+00	0.000029	0.000000	1.688712	0.000000	0.000172	0.000378
7	0.000000	0.000067	1.950688e+00	0.000000	2.144485e-07	0.000139	0.001383	0.000000	0.000358	0.000000	0.000172
8	0.000000	0.000000	5.196036e-06	0.000413	1.594873e+00	0.000027	0.001882	0.000000	0.000184	0.000000	0.000112
9	0.031762	0.000000	0.000000e+00	0.000000	0.000000e+00	0.000000	1.200351	0.000000	0.000000	0.000000	0.000000

© 2023 Nanyang Technological University, Singapore. All Rights Reserved.

NMF

Implementation with Scikit-Learn:



```python
for topic in range(components_df.shape[0]):
    tmp = components_df.iloc[topic]
    print(f'For topic {topic+1} the words with the highest value are:')
    print(tmp.nlargest(10))
    print('\n')
```

```
For topic 1 the words with the highest value are:
co             43.263244
covid           0.073465
supermarket     0.002273
coronavirus     0.000026
19              0.000000
covid19         0.000000
food            0.000000
grocery         0.000000
people          0.000000
prices          0.000000
Name: 0, dtype: float64


For topic 2 the words with the highest value are:
covid          2.717343
19             2.631301
co             0.000000
coronavirus    0.000000
covid19        0.000000
food           0.000000
grocery        0.000000
people         0.000000
```

Printing the top 10 words in each topic.

© 2023 Nanyang Technological University, Singapore. All Rights Reserved.

NMF

Advantages

- **Parts-Based Representation:**
  NMF naturally produces parts-based representations, which is valuable in applications like image processing and text mining. It can discover fundamental components or topics within data.
- **Noise Reduction:**
  NMF can help reduce the impact of noise in data because it focuses on capturing underlying patterns and structures rather than specific noisy details.

Limitations

- **Non-convex Optimization:**
  NMF is based on non-convex optimisation, which means it may converge to local minima rather than the global minimum. The quality of results can be sensitive to initialisation.
- **Preprocessing Sensitivity:**
  NMF is highly sensitive to the preprocessing used on data, as well as the choice of cost function for the model.

© 2023 Nanyang Technological University, Singapore. All Rights Reserved.

Topic Modeling

Summary:

LSA

- A linear algebra-based technique that uses singular value decomposition (SVD) for dimensionality reduction.
- Used to capture semantic relationships between terms and documents, improve information retrieval, or perform dimensionality reduction on text data.

LDA

- A probabilistic generative model used for topic modeling in text data.
- Used to discover latent topics in text data, categorise documents into topics, or perform content recommendation based on topics. Suited for clustering tasks

NMF

- Factorises a term-document matrix into two lower-dimensional non-negative matrices.
- Used to create parts-based representations to discover interpretable topics in text. NMF is valuable in scenarios where non-negativity and sparsity constraints are important.

© 2023 Nanyang Technological University, Singapore. All Rights Reserved.

Dimensionality Reduction

Why do we need dimensionality reduction?

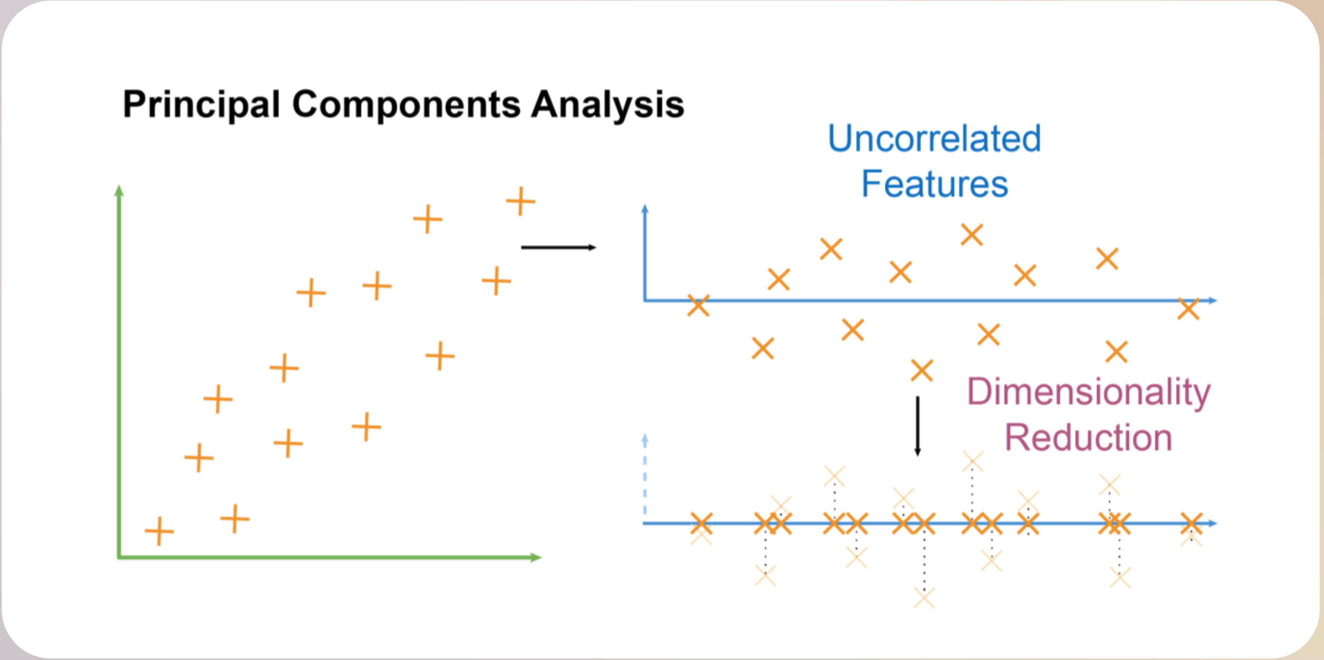
- Text data is often high-dimensional, sparse and complex.
- Computational requirements increase exponentially with the number of dimensions.
- It is hence necessary to condense this data into a more usable form.
- Vector dimensionality reduction in Natural Language Processing (NLP) is the process of reducing the number of features (dimensions) in a vector representation of text data.
- Dimensionality Reduction techniques include:
  - PCA
  - SVD

