Web and Social Media Analytics

Social media (exploration and topic modelling)

Dr Philip Worrall

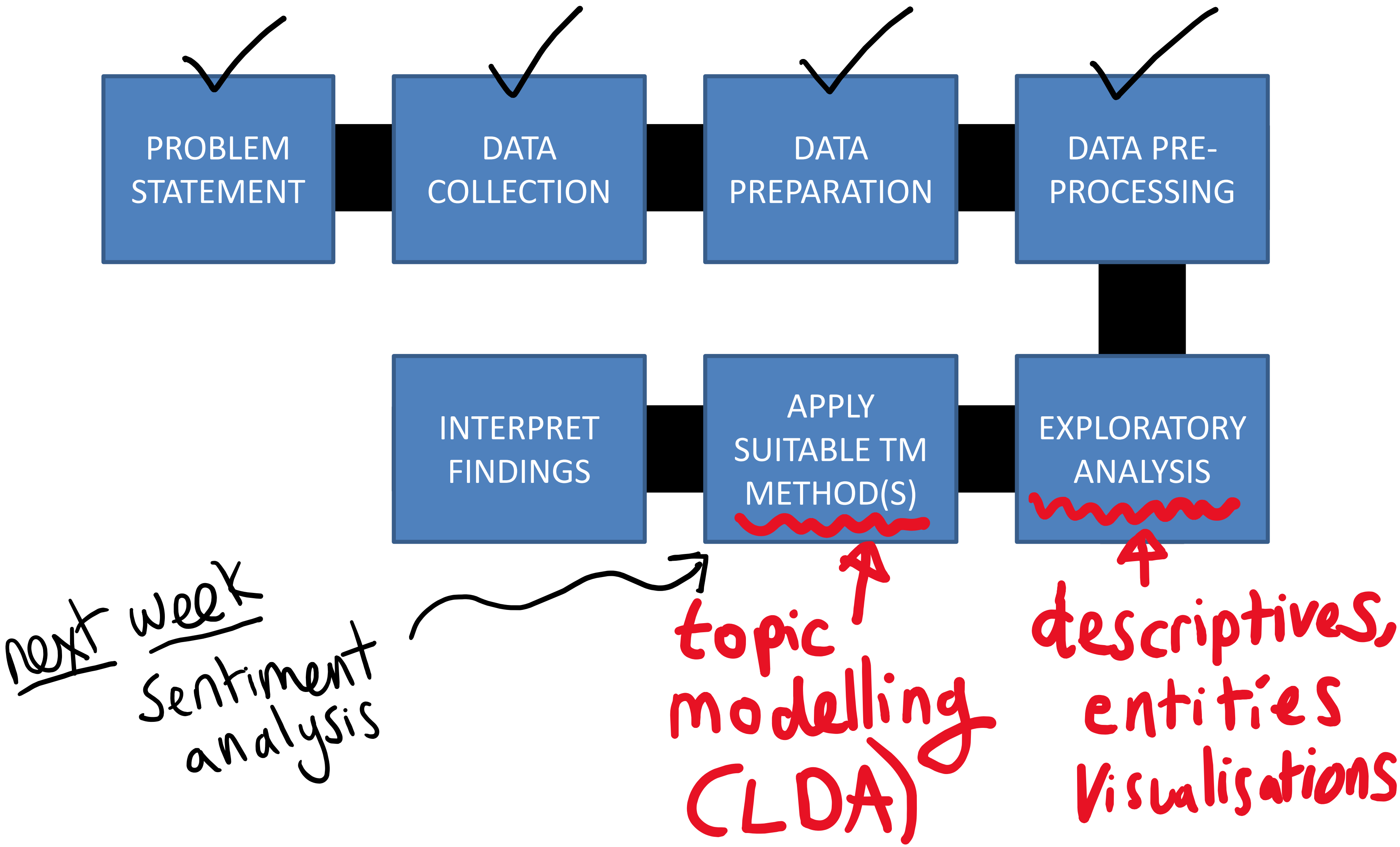
School of Computer Science and Engineering
115 New Cavendish Street
University of Westminster
London, W1W 6UW
worralph@westminster.ac.uk

LW10

Outline

- Recap of last week's material
- Data exploration with Pandas
  - Visualisations
  - Term Frequency (TF)
  - Word Clouds
  - Named Entity Recognition (NER/NPE)
- Topic Modelling
  - Latent Dirichlet Analysis (LDA)

The text mining process…



Sample data...

	text	cleaned_tweet	len	lang
0	I just realised that the episode of Futurama w...	i just realised that the episode of futurama w...	139	en
1	Gravity is not a good film\n	gravity is not a good film	26	en
2	Gravity , a very good film, better than what I...	gravity , a very good film, better than what i...	64	en
3	finally watched gravity, what an absolutely in...	finally watched gravity, what an absolutely in...	59	en
4	Gravity Review- 40%. A+ visuals, F script, arr...	gravity review- 40%. a+ visuals, f script, arr...	135	en
...	...	...	...	...
3522	I look forward to my graduation this year whic...	i look forward to my graduation this year whic...	139	en
3523	Oh it's on film night #Gravity http://t.co/la...	oh it's on film night #gravity	30	en
3524	Photo: elliotexplicit: Deleted scene from the ...	photo: elliotexplicit: deleted scene from the ...	81	en
3525	Cannot figure out how the film Gravity got an ...	cannot figure out how the film gravity got an ...	93	en
3526	Gravity....Harrison says its the worst film he...	gravity....harrison says its the worst film he...	106	en

3509 rows × 4 columns

from the completed wk9 exercise

Frequencies...

