

# Web and Social Media Analytics

Social media (exploration and topic modelling)

Dr Philip Worrall

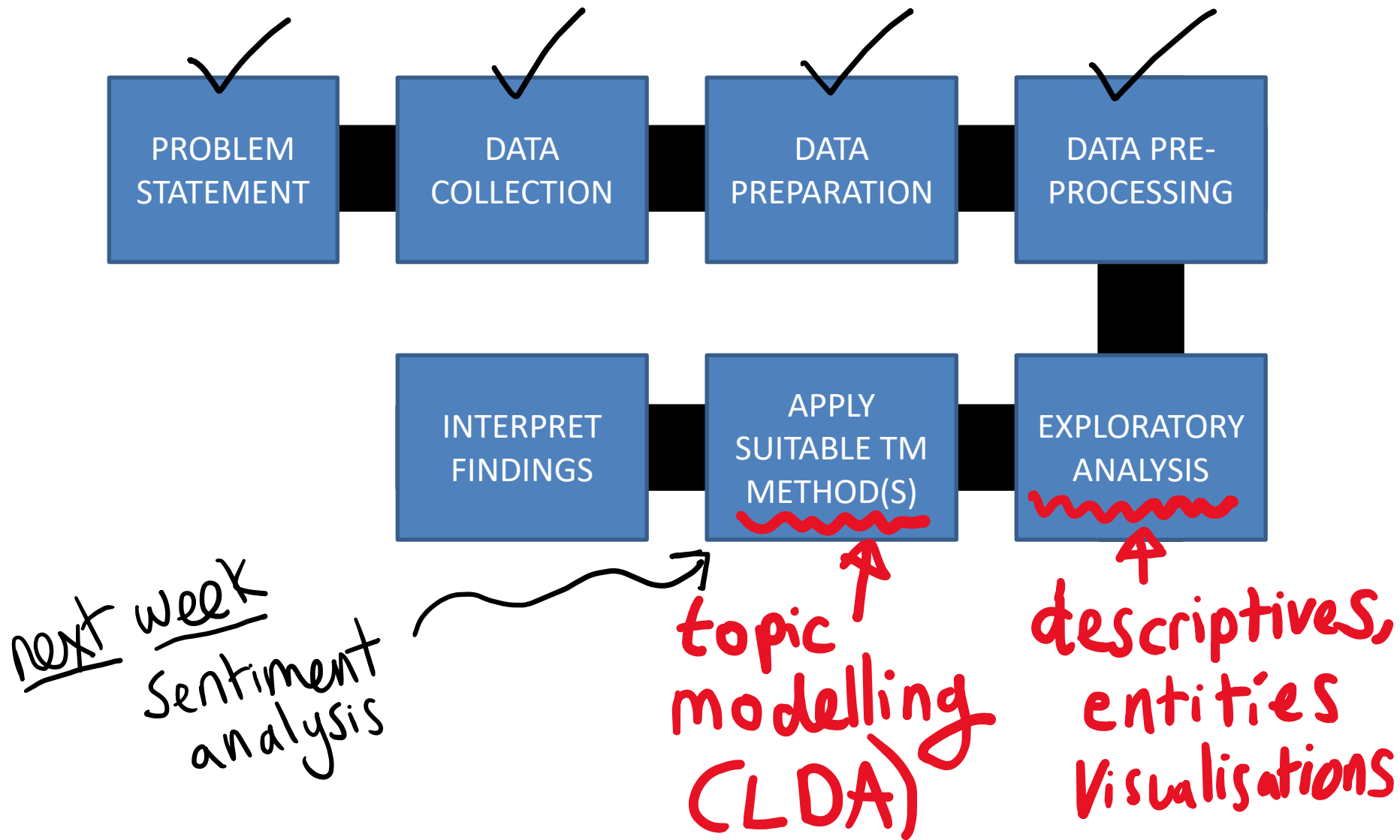
School of Computer Science and Engineering  
115 New Cavendish Street  
University of Westminster  
London, W1W 6UW  
[worrallph@westminster.ac.uk](mailto:worrallph@westminster.ac.uk)

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 x 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

