multiverse

multiverse

Natural Language Processing

Data Science Unit 4

Before we start...

- → Make sure you are comfortable
- → Have water and maybe a strong coffee handy
- → If you need a break... take it!
- → If you need a stretch – please go ahead!
- → Please mute yourselves if you are not talking
- → Have your video on at all times

...and let's get started!

In this session we will...

- → **Define** Natural Language Processing
- → **Explore** CountVectorizer
- → **Investigate** texts with Sentiment Analysis

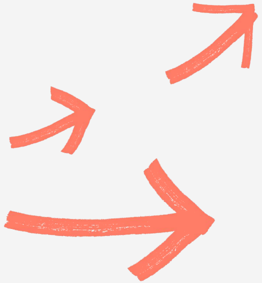
multiverse

NLP

NLP

Analysis
NLP techniques provide tools to allow us to understand and analyse large amounts of text. For example:

- Analyse the positivity/negativity of comments on different websites.
- Extract keywords from meeting notes and visualise how meeting topics change over time.



Vectorising for machine learning
When building a machine learning model, we typically must transform our data into numeric features. This process of transforming non-numeric data such as natural language into numeric features is called vectorisation. For example:

- Understanding related words. Using stemming, NLP lets us know that "swim", "swims", and "swimming" all refer to the same base word. This allows us to reduce the number of features used in our model.
- Identifying important and unique words. Using TF-IDF (term frequency-inverse document frequency), we can identify which words are most likely to be meaningful in a document.

What is it?

Using computers to process (analyse, understand, generate) natural human languages.

Making sense of human knowledge stored as unstructured text.

Building probabilistic models using data about a language.

Examples

High Level

Chatbots:
Understand natural language from the user and return intelligent responses.

- ➔ Api.ai

Information retrieval:
Find relevant results and similar results.

- ➔ Google

Information extraction:
Structured information from unstructured documents.

- ➔ Events from Gmail

Machine translation:
One language to another.

- ➔ Google Translate

Text simplification:
Preserve the meaning of text, but simplify the grammar and vocabulary.

- ➔ Rewordify
- ➔ Simple English Wikipedia

Predictive text input:
Faster or easier typing.

- ➔ Phrase completion application
- ➔ A much better application

Sentiment analysis:
Attitude of speaker.

- ➔ Hater News

Speech recognition and generation:
Speech-to-text, text-to-speech.

- ➔ Google's Web Speech API demo
- ➔ Vocalware Text-to-Speech demo

Low Level

Tokenization:
Breaking text into tokens (words, sentences, n-grams)

Stop-word removal:
a/an/the

Stemming and lemmatization:
root word

TF-IDF:
word importance

Part-of-speech tagging:
noun/verb/adjective

Named entity recognition:
person/organization /location

Spelling correction:
"New Yrok City"

Word sense disambiguation:
"buy a mouse"

Segmentation:
"New York City subway"

Language detection:
"translate this page"

Machine learning:
specialized models that work well with text

Challenges

Ambiguity:

- ➔ Hospitals Are Sued by 7 Foot Doctors
- ➔ Juvenile Court to Try Shooting Defendant
- ➔ Local High School Dropouts Cut in Half

Non-standard English:
text messages

Idioms:
"throw in the towel"

Newly coined words:
"retweet"

Tricky entity names:
"Where is A Bug's Life playing?"

World knowledge:
"Mary and Sue are sisters", "Mary and Sue are mothers"