pd.Series

```
df["lang"].value_counts().head(10)
```

```
en    3425
sv      15
sw      15
no      12
nl       9
it       6
sk       4
fr       3
af       2
et       2
Name: lang, dtype: int64
```

```
df["lang"].value_counts().tail(10)
```

```
tl    1
es    1
id    1
sl    1
cy    1
pl    1
hu    1
lv    1
pt    1
fi    1
Name: lang, dtype: int64
```

Identifying commonly used languages

Visualising the frequencies…

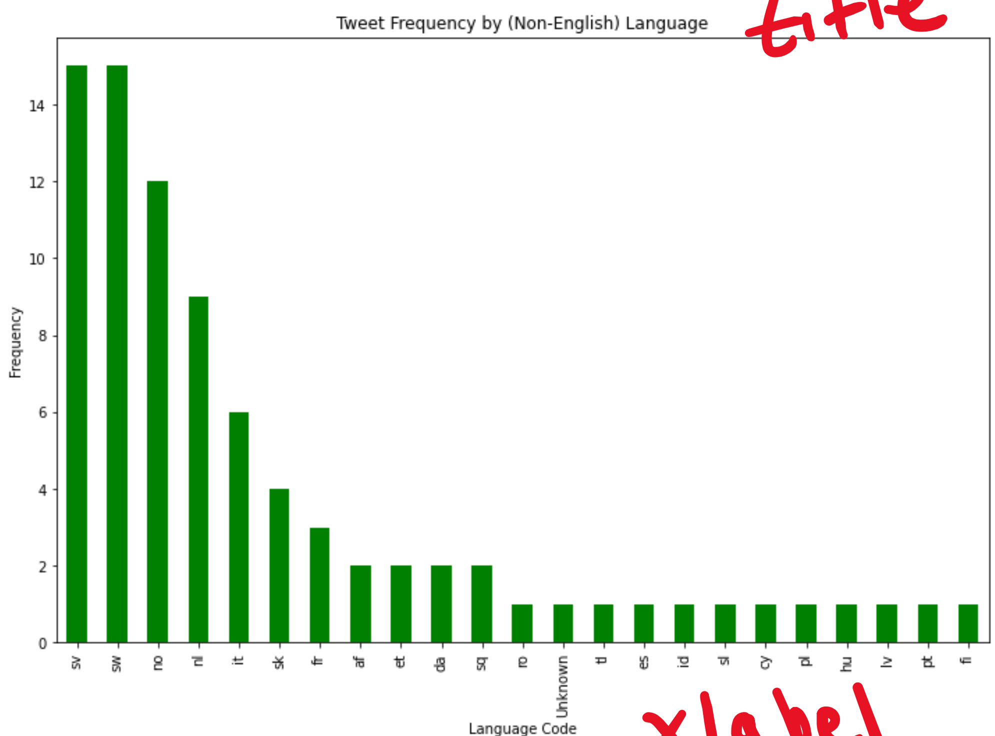
Vertical bar

green

```
df["lang"].value_counts().iloc[1:].plot(kind="bar",
                                         figsize=(12,8), color="g",
                                         title="Tweet Frequency by (Non-English) Language",
                                         xlabel="Language Code",
                                         ylabel="Frequency")
```

?

