Text-Mining: Advanced Options

Dr. J. Kasmire
Research Fellow at Cathie Marsh Institute and UK Data Service

julia.kasmire@manchester.ac.uk

@JKasmireComplex

You might also be interested in...

Recent -

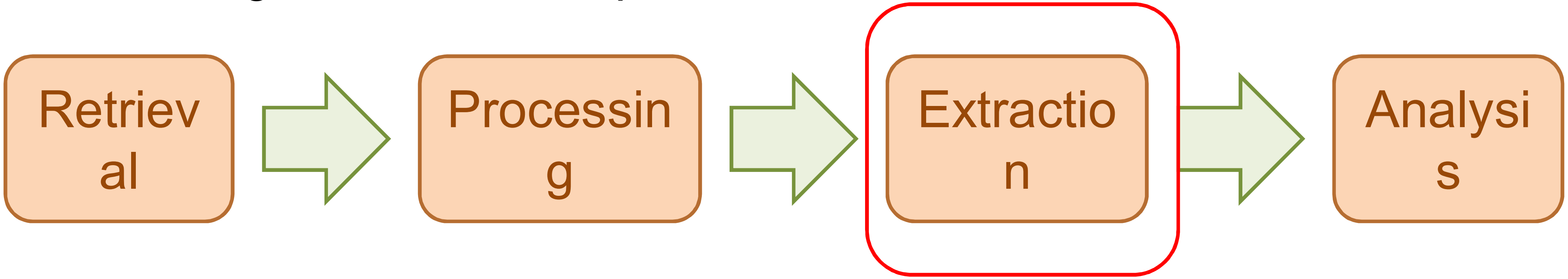
- Being a Computational Social Scientist
- Text-mining – Intro and Theory, Basic Processes
- Web-scraping for Social Science Research (case study, from websites, and from API's)
- Code Demos
- https://www.ukdataservice.ac.uk/news-and-events/events/past-events.aspx
- https://www.youtube.com/user/UKDATASERVICE

Upcoming -

- Health Studies User Conference 30 June 20
- Social Data and the Third Sector 2 to 16 July 20
- Text-mining Code Demos – expected in September

UK Data Service

Text-mining has 4 basic steps



Processes:
- Tokenisation, standardisation, removing irrelevancies, linguistic form consolidation

Basic NLP:
- Tagging, Chunking, etc.

Basic Extraction:
- Frequency counts, similarity, discovery

Advanced Extraction:
- Classification tasks
- Sentiment analysis
- Extracted named entities
- Network graphs