© 2023 Nanyang Technological University, Singapore. All Rights Reserved.

Principal Component Analysis (PCA):

PCA is a linear technique that finds orthogonal axes in the data that capture the most variance.

- PCA aims to project the data onto a lower-dimensional space while minimizing information loss.
- PCA works by transforming the original data to a new set of variables called the principal components (PCs), which are uncorrelated and can be ordered so that the first few PCs can retain most of the variation present in all the original variables of the dataset.

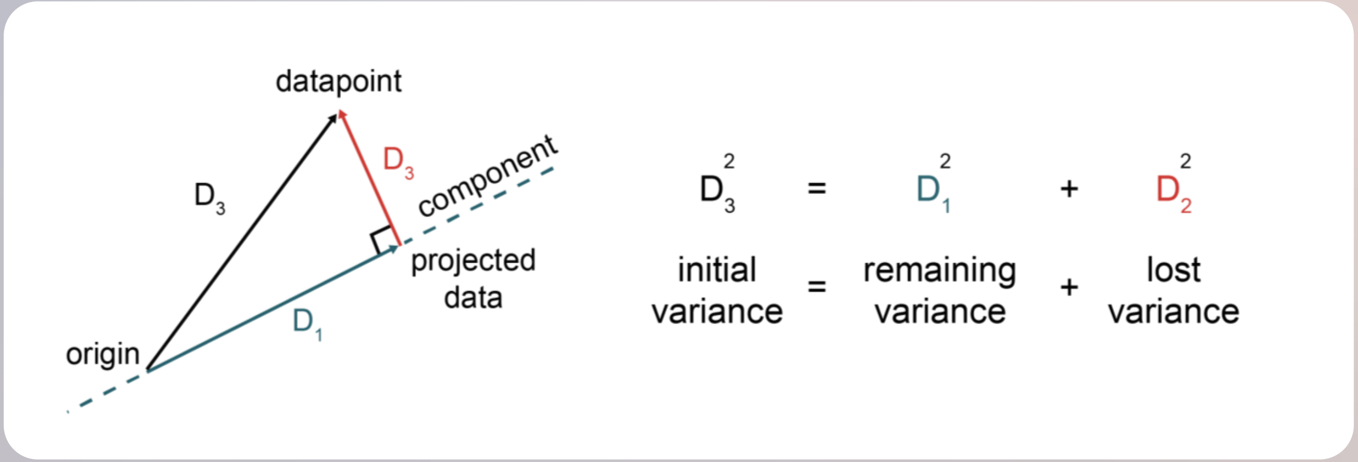


© 2023 Nanyang Technological University, Singapore. All Rights Reserved.

PCA

How it works:

- PCA aims to construct lines (2D), planes (3D) or hyperplanes (>3D) which are unit vectors orthogonal to the original features.
- To do this, we want to minimize the mean perpendicular distance from the PC line for all points through min $(\frac{1}{n}\sum_{i}^{n}(\boldsymbol{x}_i^T\boldsymbol{x}_i - (\boldsymbol{u}_1^T\boldsymbol{x}_i)^2))$
- The function reaches a minimum when $\boldsymbol{u\_1}$= the eigenvector of the covariance matrix of *x.*



The goal is to maximize $D_1$and minimize D_2while keeping D_3 constant.

© 2023 Nanyang Technological University, Singapore. All Rights Reserved.

PCA

How it works:

PCA can be done manually using the following steps:

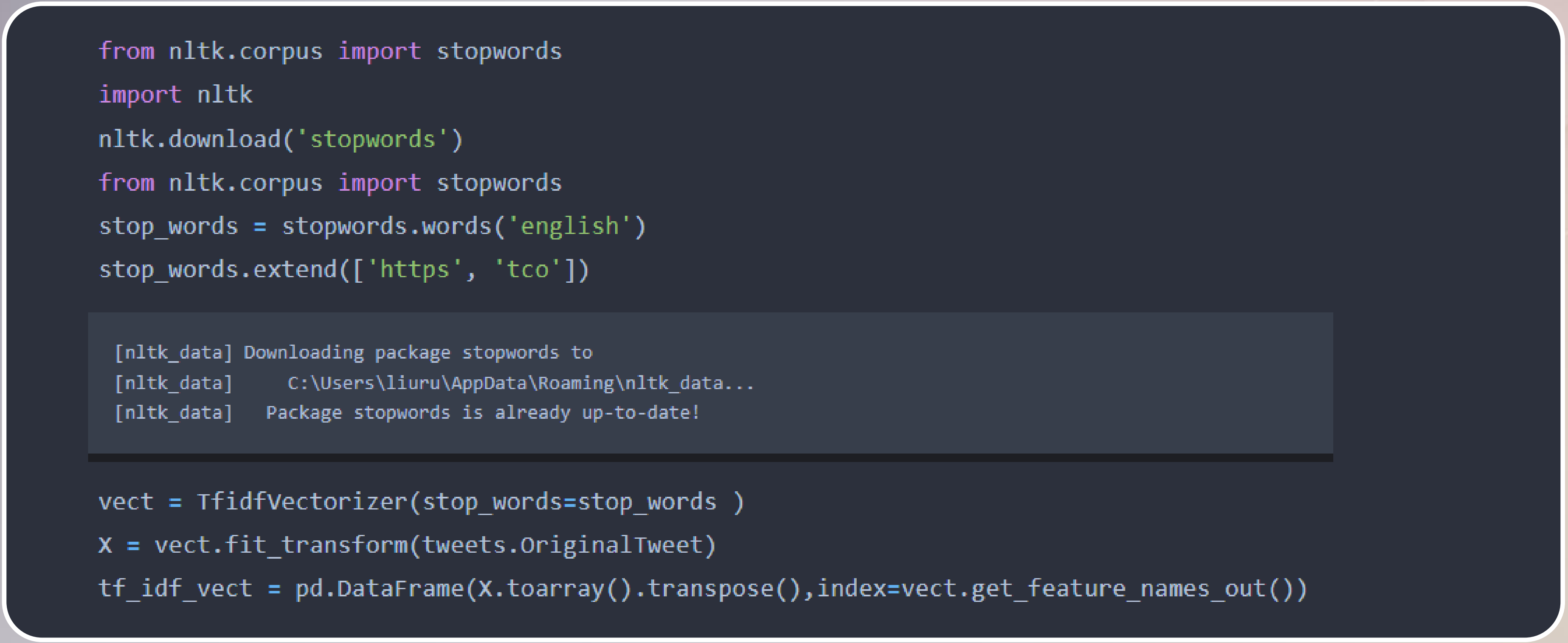
1. Standardise the data.
2. Compute the covariance matrix of the dataset.
3. Derive the corresponding eigenvalues and eigenvectors on the normalised dataset.
4. PCs can be calculated using the dot product of the eigenvectors and the standardised columns.

© 2023 Nanyang Technological University, Singapore. All Rights Reserved.

PCA

PCA in Scikit-Learn:

Using the same twitter dataset



```
from nltk.corpus import stopwords
import nltk
nltk.download('stopwords')
from nltk.corpus import stopwords
stop_words = stopwords.words('english')
stop_words.extend(['https', 'tco'])
```

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

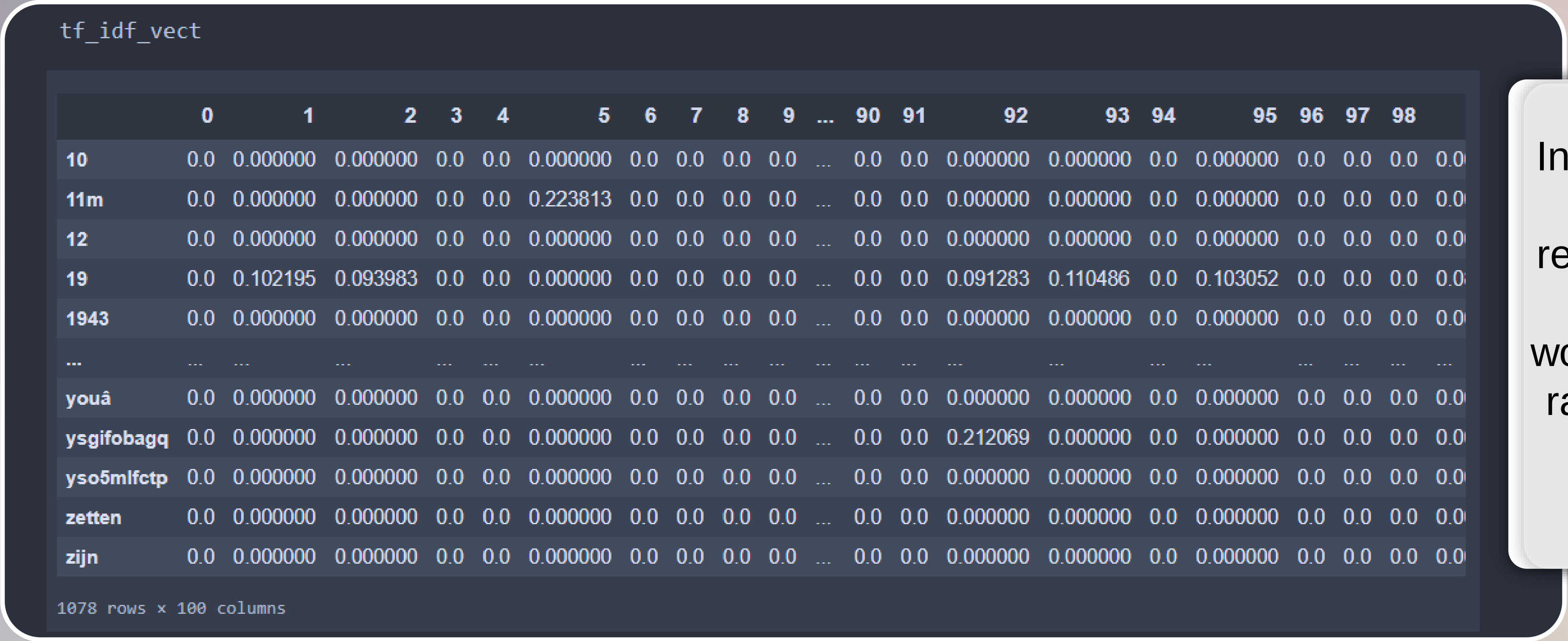
```
vect = TfidfVectorizer(stop_words=stop_words )
X = vect.fit_transform(tweets.OriginalTweet)
tf_idf_vect = pd.DataFrame(X.toarray().transpose(),index=vect.get_feature_names_out())
```