title

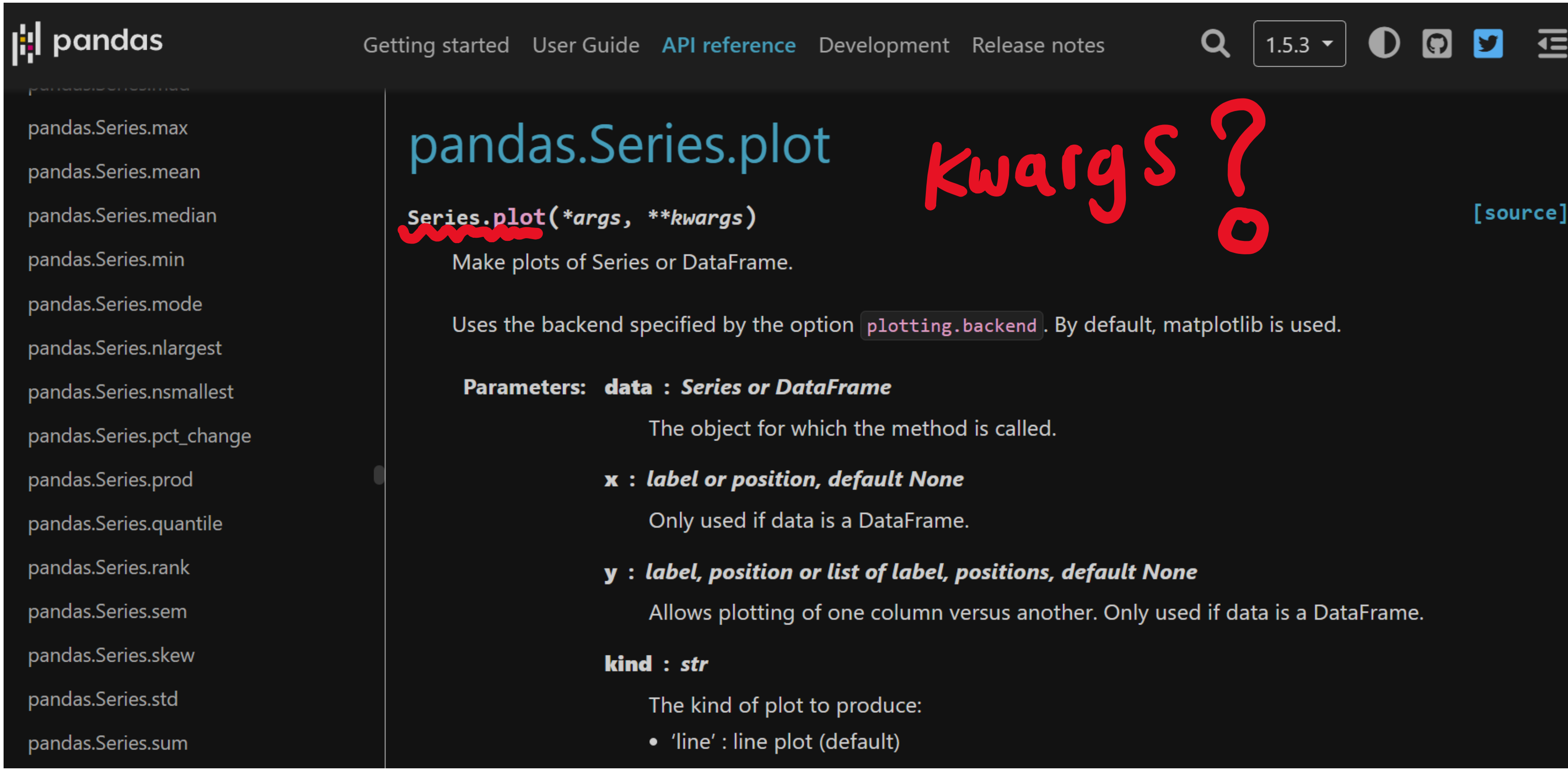


fontsize =10

ylabel

xlabel

Looking up the method signature…


pandas Getting started User Guide API reference Development Release notes 1.5.3

pandas.Series.max
pandas.Series.mean
pandas.Series.median
pandas.Series.min
pandas.Series.mode
pandas.Series.nlargest
pandas.Series.nsmallest
pandas.Series.pct_change
pandas.Series.prod
pandas.Series.quantile
pandas.Series.rank
pandas.Series.sem
pandas.Series.skew
pandas.Series.std
pandas.Series.sum

pandas.Series.plot

kwargs ?

`Series.plot(*args, **kwargs)` [source]

Make plots of Series or DataFrame.

Uses the backend specified by the option `plotting.backend`. By default, matplotlib is used.

Parameters: **data** : ***Series or DataFrame***

The object for which the method is called.

x : ***label or position, default None***

Only used if data is a DataFrame.

y : ***label, position or list of label, positions, default None***

Allows plotting of one column versus another. Only used if data is a DataFrame.

kind : ***str***

The kind of plot to produce:

- 'line' : line plot (default)

https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.Series.plot.html

Descriptives...

```
df["len"].describe()
```

```
count    3509.000000
mean       81.949273
std        35.475235
min        10.000000
25%        52.000000
50%        81.000000
75%       113.000000
max       148.000000
Name: len, dtype: float64
```

rows

```
df["len"].mode()
```

```
0    113
Name: len, dtype: int64
```

most common length

Grouping...

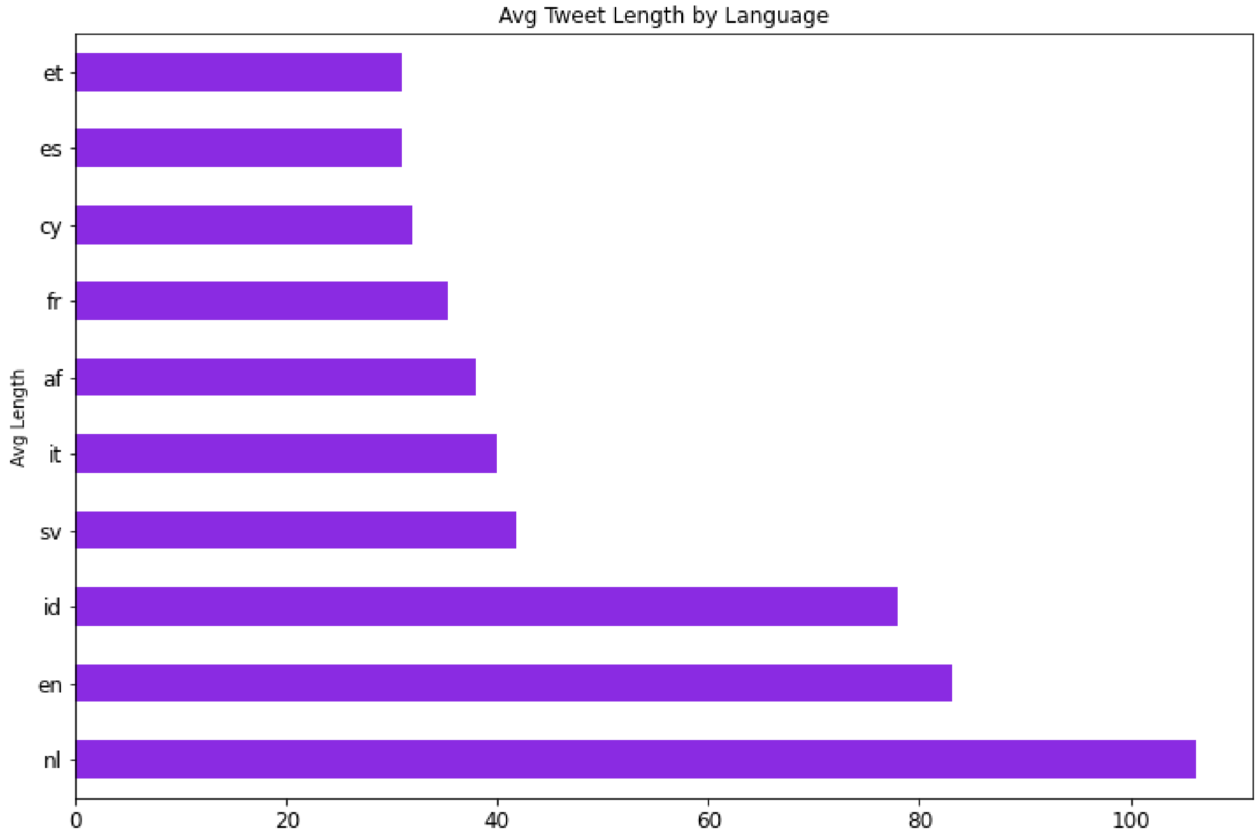
```
df.groupby("lang")["len"].mean().sort_values(ascending=False).head(10)
```

```
lang
nl     106.222222
en      83.078248
id      78.000000
sv      41.733333
it      40.000000
af      38.000000
fr      35.333333
cy      32.000000
es      31.000000
et      31.000000
Name: len, dtype: float64
```