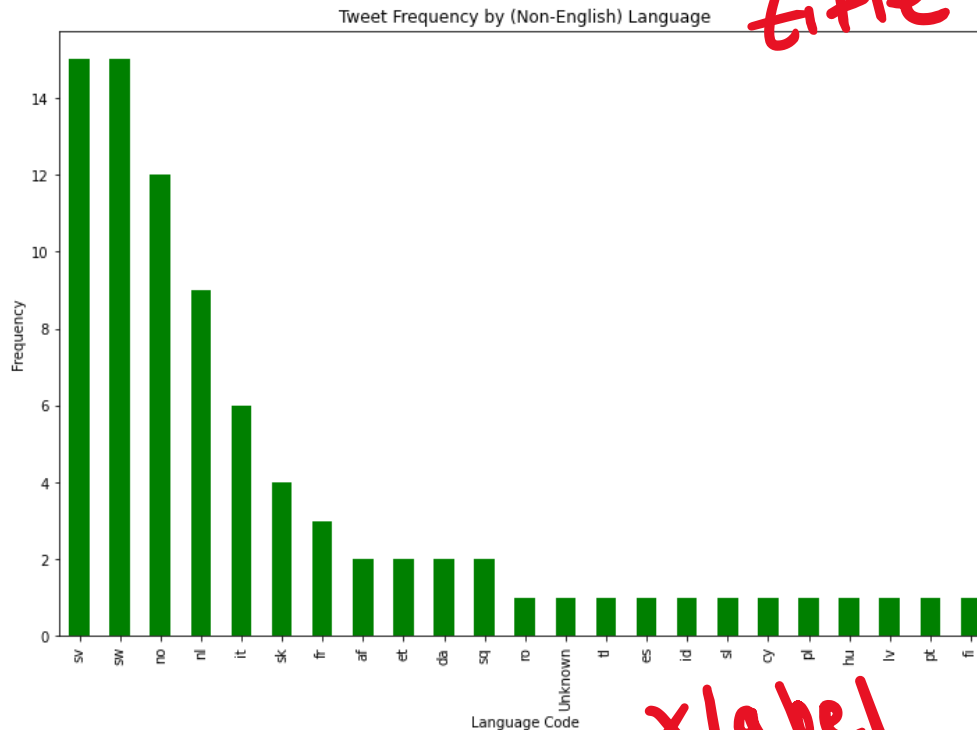
Name: lang, dtype: int64

Identifying commonly used languages

# Visualising the frequencies...

```
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")
```

*Vertical bar*  
*green*  
*?*



# Looking up the method signature...

The screenshot shows the pandas API reference page for `pandas.Series.plot`. The page has a dark theme. On the left is a sidebar with a list of pandas methods. The main content area shows the method signature `Series.plot(*args, **kwargs)` with `Series.plot` underlined in red. To the right of the signature, the text `Kwargs ?` is handwritten in red. Below the signature is the description "Make plots of Series or DataFrame." and a note about the backend. The parameters section lists `data`, `x`, `y`, and `kind` with their descriptions. A [source] link is in the top right.

pandas

Getting started User Guide **API reference** Development Release notes

1.5.3

pandas.Series.plot

Series.plot(\*args, \*\*kwargs) **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...