multiverse

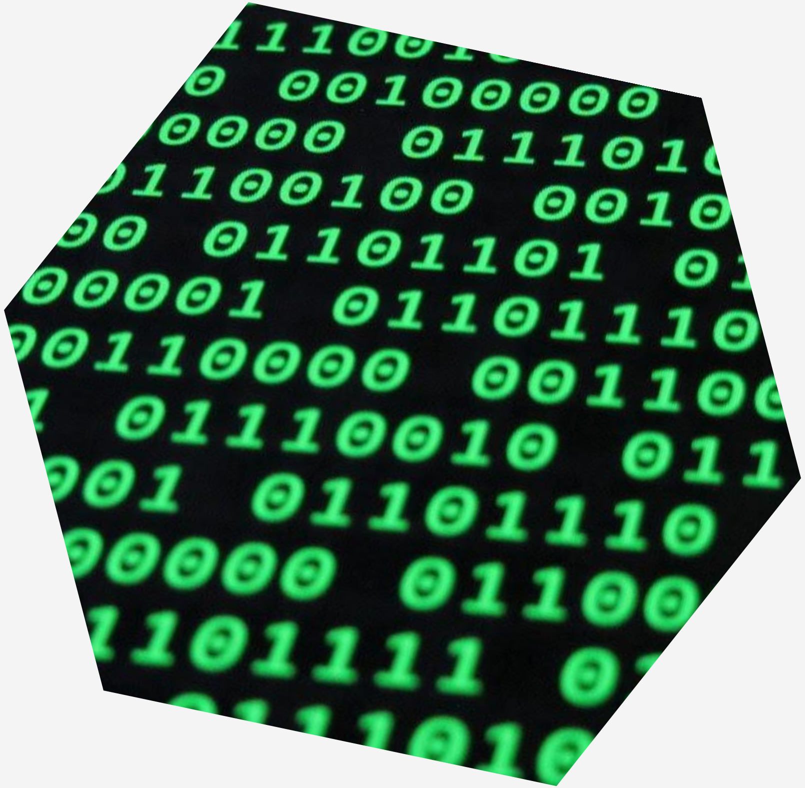
Text Classification

Text Classification

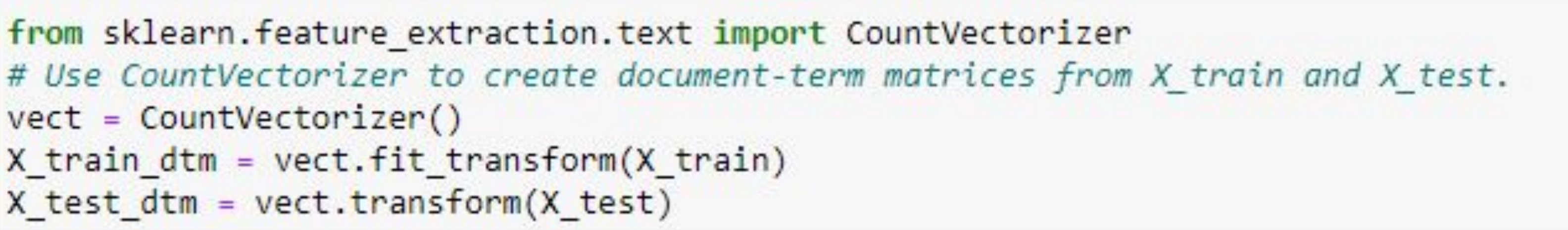
Text Classification

Text Classification

Text Classification

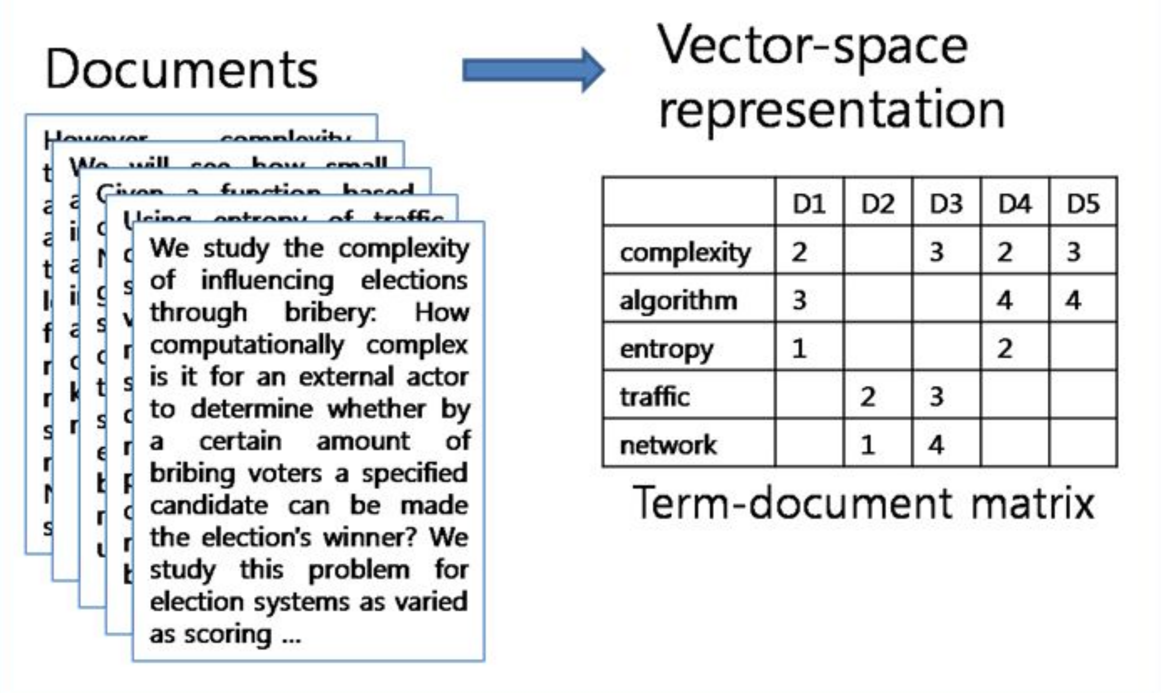


Text Classification



```python
from sklearn.feature_extraction.text import CountVectorizer
# Use CountVectorizer to create document-term matrices from X_train and X_test.
vect = CountVectorizer()
X_train_dtm = vect.fit_transform(X_train)
X_test_dtm = vect.transform(X_test)
```

Text Classification



We study the complexity of influencing elections through bribery: How computationally complex is it for an external