by date, user... etc

Group, select, sort and plot…

```
df.groupby("lang")["len"].mean().sort_values(ascending=False).head(10).plot(
    kind="barh", figsize=(12,8), title="Avg Tweet Length by Language",
    xlabel="Avg Length", ylabel="Lanuage", fontsize=12, color="blueviolet")
```



Term Frequency - TF

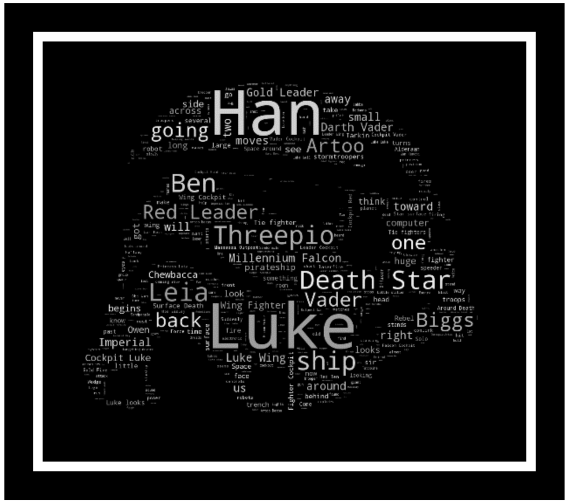
```
from collections import Counter
word_counter = Counter()
for row in df.to_dict("records"):
  word_counter.update(row["cleaned_tweet"].split())

word_counter.most_common(15)
```

```
[('film', 2297),
 ('gravity', 2121),
 ('the', 2094),
 ('a', 1306),
 ('is', 1117),
 ('of', 718),
 ('rt', 638),
 ('to', 622),
 ('on', 558),
 ('in', 557),
 ('for', 530),
 ('an', 500),
 ('cameraman', 489),
 ('i', 485),
 ('#gravity', 407)]
```

A list of ....

Are any pre-processing steps missing?

Word Clouds...



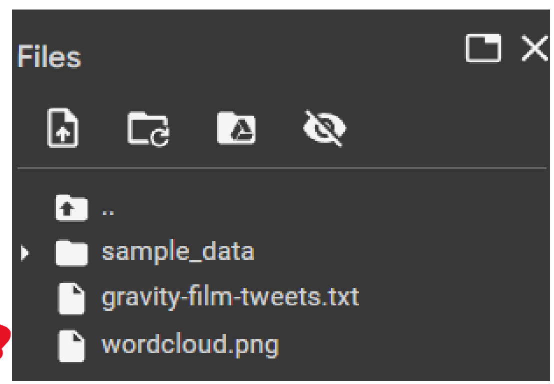
github.com/amueller/word_cloud/blob/master/examples/