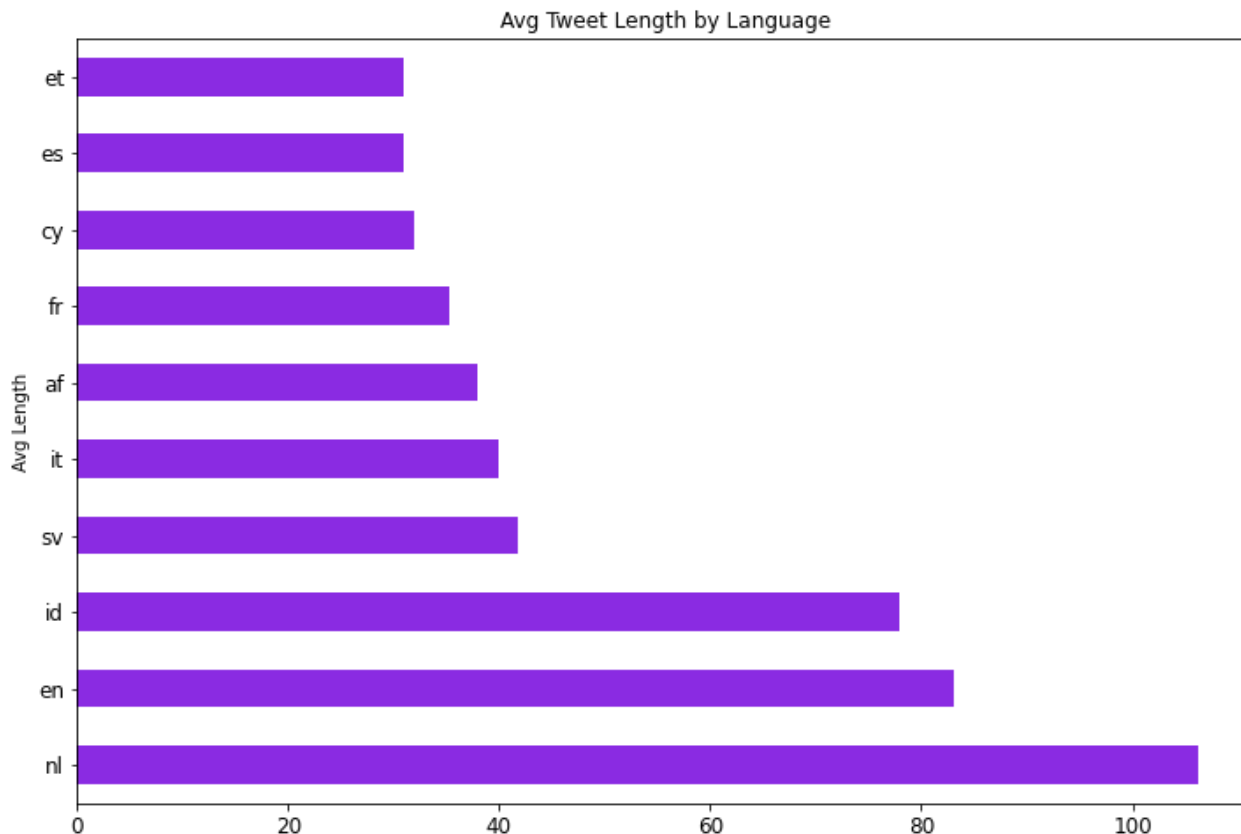
```
df.groupby("lang")["len"].mean().sort_values(ascending=False).head(10)
```

```
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="Language", 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/](https://github.com/amueller/word_cloud/blob/master/examples/)

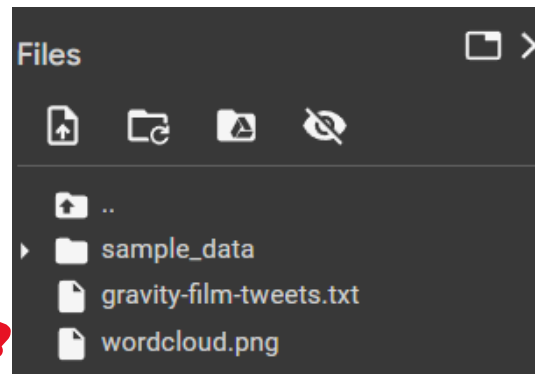
# Generating a Word Cloud...

[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



1998, 1999, 2000, 2001, 2002, 2003, 2004, 2005, 2006, 2007, 2008, 2009, 2010, 2011, 2012, 2013, 2014, 2015, 2016, 2017, 2018, 2019, 2020, 2021, 2022, 2023, 2024, 2025, 2026, 2027, 2028, 2029, 2030, 2031, 2032, 2033, 2034, 2035, 2036, 2037, 2038, 2039, 2040, 2041, 2042, 2043, 2044, 2045, 2046, 2047, 2048, 2049, 2050, 2051, 2052, 2053, 2054, 2055, 2056, 2057, 2058, 2059, 2060, 2061, 2062, 2063, 2064, 2065, 2066, 2067, 2068, 2069, 2070, 2071, 2072, 2073, 2074, 2075, 2076, 2077, 2078, 2079, 2080, 2081, 2082, 2083, 2084, 2085, 2086, 2087, 2088, 2089, 2090, 2091, 2092, 2093, 2094, 2095, 2096, 2097, 2098, 2099, 2100, 2101, 2102, 2103, 