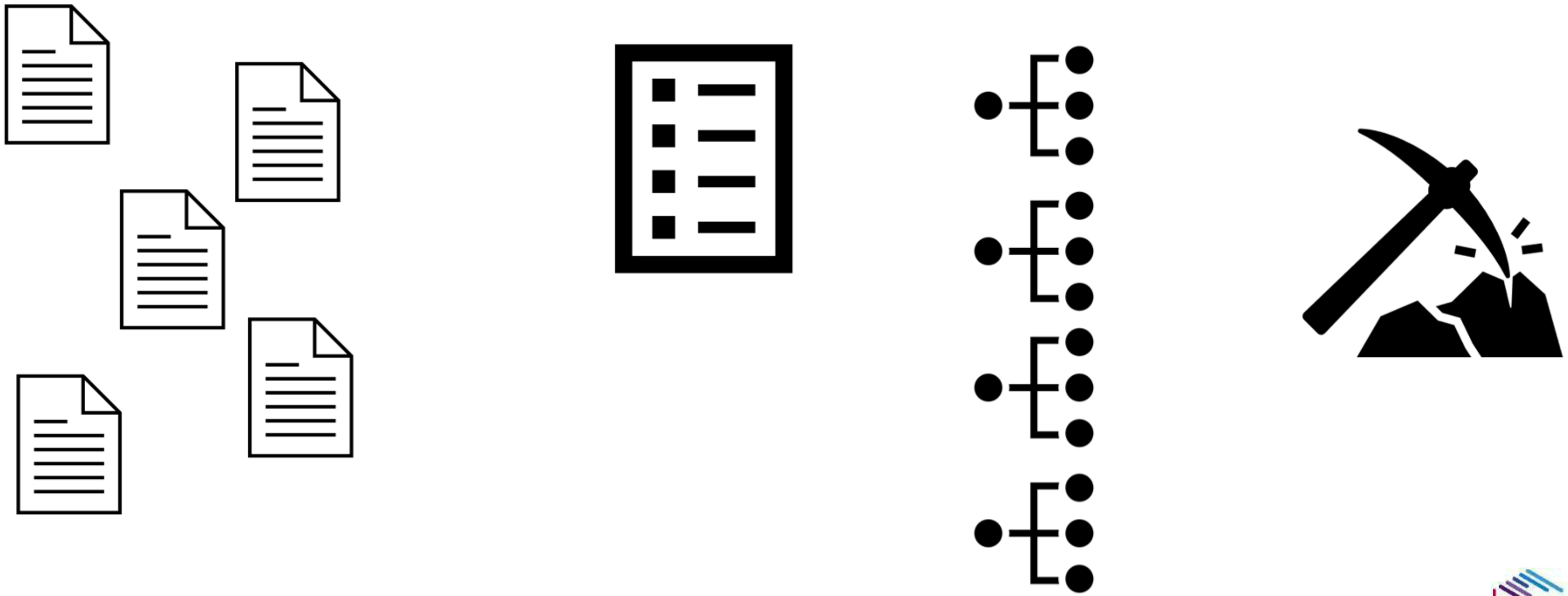
UK Data Service

Automatic classification – a thought experiment

- 100,000 old scientific articles to be sorted into which modern scientific field they match best
- No keywords, not published in journals or published in journals that don’t match current fields, use old terminology, etc.
- Rather than read and manually classify them all, how about we teach a computer to classify them for us

Classification tasks require:

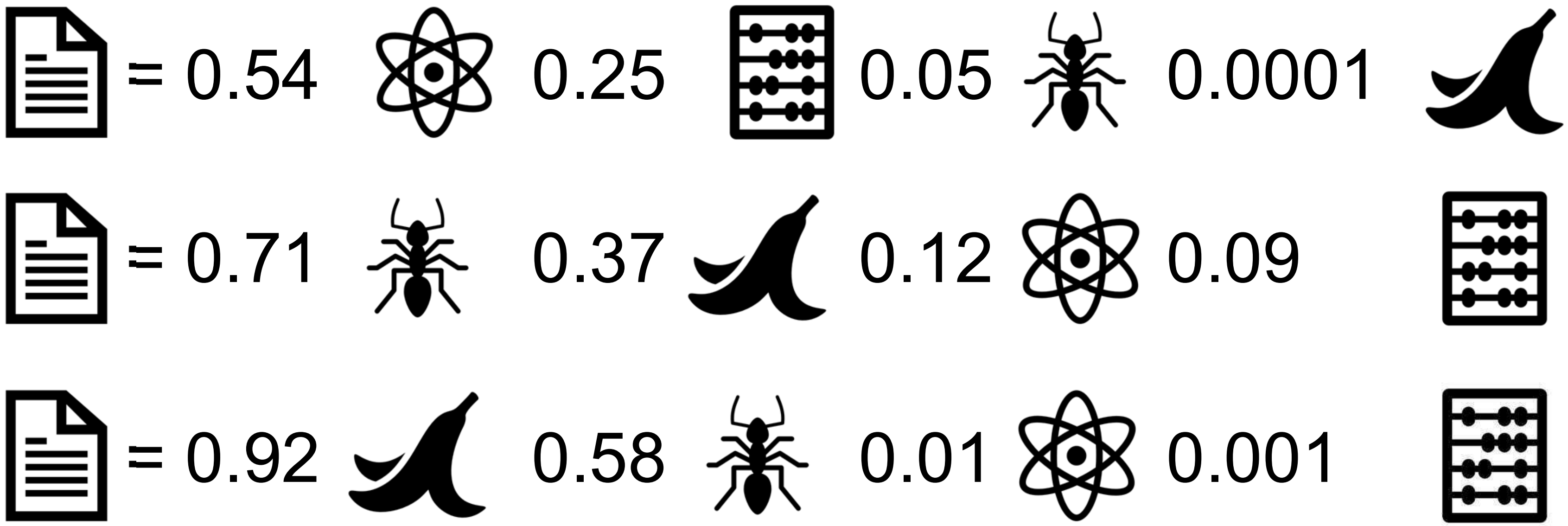
- A set of documents
- A set of classes to which the documents may belong.
- A tool that makes predictions about what classes the documents belong to



UK Data Service

Classification returns:

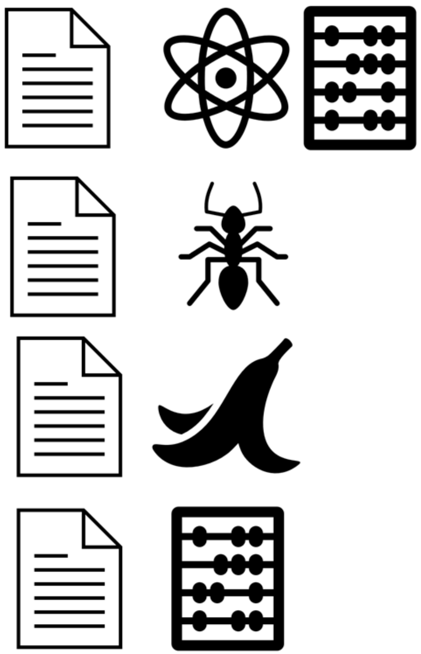
- A prediction about what class a new document belongs to
- A number between 0 and 1 for each class

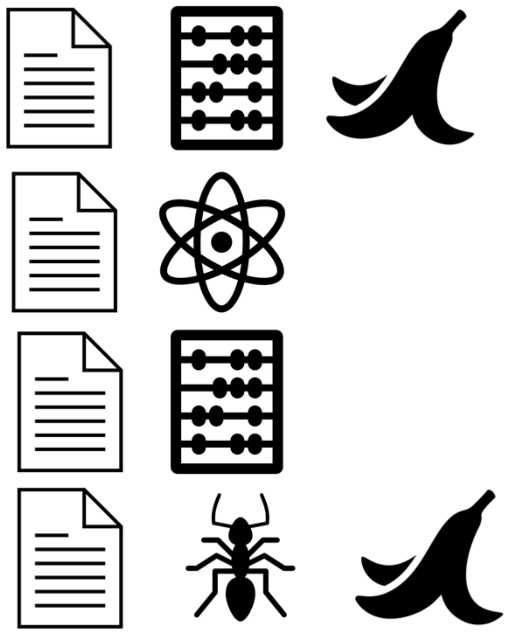


UK Data Service

Machine learning

- A training set of documents that are already correctly classified.
- Train a model or learning algorithm on the training set.
- A test set of documents that are already correctly classified.
- Test the model, comparing performance to a benchmark if possible.

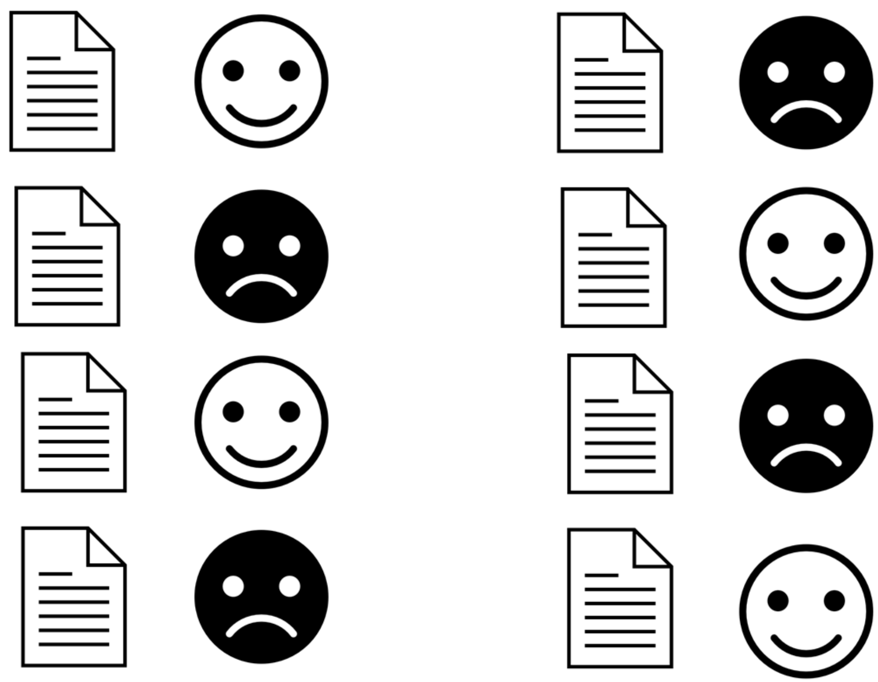




UK Data Service

Sentiment analysis

- Training & test sets = csv/data frame/etc. with text documents and ‘pos’ or ‘neg’S tags
- Learning algorithm = spaCy/nltk/other nlp option
- Benchmark not always relevant, performance metrics still are



UK Data Service

Sample training and test data

```
train = [
    ('I love this sandwich.', 'pos'),
    ('this is an amazing place!', 'pos'),
    ('I feel very good about these beers.', 'pos'),
    ('this is my best work.', 'pos'),
    ("what an awesome view", 'pos'),
    ('I do not like this restaurant', 'neg'),
    ('I am tired of this stuff.', 'neg'),
    ("I can't deal with this", 'neg'),
    ('he is my sworn enemy!', 'neg'),
    ('my boss is horrible.', 'neg')]
```

```
test = [
    ('the beer was good.', 'pos'),
    ('I do not enjoy my job', 'neg'),
    ("I ain't feeling dandy today.", 'neg'),
    ("I feel amazing!", 'pos'),
    ('Gary is a friend of mine.', 'pos'),
    ("I can't believe I'm doing this.", 'neg')]
```