© 2023 Nanyang Technological University, Singapore. All Rights Reserved.

PCA

PCA in Scikit-Learn:

Apply TF- IDF on the data.



```
tf_idf_vect
```

```
             0         1         2    3    4         5    6    7    8    9  ...   90   91        92        93   94        95   96   97   98
10         0.0  0.000000  0.000000  0.0  0.0  0.000000  0.0  0.0  0.0  0.0  ...  0.0  0.0  0.000000  0.000000  0.0  0.000000  0.0  0.0  0.0  0.0
11m        0.0  0.000000  0.000000  0.0  0.0  0.223813  0.0  0.0  0.0  0.0  ...  0.0  0.0  0.000000  0.000000  0.0  0.000000  0.0  0.0  0.0  0.0
12         0.0  0.000000  0.000000  0.0  0.0  0.000000  0.0  0.0  0.0  0.0  ...  0.0  0.0  0.000000  0.000000  0.0  0.000000  0.0  0.0  0.0  0.0
19         0.0  0.102195  0.093983  0.0  0.0  0.000000  0.0  0.0  0.0  0.0  ...  0.0  0.0  0.091283  0.110486  0.0  0.103052  0.0  0.0  0.0  0.0
1943       0.0  0.000000  0.000000  0.0  0.0  0.000000  0.0  0.0  0.0  0.0  ...  0.0  0.0  0.000000  0.000000  0.0  0.000000  0.0  0.0  0.0  0.0
...        ...       ...       ...  ...  ...       ...  ...  ...  ...  ...  ...  ...  ...       ...       ...  ...       ...  ...  ...  ...  ...
youâ       0.0  0.000000  0.000000  0.0  0.0  0.000000  0.0  0.0  0.0  0.0  ...  0.0  0.0  0.000000  0.000000  0.0  0.000000  0.0  0.0  0.0  0.0
ysgifobagq 0.0  0.000000  0.000000  0.0  0.0  0.000000  0.0  0.0  0.0  0.0  ...  0.0  0.0  0.212069  0.000000  0.0  0.000000  0.0  0.0  0.0  0.0
yso5mlfctp 0.0  0.000000  0.000000  0.0  0.0  0.000000  0.0  0.0  0.0  0.0  ...  0.0  0.0  0.000000  0.000000  0.0  0.000000  0.0  0.0  0.0  0.0
zetten     0.0  0.000000  0.000000  0.0  0.0  0.000000  0.0  0.0  0.0  0.0  ...  0.0  0.0  0.000000  0.000000  0.0  0.000000  0.0  0.0  0.0  0.0
zijn       0.0  0.000000  0.000000  0.0  0.0  0.000000  0.0  0.0  0.0  0.0  ...  0.0  0.0  0.000000  0.000000  0.0  0.000000  0.0  0.0  0.0  0.0

1078 rows × 100 columns
```

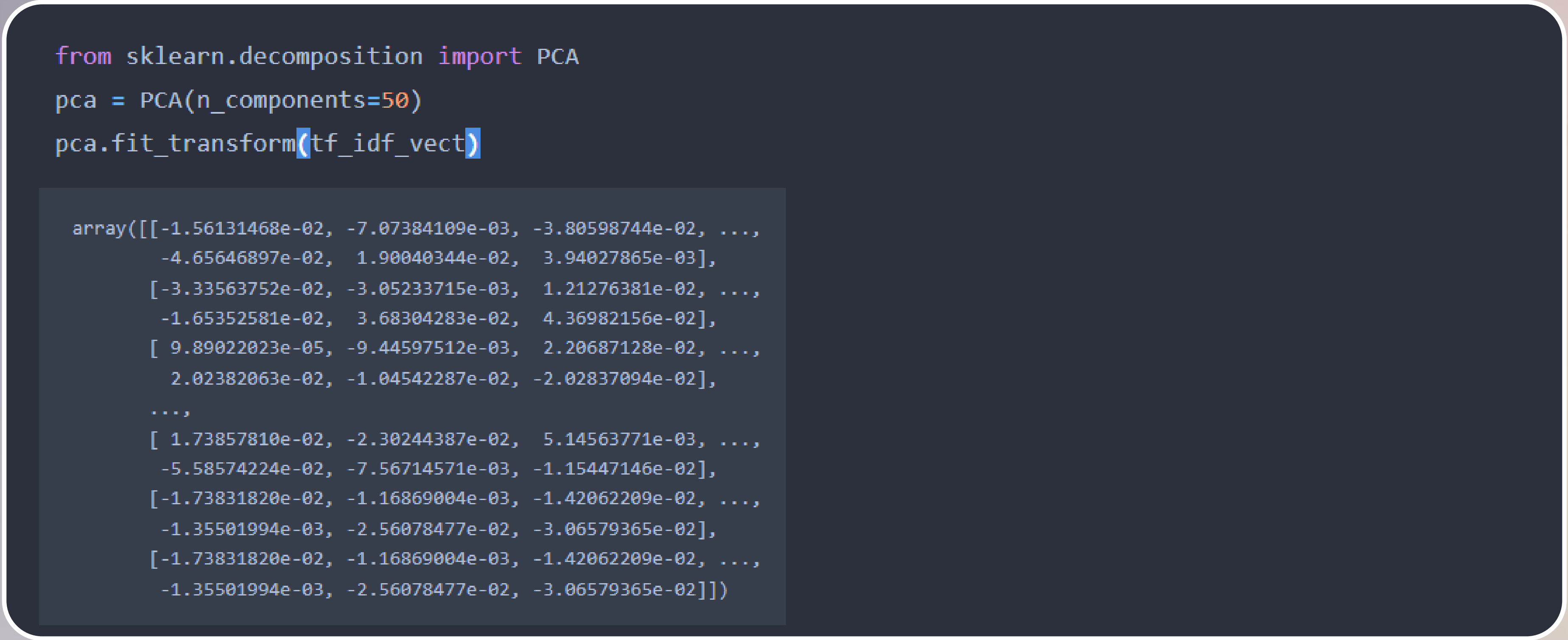
In this Data Frame, each column represents a tweet. Even with stop words removed, the raw TF-IDF vector still has 1078 dimensions.