2104, 2105, 2106, 2107, 2108, 2109, 2110, 2111, 2112, 2113, 2114, 2115, 2116, 2117, 2118, 2119, 2120, 2121, 2122, 2123, 2124, 2125, 2126, 2127, 2128, 2129, 2130, 2131, 2132, 2133, 2134, 2135, 2136, 2137, 2138, 2139, 2140, 2141, 2142, 2143, 2144, 2145, 2146, 2147, 2148, 2149, 2150, 2151, 2152, 2153, 2154, 2155, 2156, 2157, 2158, 2159, 2160, 2161, 2162, 2163, 2164, 2165, 2166, 2167, 2168, 2169, 2170, 2171, 2172, 2173, 2174, 2175, 2176, 2177, 2178, 2179, 2180, 2181, 2182, 2183, 2184, 2185, 2186, 2187, 2188, 2189, 2190, 2191, 2192, 2193, 2194, 2195, 2196, 2197, 2198, 2199, 2200, 2201, 2202, 2203, 2204, 2205, 2206, 2207, 2208, 2209, 2210, 2211, 2212, 2213, 2214, 2215, 2216, 2217, 2218, 2219, 2220, 2221, 2222, 2223, 2224, 2225, 2226, 2227, 2228, 2229, 2230, 2231, 2232, 2233, 2234, 2235, 2236, 2237, 2238, 2239, 2240, 2241, 2242, 2243, 2244, 2245, 2246, 2247, 2248, 2249, 2250, 2251, 2252, 2253, 2254, 2255, 2256, 2257, 2258, 2259, 2260, 2261, 2262, 2263, 2264, 2265, 2266, 2267, 2268, 2269, 2270, 2271, 2272, 2273, 2274, 2275, 2276, 2277, 2278, 2279, 2280, 2281, 