Generating a Word Cloud…

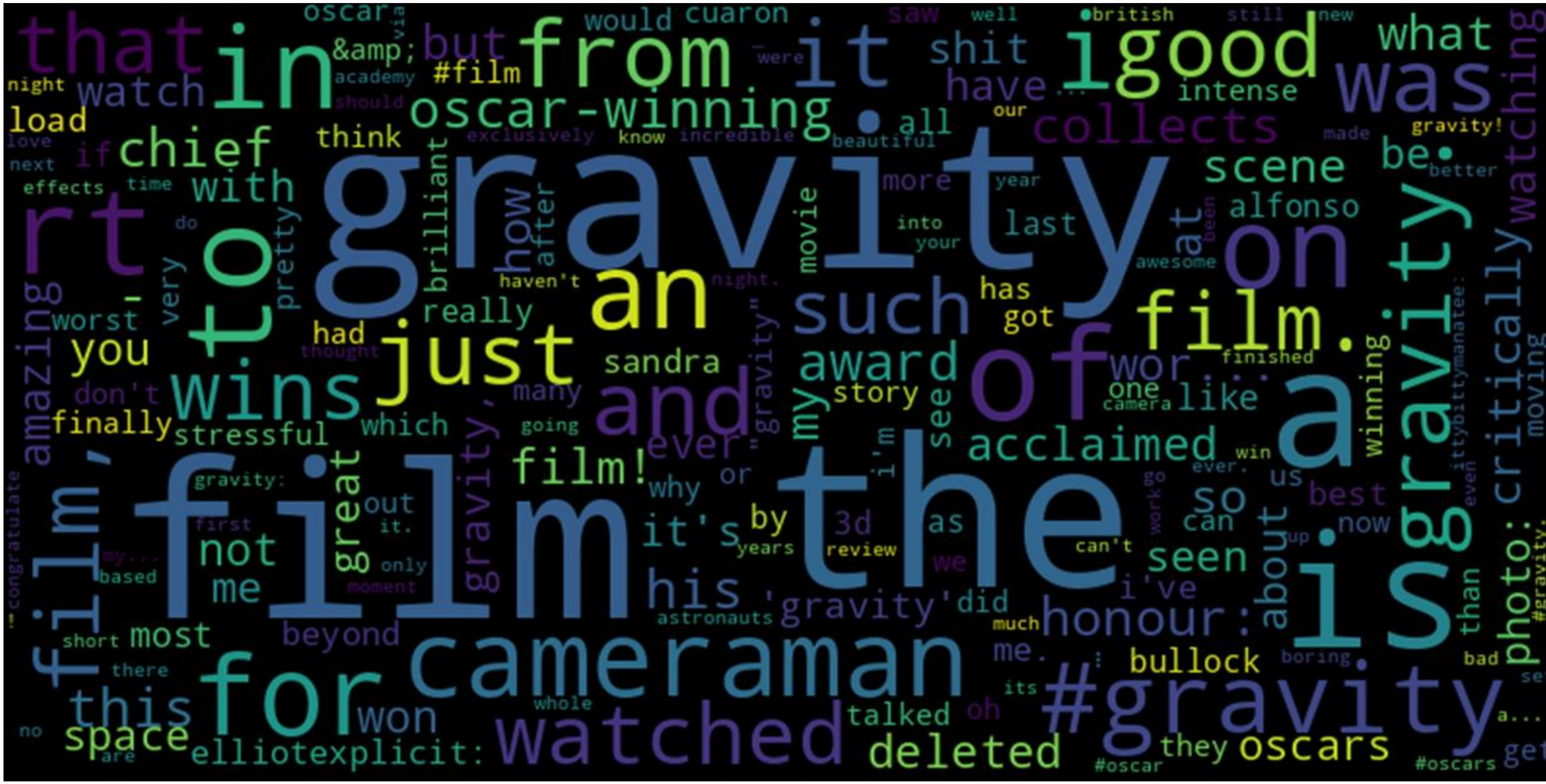
[Python WordCloud package]

```
from wordcloud import WordCloud

cloud = WordCloud(width=800, height=400)
cloud.generate_from_frequencies(dict(word_counter.most_common(200)))
image = cloud.to_image()
image.save("wordcloud.png")
```

saved in the session files



Simple Word Cloud…



Noun Phrase Extraction...

- Noun phrase extraction (NPE) is a special case of more general form entity recognition.
- Pick out and classify named entities
  - People
  - Organisations
  - Places or Addresses

* Who is this tweet referring to?

An entity is any real world object.

* An email address
* A zipcode

Noun Phrase Extraction...

- Speech is interpreted by the reader

Pragmatics

I lvoe cmoing to tihs calss

- Non-humans (machines, pets) find it especially hard to understand complex sentences.
- The same idea can be phrased in several ways.
- The context of a particular word is important.

Noun Phrase Extraction...

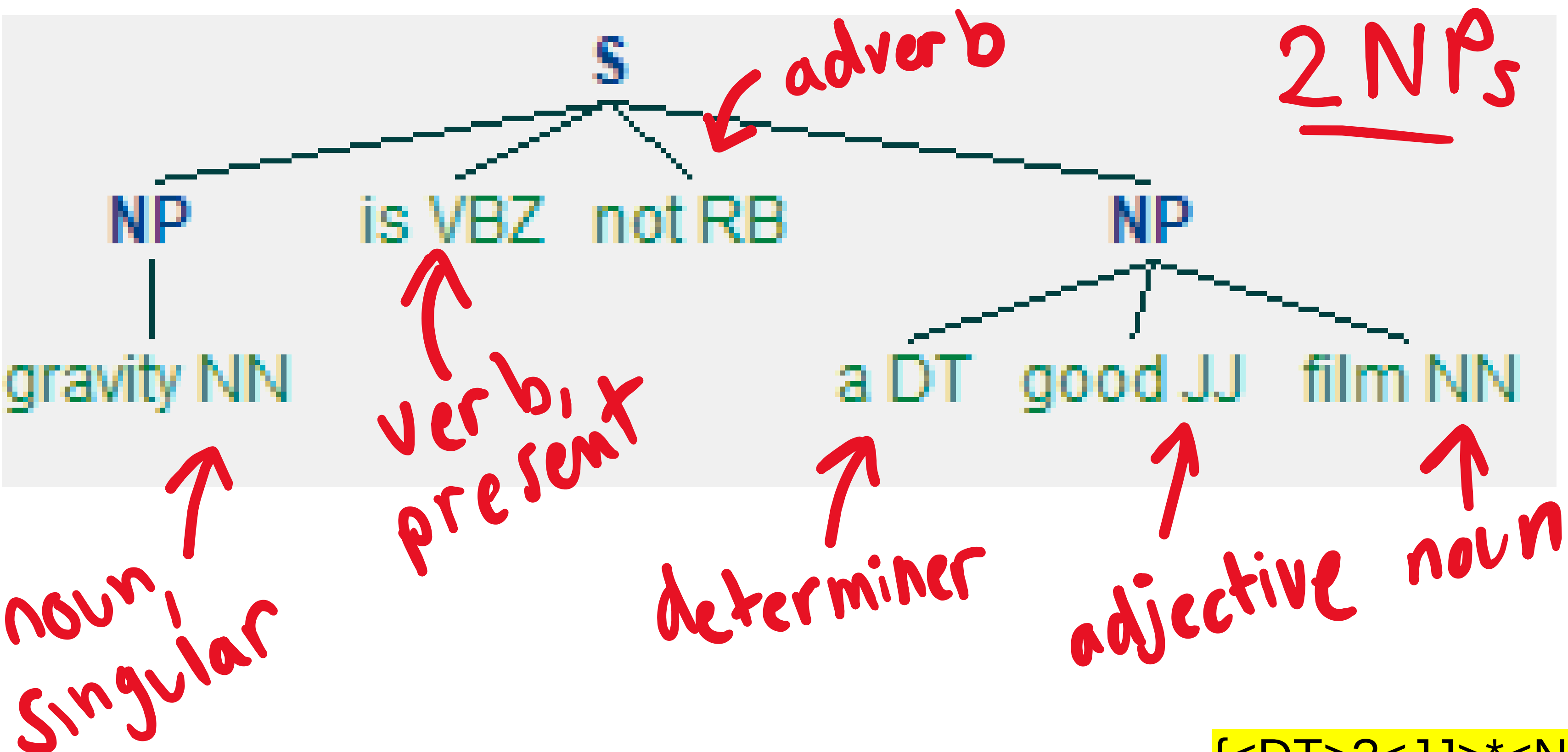
Modern search engines still struggle to distinguish between these two very similar concepts..