© 2023 Nanyang Technological University, Singapore. All Rights Reserved.

PCA

PCA in Scikit-Learn:

Applying PCA to reduce the number of components in the data to 50.



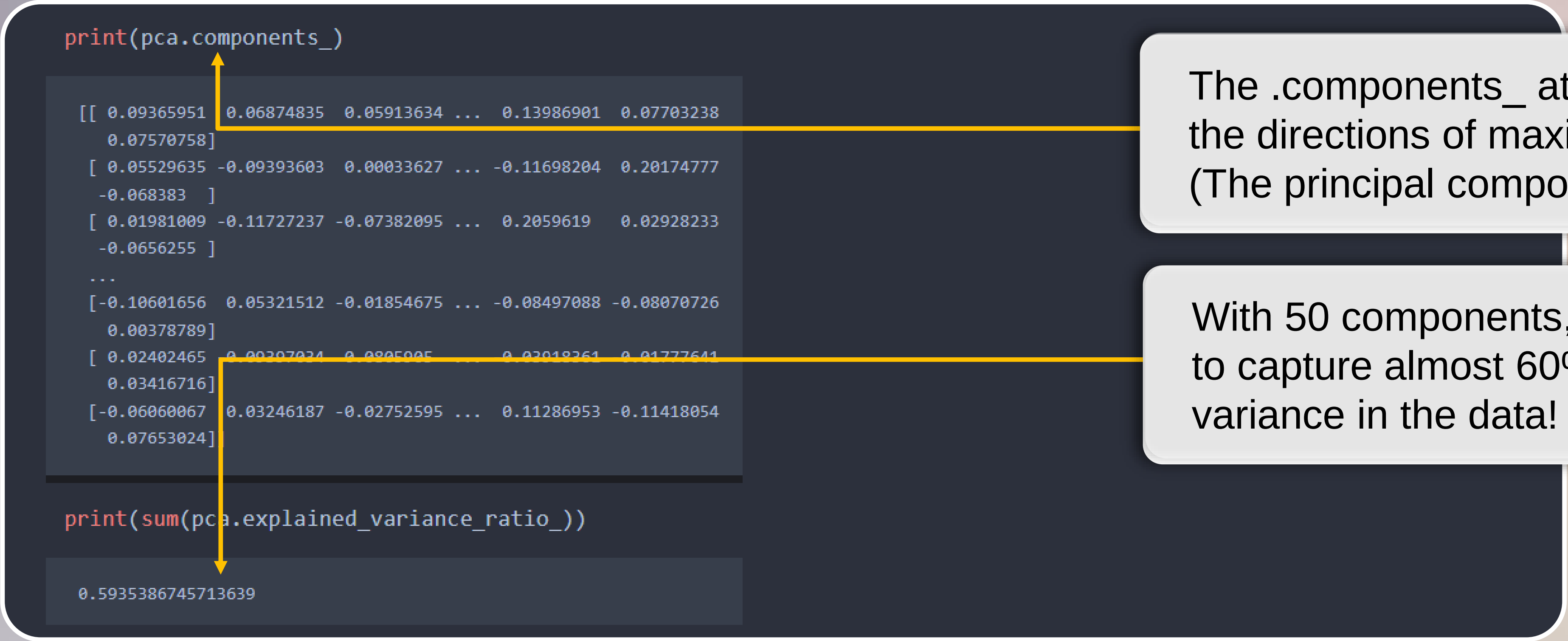
```python
from sklearn.decomposition import PCA
pca = PCA(n_components=50)
pca.fit_transform(tf_idf_vect)
```

```
array([[-1.56131468e-02, -7.07384109e-03, -3.80598744e-02, ...,
        -4.65646897e-02,  1.90040344e-02,  3.94027865e-03],
       [-3.33563752e-02, -3.05233715e-03,  1.21276381e-02, ...,
        -1.65352581e-02,  3.68304283e-02,  4.36982156e-02],
       [ 9.89022023e-05, -9.44597512e-03,  2.20687128e-02, ...,
         2.02382063e-02, -1.04542287e-02, -2.02837094e-02],
       ...,
       [ 1.73857810e-02, -2.30244387e-02,  5.14563771e-03, ...,
        -5.58574224e-02, -7.56714571e-03, -1.15447146e-02],
       [-1.73831820e-02, -1.16869004e-03, -1.42062209e-02, ...,
        -1.35501994e-03, -2.56078477e-02, -3.06579365e-02],
       [-1.73831820e-02, -1.16869004e-03, -1.42062209e-02, ...,
        -1.35501994e-03, -2.56078477e-02, -3.06579365e-02]])
```