2282, 2283, 2284, 2285, 2286, 2287, 2288, 2289, 2290, 2291, 2292, 2293, 2294, 2295, 2296, 2297, 2298, 2299, 2300, 2301, 2302, 2303, 2304, 2305, 2306, 2307, 2308, 2309, 2310, 2311, 2312, 2313, 2314, 2315, 2316, 2317, 2318, 2319, 2320, 2321, 2322, 2323, 2324, 2325, 2326, 2327, 2328, 2329, 2330, 2331, 2332, 2333, 2334, 2335, 2336, 2337, 2338, 2339, 2340, 2341, 2342, 2343, 2344, 2345, 2346, 2347, 2348, 2349, 2350, 2351, 2352, 2353, 2354, 2355, 2356, 2357, 2358, 2359, 2360, 2361, 2362, 2363, 2364, 2365, 2366, 2367, 2368, 2369, 2370, 2371, 2372, 2373, 2374, 2375, 2376, 2377, 2378, 2379, 2380, 2381, 2382, 2383, 2384, 2385, 2386, 2387, 2388, 2389, 2390, 2391, 2392, 2393, 2394, 2395, 2396, 2397, 2398, 2399, 2400, 2401, 2402, 2403, 2404, 2405, 2406, 2407, 2408, 2409, 2410, 2411, 2412, 2413, 2414, 2415, 2416, 2417, 2418, 2419, 2420, 2421, 2422, 2423, 2424, 2425, 2426, 2427, 2428, 2429, 2430, 2431, 2432, 2433, 2434, 2435, 2436, 2437, 2438, 2439, 2440, 2441, 2442, 2443, 2444, 2445, 2446, 2447, 2448, 2449, 2450, 2451, 2452, 2453, 2454, 2455, 2456, 2457, 2458, 2459, 2460, 