actor to determine whether by a certain amount of bribing voters a specified candidate can be made the election's winner? We study this problem for election systems as varied as scoring ...

	D1	D2	D3	D4	D5
complexity	2		3	2	3
algorithm	3			4	4
entropy	1			2	
traffic		2	3		
network		1	4		

Term-document matrix

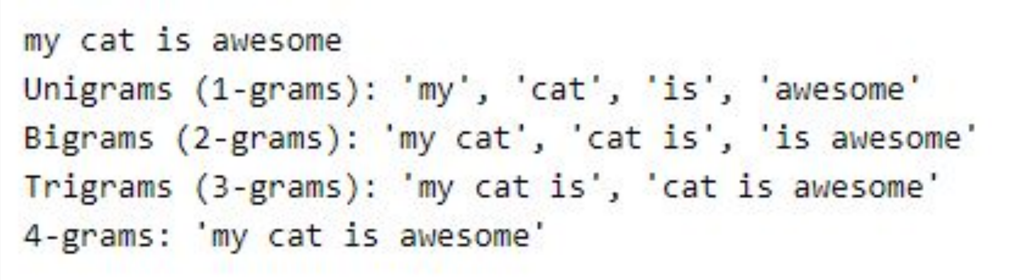
multiverse

Let’s Practice

multiverse

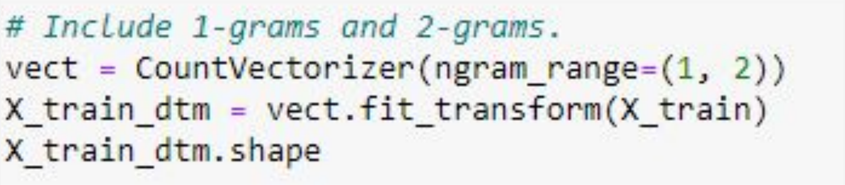
NGrams

NGrams



```
my cat is awesome
Unigrams (1-grams): 'my', 'cat', 'is', 'awesome'
Bigrams (2-grams): 'my cat', 'cat is', 'is awesome'
Trigrams (3-grams): 'my cat is', 'cat is awesome'
4-grams: 'my cat is awesome'
```