© 2023 Nanyang Technological University, Singapore. All Rights Reserved.

PCA in Scikit-Learn:

Applying PCA to reduce the number of components in the data to 50.



```
print(pca.components_)
```

```
[[ 0.09365951  0.06874835  0.05913634 ...  0.13986901  0.07703238
   0.07570758]
 [ 0.05529635 -0.09393603  0.00033627 ... -0.11698204  0.20174777
  -0.068383  ]
 [ 0.01981009 -0.11727237 -0.07382095 ...  0.2059619   0.02928233
  -0.0656255 ]
 ...
 [-0.10601656  0.05321512 -0.01854675 ... -0.08497088 -0.08070726
   0.00378789]
 [ 0.02402465  0.00397034  0.0805905  ...  0.03018361  0.01777641
   0.03416716]
 [-0.06060067  0.03246187 -0.02752595 ...  0.11286953 -0.11418054
   0.07653024]]
```

```
print(sum(pca.explained_variance_ratio_))
```

```
0.5935386745713639
```

The .components_ attribute stores the directions of maximum variance (The principal components)

With 50 components, we were able to capture almost 60% of the variance in the data!

© 2023 Nanyang Technological University, Singapore. All Rights Reserved.

PCA

Advantages

- **Visualisation:**
  PCA is valuable for visualising high-dimensional data. It projects data onto lower-dimensional spaces, making it easier to plot and analyse in 2D or 3D.
- **Feature Engineering:**
  PCA can be used for feature engineering by creating new features as linear combinations of the original features. These new features may have improved discriminative power.

Limitations

- **Lack of Interpretability:**
  While PCA simplifies data representation, the new features (principal components) are linear combinations of the original features and may not have clear semantic meaning.
- **Linearity Assumption:**
  PCA assumes that relationships between variables are linear. If the data contains non-linear relationships, PCA may not capture them effectively.

© 2023 Nanyang Technological University, Singapore. All Rights Reserved.

SVD

Singular Value Decomposition (SVD)

SVD aims to factorise a given matrix into three separate matrices to uncover hidden patterns and structure within the data.

Given an input matrix (a term-document matrix for NLP), SVD decomposes the matrix into 3 separate matrices:

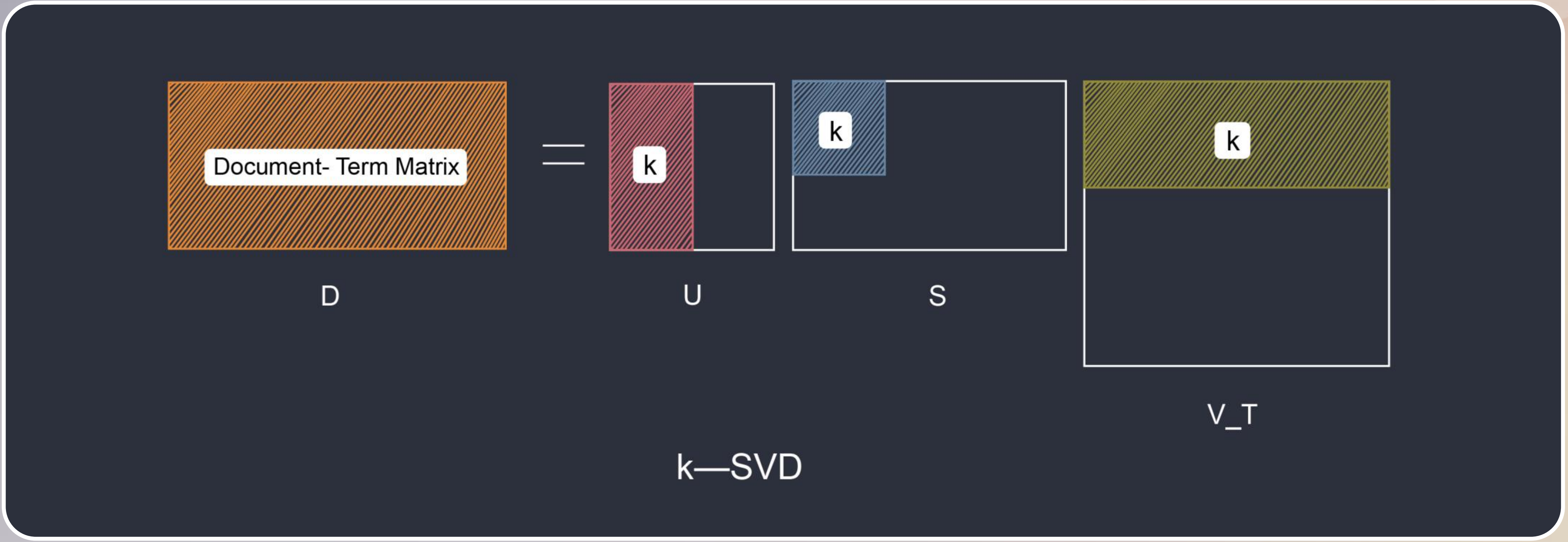
- $\boldsymbol{U}$: Left singular vectors matrix, representing relationships between rows in the original data. This often represents the relationship between terms and topics.
- $\boldsymbol{\Sigma}$: Diagonal matrix of singular values, capturing the importance of each singular vector (higher values indicate more significance).
- $\boldsymbol{V^T}$: Right singular vectors matrix, representing relationships between columns in the original data.
  This often represents the relationship between documents and terms.