2461, 2462, 2463, 2464, 2465, 2466, 2467, 2468, 2469, 2470, 2471, 2472, 2473, 2474, 2475, 2476, 2477, 2478, 2479, 2480, 2481, 2482, 2483, 2484, 2485, 2486, 2487, 2488, 2489, 2490, 2491, 2492, 2493, 2494, 2495, 2496, 2497, 2498, 2499, 2500, 2501, 2502, 2503, 2504, 2505, 2506, 2507, 2508, 2509, 2510, 2511, 2512, 2513, 2514, 2515, 2516, 2517, 2518, 2519, 2520, 2521, 2522, 2523, 2524, 2525, 2526, 2527, 2528, 2529, 2530, 2531, 2532, 2533, 2534, 2535, 2536, 2537, 2538, 2539, 2540, 2541, 2542, 2543, 2544, 2545, 2546, 2547, 2548, 2549, 2550, 2551, 2552, 2553, 2554, 2555, 2556, 2557, 2558, 2559, 2560, 2561, 2562, 2563, 2564, 2565, 2566, 2567, 2568, 2569, 2570, 2571, 2572, 2573, 2574, 2575, 2576, 2577, 2578, 2579, 2580, 2581, 2582, 2583, 2584, 2585, 2586, 2587, 2588, 2589, 2590, 2591, 2592, 2593, 2594, 2595, 2596, 2597, 2598, 2599, 2600, 2601, 2602, 2603, 2604, 2605, 2606, 2607, 2608, 2609, 2610, 2611, 2612, 2613, 2614, 2615, 2616, 2617, 2618, 2619, 2620, 2621, 2622, 2623, 2624, 2625, 2626, 2627, 2628, 2629, 2630, 2631, 2632, 2633, 2634, 2635, 2636, 2637, 2638, 2639, 