- A book about children
- A book written by children

Do we agree these two statements are fundamentally different?

PoS Tagging...

```
tweet = "gravity is not a good film"
```



{<DT>?<JJ>*<NN>}

[Parts of Speech Tags used in NLTK]

Extracting NPs with TextBlob…

```
import nltk
nltk.download('brown')
nltk.download('punkt')

from collections import Counter
from textblob import TextBlob

noun_counter = Counter()
for tweet in df["cleaned_tweet"].to_list():
  blob = TextBlob(tweet)
  noun_counter.update(blob.noun_phrases)

for np in noun_counter.most_common(10):
  print(np)
```

Samples of US lang

tokenizer from LW9

The extracted NPs…

```
('film gravity', 483)
('# gravity', 284)
('chief cameraman', 245)
('gravity cameraman', 221)
('wor ...', 182)
('good film', 157)
('shit film', 113)
('rt gravity', 110)
('great film', 92)
('amazing film', 88)
```

Topic Modelling...

- Assumes that a text corpus contains different themes/dimensions of discussion.
- In the structured world these may manifest as sections or chapters of a book.
- We are interested in discovering these hidden topics or categories of conversation.

We have little/no idea of what these topics might be

The LDA topic allocation model...

we choose
K

abstract

Latent Dirichlet allocation

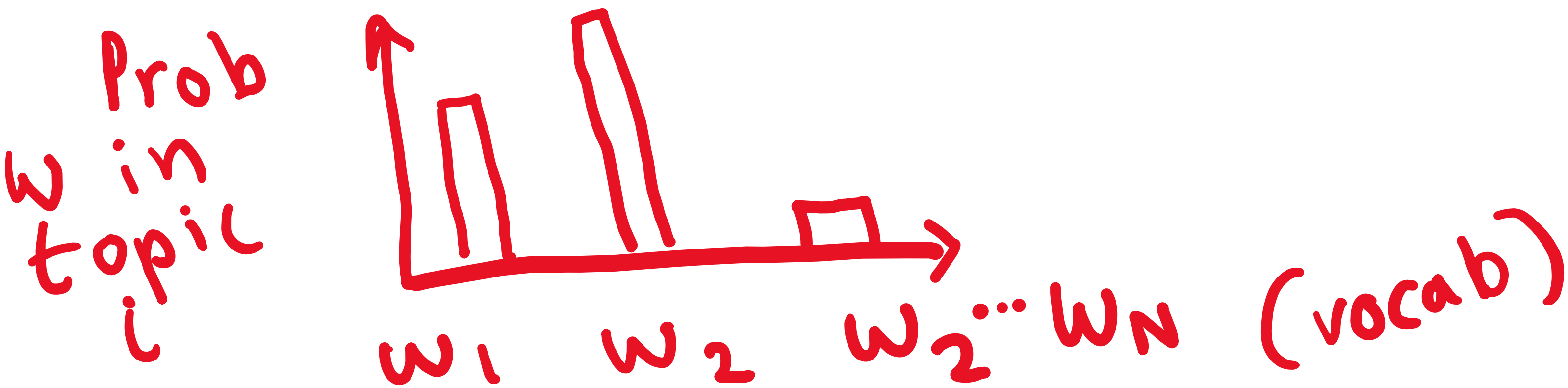
LDA allocation is a type of generative probabilistic topic model that can be used to identify a set of k topics, where each topic is defined by a probability of containing certain words and each phrase/document is a probabilistic mixture of all topics. It's a generative approach because it creates a full probabilistic model of the entire dataset.

di-rich-let

The LDA topic allocation model...