© 2023 Nanyang Technological University, Singapore. All Rights Reserved.

SVD

Singular Value Decomposition (SVD)

A variation of SVD, k-SVD only takes into account the k largest singular values to approximate the data.

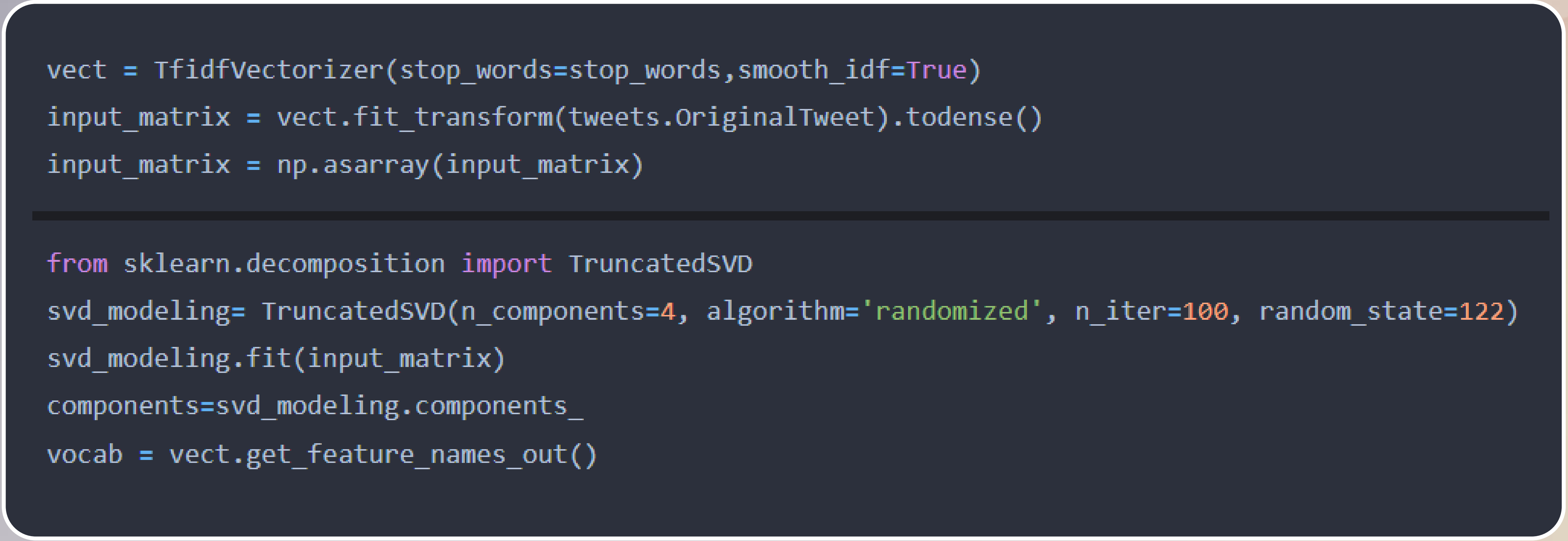


© 2023 Nanyang Technological University, Singapore. All Rights Reserved.

SVD

SVD in Scikit-Learn:

For this example, we will use the truncated SVD function.



```
vect = TfidfVectorizer(stop_words=stop_words,smooth_idf=True)
input_matrix = vect.fit_transform(tweets.OriginalTweet).todense()
input_matrix = np.asarray(input_matrix)
```

```
from sklearn.decomposition import TruncatedSVD
svd_modeling= TruncatedSVD(n_components=4, algorithm='randomized', n_iter=100, random_state=122)
svd_modeling.fit(input_matrix)
components=svd_modeling.components_
vocab = vect.get_feature_names_out()
```

© 2023 Nanyang Technological University, Singapore. All Rights Reserved.

SVD

SVD in Scikit-Learn:

Function to obtain the various topics from the list.