NGrams



```
# Include 1-grams and 2-grams.
vect = CountVectorizer(ngram_range=(1, 2))
X_train_dtm = vect.fit_transform(X_train)
X_train_dtm.shape
```

multiverse

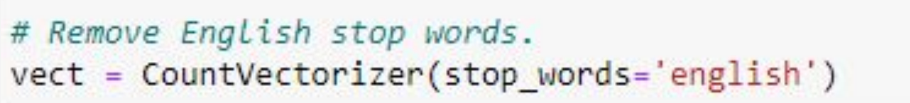
Let’s Practice

multiverse

Stop Words and Other Options

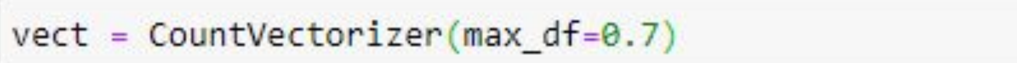
Stop Words

Stop Words

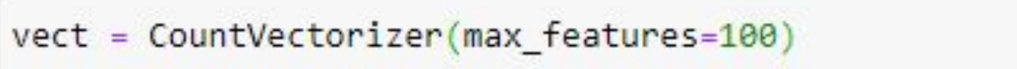


```
# Remove English stop words.
vect = CountVectorizer(stop_words='english')
```

MAX



```python
vect = CountVectorizer(max_df=0.7)
```



```python
vect = CountVectorizer(max_features=100)
```

multiverse

Let’s Practice

multiverse

Stemming and Lemmatization

Stemming

Lemmatization

Lemmatization	Stemming
shouted – shout	badly – bad
best – good	computing – comput
better – good	computed – comput
good – good	wipes – wip
wiping – wipe	wiped – wip
hidden – hide	wiping – wip

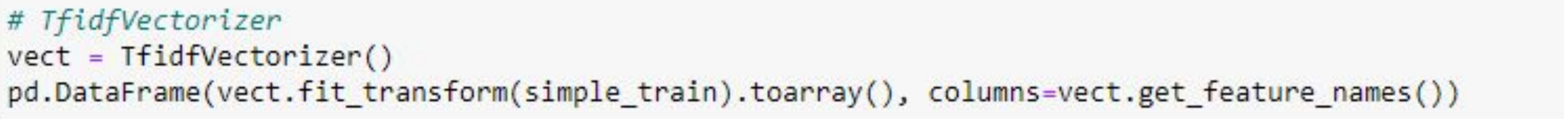
multiverse

Let’s Practice

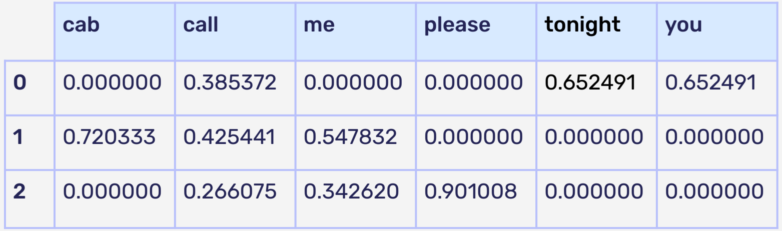
multiverse

TF-IDF

TF-IDF



```
# TfidfVectorizer
vect = TfidfVectorizer()
pd.DataFrame(vect.fit_transform(simple_train).toarray(), columns=vect.get_feature_names())
```



	cab	call	me	please	tonight	you
0	0.000000	0.385372	0.000000	0.000000	0.652491	0.652491
1	0.720333	0.425441	0.547832	0.000000	0.000000	0.000000
2	0.000000	0.266075	0.342620	0.901008	0.000000	0.000000

multiverse

Let’s Practice

multiverse

Sentiment Analysis

I Appreciate the Sentiment

I Appreciate the Sentiment

multiverse

Let’s Practice