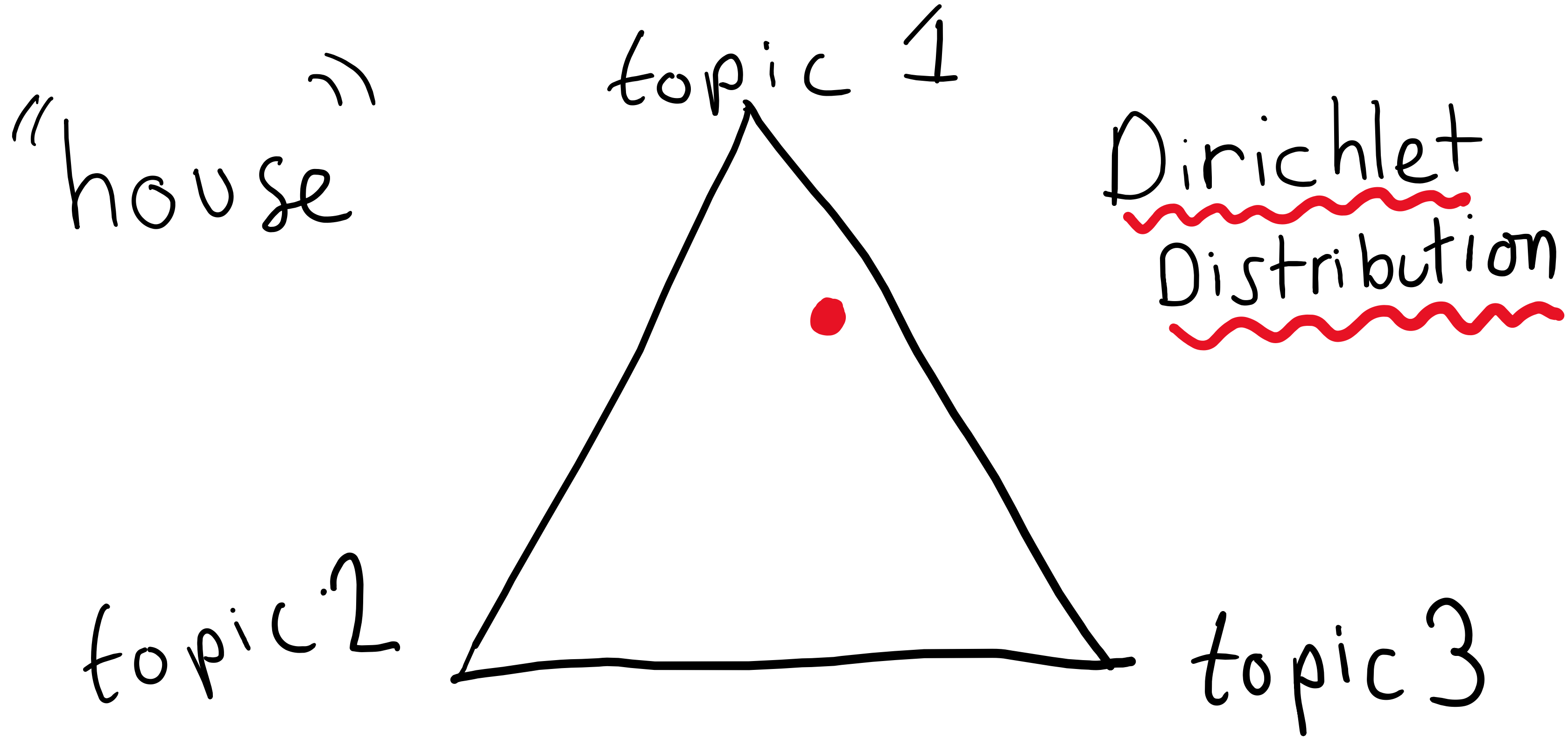
Topic

Each topic is essentially a probability distribution over different terms. Terms in the same topic have a high probability of occurring together in the same piece of text. Every topic contains the probability of every word occurring in it, even if its very small.



Intuition behind the LDA…

- Every word belongs to each topic but with a different probability

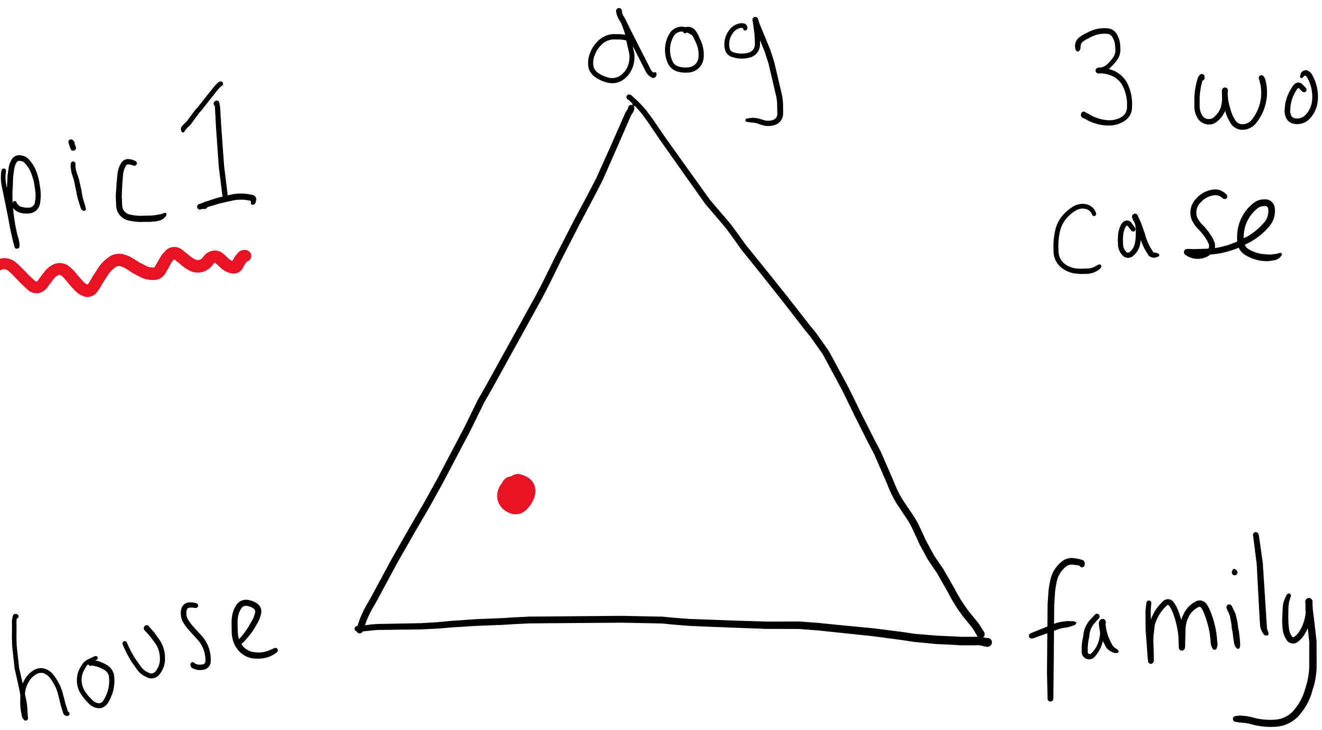


Intuition behind the LDA…

- By implication, every topic contains every word albeit with a different probability

topic 1

3 word case



Intuition behind the LDA…

- The objective of the LDA is to place words in the same topic that have the highest probability of occurring together in the same document (tweet, submission)

- We should almost be able to take a topic at random and produce a document that we would find in the underlying dataset.