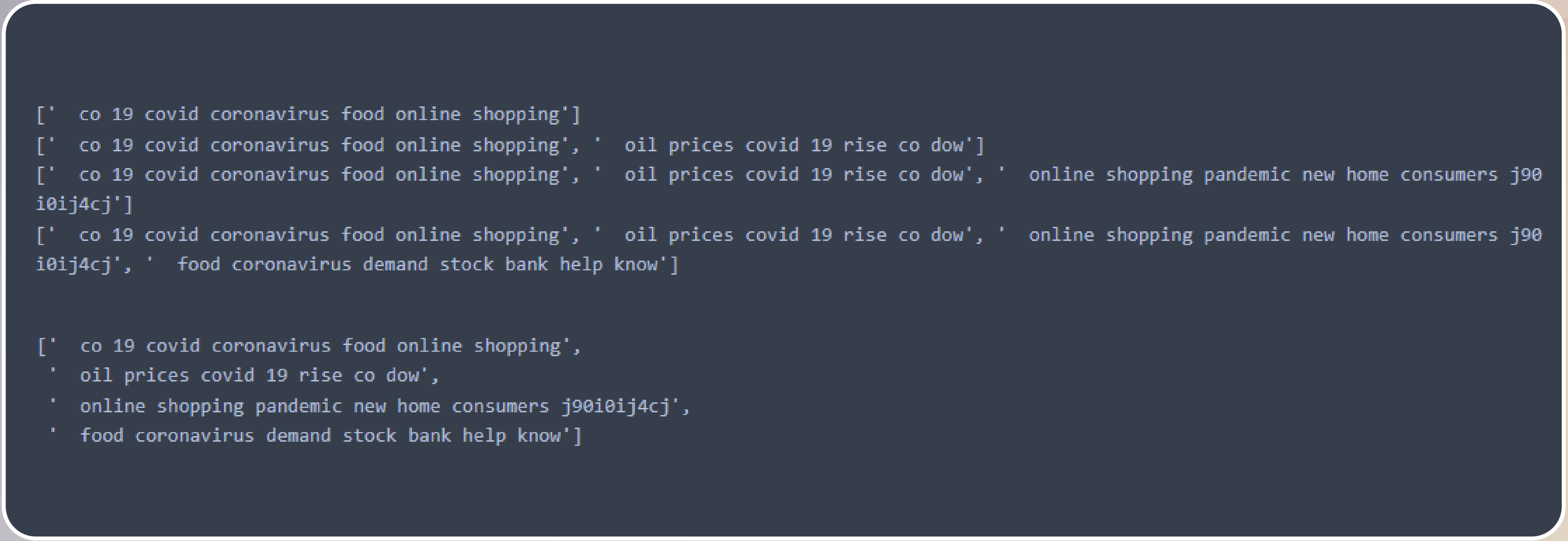
```
topic_word_list = []
def get_topics(components):
    for i, comp in enumerate(components):
        terms_comp = zip(vocab,comp)
        sorted_terms = sorted(terms_comp, key= lambda x:x[1], reverse=True)[:7]
        topic=" "
        for t in sorted_terms:
            topic= topic + ' ' + t[0]
        topic_word_list.append(topic)
        print(topic_word_list)
    return topic_word_list
get_topics(components)
```

© 2023 Nanyang Technological University, Singapore. All Rights Reserved.

SVD

SVD in Scikit-Learn:

Main topics distilled by SVD



```
['  co 19 covid coronavirus food online shopping']
['  co 19 covid coronavirus food online shopping', '  oil prices covid 19 rise co dow']
['  co 19 covid coronavirus food online shopping', '  oil prices covid 19 rise co dow', '  online shopping pandemic new home consumers j90
i0ij4cj']
['  co 19 covid coronavirus food online shopping', '  oil prices covid 19 rise co dow', '  online shopping pandemic new home consumers j90
i0ij4cj', '  food coronavirus demand stock bank help know']


['  co 19 covid coronavirus food online shopping',
 '  oil prices covid 19 rise co dow',
 '  online shopping pandemic new home consumers j90i0ij4cj',
 '  food coronavirus demand stock bank help know']
```

© 2023 Nanyang Technological University, Singapore. All Rights Reserved.

SVD

Advantages

- **Semantic Relationships:**
  SVD captures semantic relationships between terms and documents. The resulting low-dimensional representations often contain meaningful information about the underlying structure of the data.
- **Interpretability:**
  The reduced dimensions are often more interpretable, making it easier to understand and analyse the data. It aids in tasks like topic modeling and sentiment analysis.

Limitations

- **Scaling Sensitivity:**
  SVD is sensitive to the scaling of features. It is essential to standardise or scale features appropriately before applying SVD.
- **Choosing Dimensionality:**
  Deciding the appropriate number of dimensions (singular values) to retain can be subjective. An incorrect choice may lead to information loss or overfitting.

© 2023 Nanyang Technological University, Singapore. All Rights Reserved.

Summary

- **Term weighting schemes:**
  - Term weighting involve assigning numerical values to words in a document based on their importance, facilitating tasks like information retrieval and text analysis.
  - Helps prioritise relevant terms and improve the accuracy of various NLP tasks.

- **Topic Modeling:**
  - Topic modeling uncovers hidden thematic structures in large text collections, enabling automated discovery of topics and patterns within documents.
  - Used for tasks like document clustering, content recommendation, and understanding the content of unstructured text data.

© 2023 Nanyang Technological University, Singapore. All Rights Reserved.

Summary

- **Vector Dimensionality Reduction:**
  - Vector dimensionality reduction simplifies high-dimensional data representations, improving computational efficiency and interpretability while retaining essential information.
  - Transforms data into lower-dimensional spaces, aiding in visualisation, noise reduction, and data analysis.

© 2023 Nanyang Technological University, Singapore. All Rights Reserved.

No part of this video shall be filmed, recorded, downloaded, reproduced, distributed, republished or transmitted in any form or by any means without written approval from the University.