2640, 2641, 2642, 2643, 2644, 2645, 2646, 2647, 2648, 2649, 2650, 2651, 2652, 2653, 2654, 2655, 2656, 2657, 2658, 2659, 2660, 2661, 2662, 2663, 2664, 2665, 2666, 2667, 2668, 2669, 2670, 2671, 2672, 2673, 2674, 2675, 2676, 2677, 2678, 2679, 26



# 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 zip code

# 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...

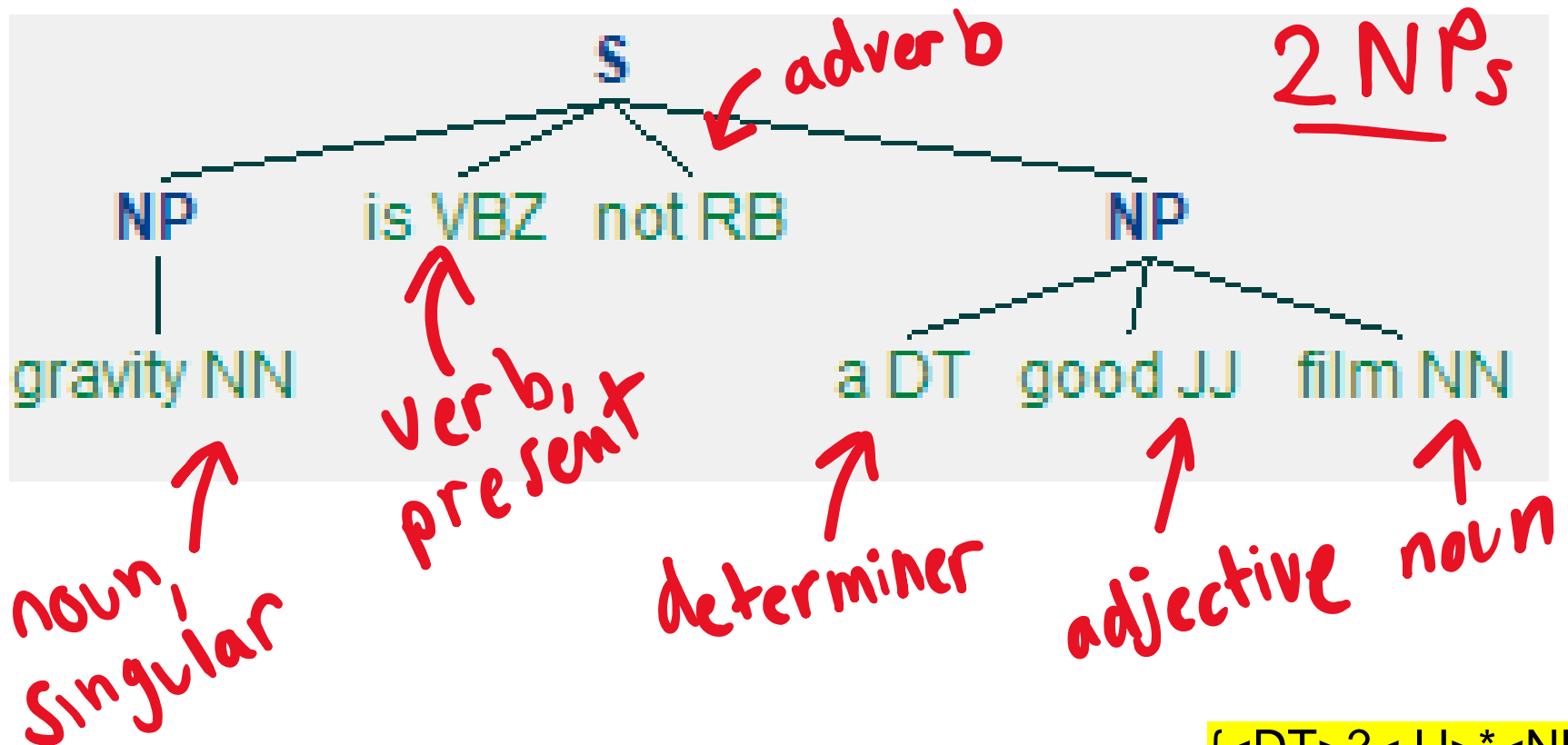
Modern search engines still struggle to distinguish between these two very similar concepts..

- 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"



Parts of Speech Tags used in NLTK

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

# 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

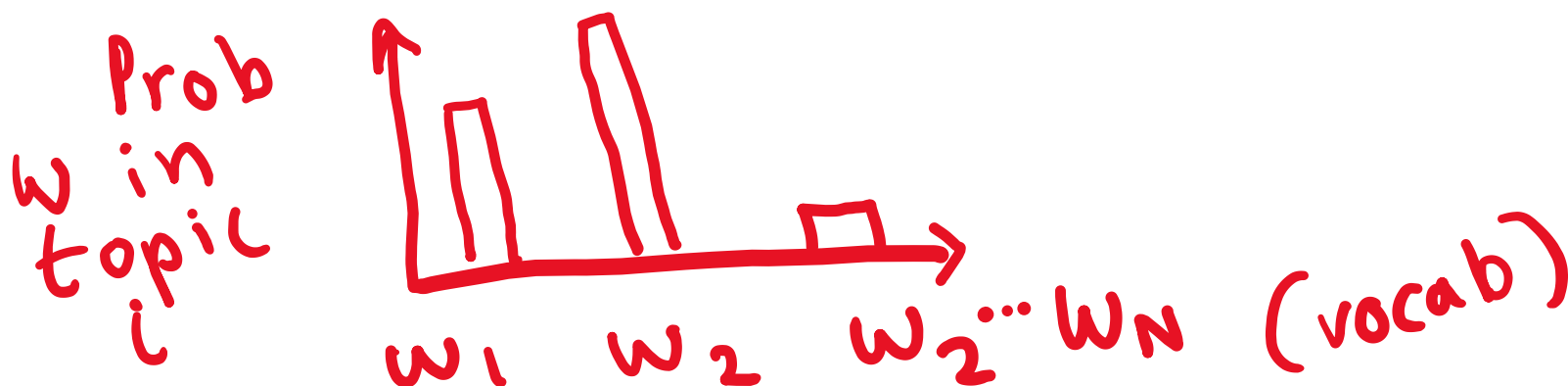
LDA allocation is a type of generative probabilistic topic model that can be used to identify a set of  $k$  topics, where each topic is defined by a probability of containing certain words and each phrase/document is a probabilistic mixture of all topics. It's a generative approach because it creates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