Gibbs Sampling...

1. Choose the number of topics (k).
2. Collect together all words in the collection.
3. Randomly assign all words across all topics.
4. Take a sample document. Update the probability distribution of words across the topics.
5. Repeat step (4) many thousands of times.
6. The words begin to resemble the same probability distributions over topics as seen in the underlying dataset.

[Blei, M., Ng, A., and Jordon, M (2003)]

Check the LDA.xlsx (LW10) for a step by step example…

Performing the LDA in Python…

```python
import gensim
import gensim.corpora as corpora
from pprint import pprint

documents = [t.split() for t in df["cleaned_tweet"]]
vocab = corpora.Dictionary(documents)
corpus = [vocab.doc2bow(text) for text in documents]
```

bag of words

```python
num_topics = 5
lda = gensim.models.LdaMulticore(corpus=corpus,
                                 id2word=vocab,
                                 num_topics=num_topics)
pprint(lda.print_topics())
```

Interpreting the abstract topics…

```
[(0,
  '0.072*"gravity" + 0.065*"film" + 0.039*"rt" + 0.017*"oscars" + '
  '0.016*"alfonso" + 0.016*"cuaron" + 0.015*"moving" + 0.014*"us" + '
  '0.014*"exclusively" + 0.014*"talked"'),
 (1,
  '0.097*"gravity" + 0.086*"film" + 0.020*"rt" + 0.017*"watch" + '
  '0.013*"winning" + 0.013*"oscar" + 0.012*"good" + 0.011*"cameraman" + '
  '0.009*"award" + 0.009*"seen"'),
 (2,
  '0.099*"gravity" + 0.098*"film" + 0.024*"good" + 0.020*"watched" + '
  '0.019*"rt" + 0.011*"acclaimed" + 0.011*"scene" + 0.011*"deleted" + '
  '0.011*"seen" + 0.010*"critically"'),
 (3,
  '0.133*"gravity" + 0.118*"film" + 0.048*"cameraman" + 0.025*"winning" + '
  '0.025*"oscar" + 0.024*"rt" + 0.024*"honour" + 0.023*"collects" + '
  '0.023*"chief" + 0.023*"wins"'),
 (4,
  '0.123*"gravity" + 0.101*"film" + 0.019*"cameraman" + 0.014*"scene" + '
  '0.014*"oscar" + 0.014*"critically" + 0.013*"deleted" + 0.013*"acclaimed" + '
  '0.012*"wins" + 0.012*"watching"')]
```

Perplexity

Assignment of tweets to topics...

```
pd.DataFrame(lda.get_document_topics(corpus))
```

	0	1	2	3	4
0	(0, 0.014569665)	(1, 0.1462009)	(2, 0.014693424)	(3, 0.8098446)	(4, 0.014691367)
1	(0, 0.050984886)	(1, 0.050601944)	(2, 0.050694432)	(3, 0.79670066)	(4, 0.05101808)
2	(0, 0.033880554)	(1, 0.033862494)	(2, 0.033825804)	(3, 0.86433977)	(4, 0.034091346)
3	(0, 0.029422835)	(1, 0.029013326)	(2, 0.029403228)	(3, 0.029321395)	(4, 0.8828392)
4	(0, 0.5019086)	(1, 0.013137137)	(2, 0.012850134)	(3, 0.45935175)	(4, 0.012752425)
...	...	...	...	...	...
3503	(0, 0.015565116)	(1, 0.015689837)	(2, 0.015574233)	(3, 0.9376548)	(4, 0.015516008)
3504	(0, 0.040806543)	(1, 0.04073839)	(2, 0.040778127)	(3, 0.041077137)	(4, 0.83659977)
3505	(0, 0.022415487)	(1, 0.02238967)	(2, 0.022322023)	(3, 0.02248399)	(4, 0.9103888)
3506	(0, 0.16032347)	(1, 0.02071863)	(2, 0.020385865)	(3, 0.77811027)	(4, 0.020461746)
3507	(0, 0.01686096)	(1, 0.016771404)	(2, 0.016738234)	(3, 0.017089387)	(4, 0.93254)

3508 rows × 5 columns

In Summary

- Following the pre-processing stage, data exploration is often carried out to increase understanding of the dataset and its characteristics.
- Through exploration we may identify features of the data that may need to be dealt with before continuing the modelling process.
- In addition to descriptive statistics of important fields and word frequency distributions, we can also visualise data in Pandas using the plot() function and create text specific representations (world clouds)
- Noun phrase extraction is concerned with identifying entities present in the dataset and can be useful to understand who or what is being talked about.
- Topic models are one potential way in which abstract themes or topics of conversation can be extracted from the dataset.
- In the LDA model, a topic consists of a probability distribution of words.
- Each document (tweet) has a probability of being associated with all topics.
- For a particular document, the topic with the highest probability is known as the dominant topic.

End