Training = associates features (words) to scores (word1 = ‘pos’, word2 = ‘pos’, wordn = ‘neg’)
Test = sums feature associations to get probable score (‘pos’ + ‘neg’ + ‘pos’ = ‘pos’)

Naïve Bayes Classification – basic frequency in action

Training = 'I' 'love' 'this' 'sandwich' (pos) plus 'I' "can't" 'deal' 'with' 'this' (neg)

'love' 'sandwich' "can't" 'deal' 'with'

'I' 'this' 'I' 'this'

Test = ‘I deal with sandwiches’ (neg)

'I deal with sandwiches’

Prediction = ‘neg’ Actual = ‘neg’

Prediction strength = –0.25.

There are more sophisticated options if you want a custom naïve bayes classifier.

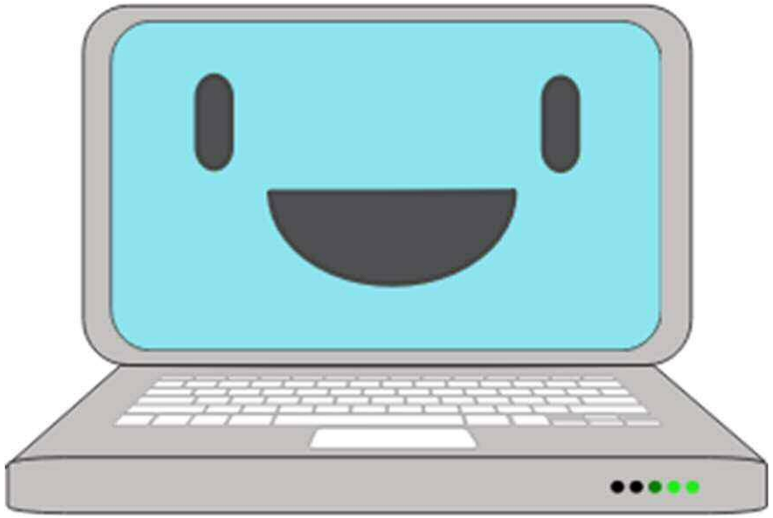
UK Data Service

Efficiency

Real training and test data sets are huge.

Processing will reduce the number of (irrelevant) features to be extracted/evaluated.

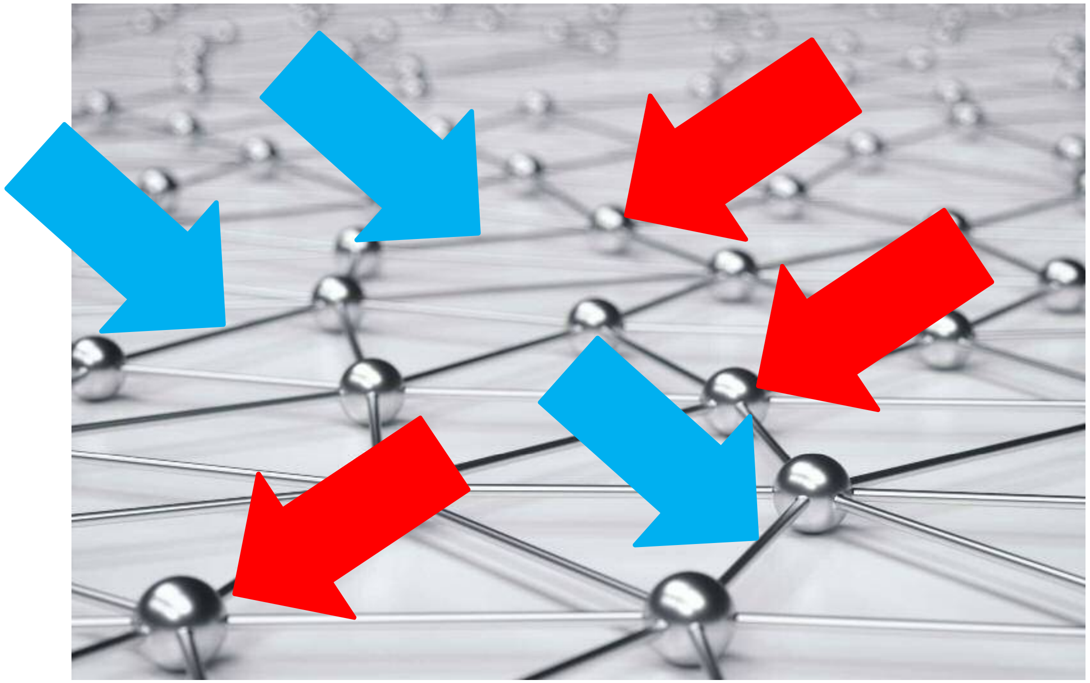
love, loves, loving …
deal, deals, dealing …