di-rich-let

# The LDA topic allocation model...

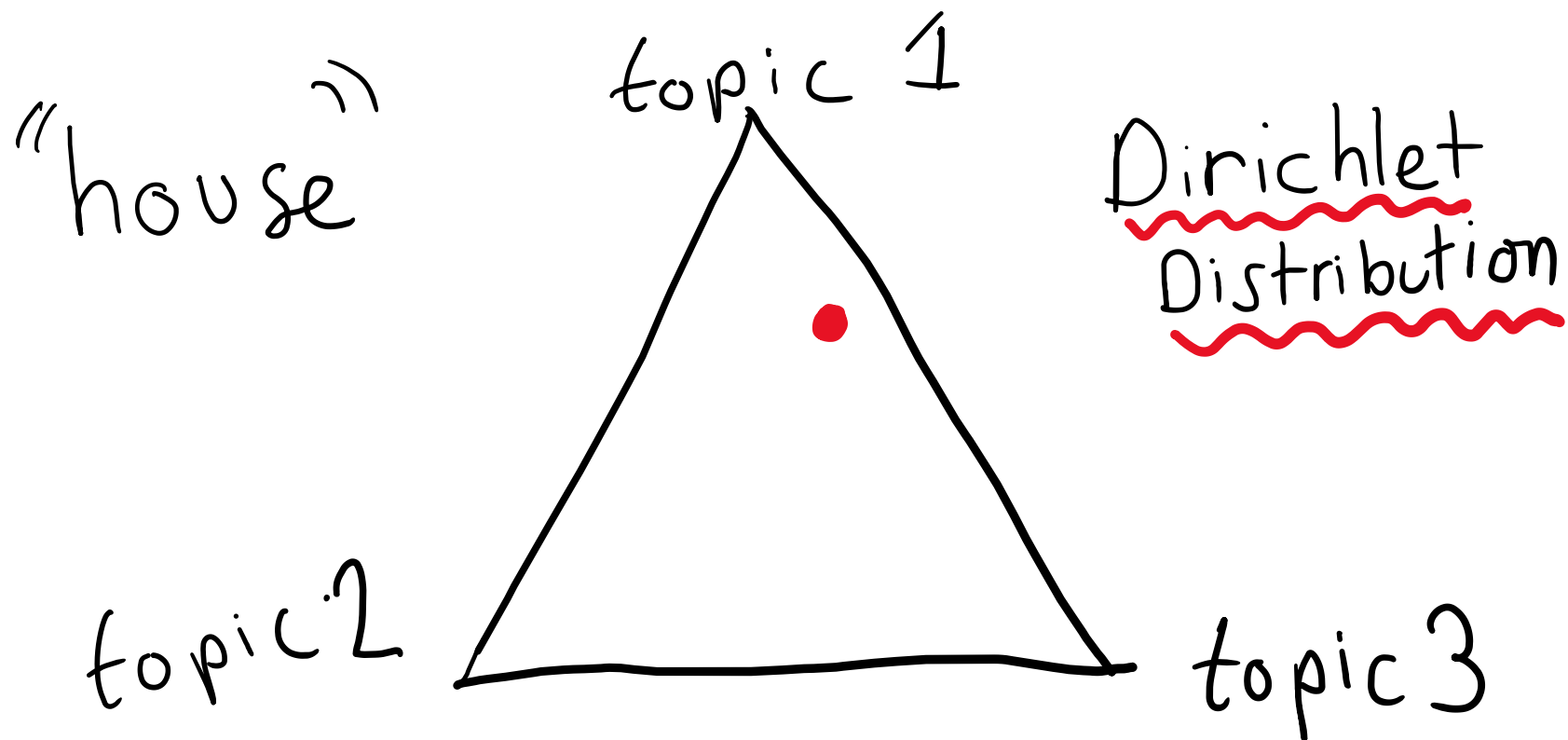
## Topic

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...

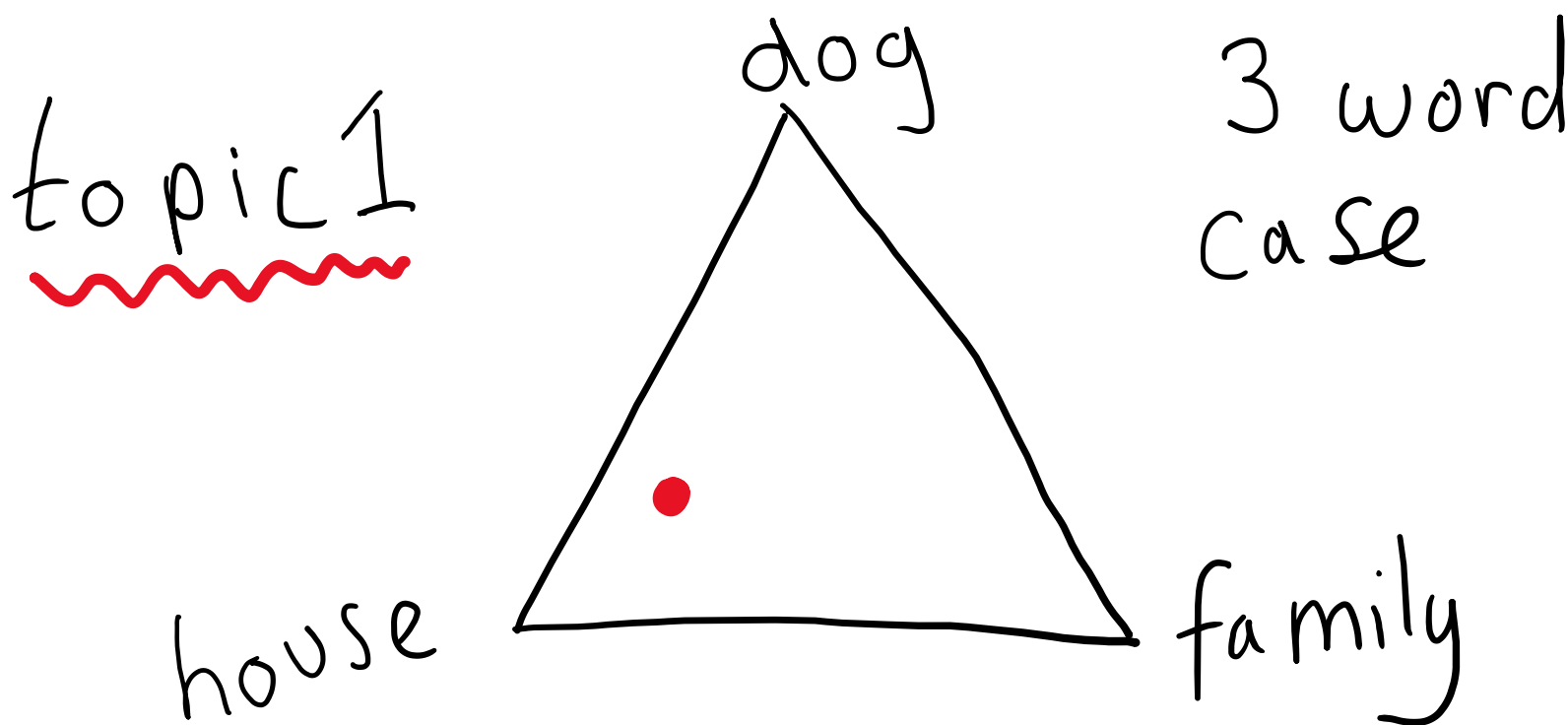
- Every word belongs to each topic but with a different probability





# Intuition behind the LDA...

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



# 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 Jordan, M \(2003\)](#)

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

# Performing the LDA in 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]

num_topics = 5
lda = gensim.models.LdaMulticore(corpus=corpus,
                                id2word=vocab,
                                num_topics=num_topics)

pprint(lda.print_topics())
```

*↑ bay of words*

# Interpreting the abstract topics...

```
WARNING:tensorflow:Model is not compatible for CPU. Few updates, training on GPU.  
[(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 (word 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