love
deal

UK Data Service

Network graphs



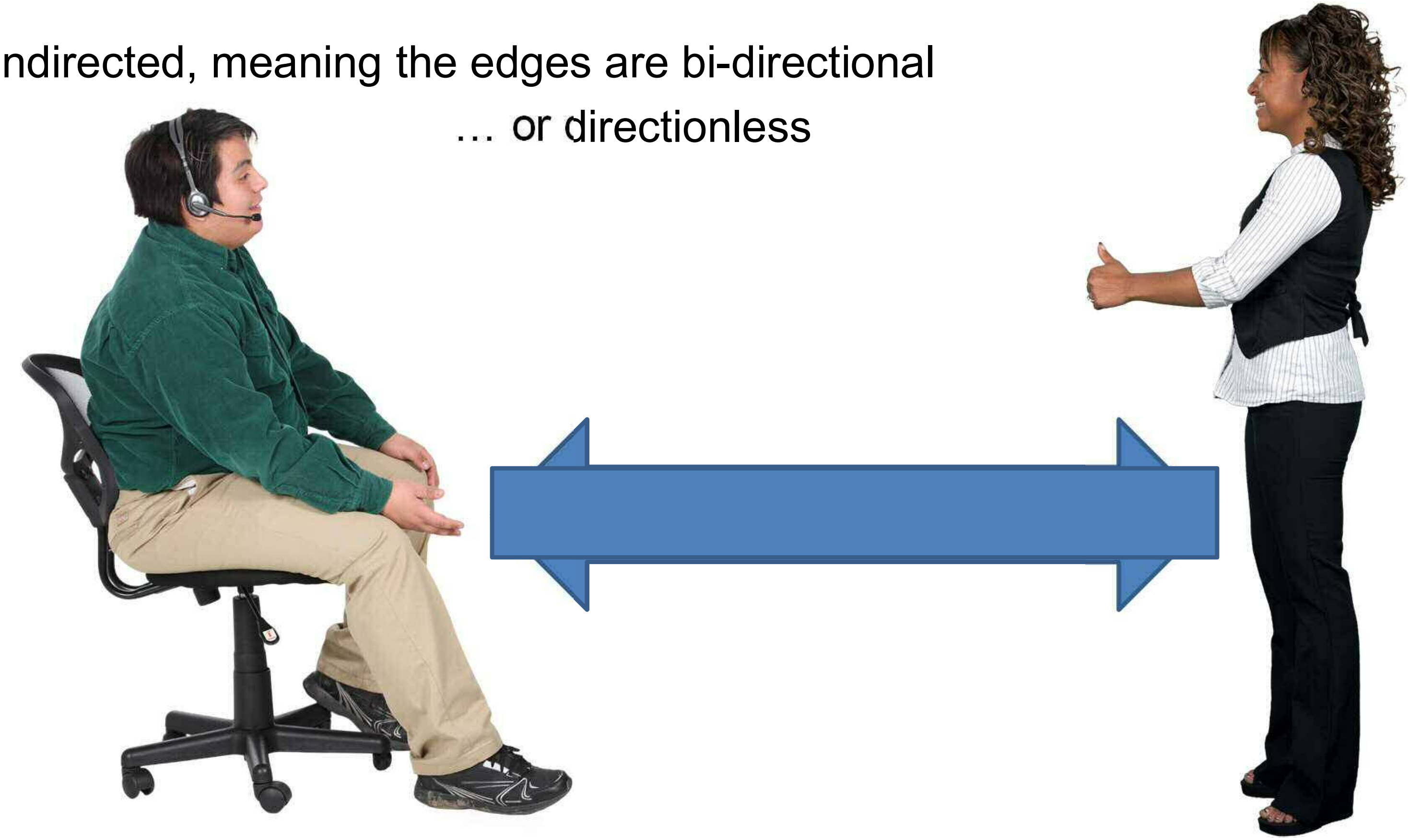
Map relationships between things

- The things are ‘nodes’
- The relationships are links or ‘edges’

Network graphs can be…

Undirected, meaning the edges are bi-directional

… or directionless

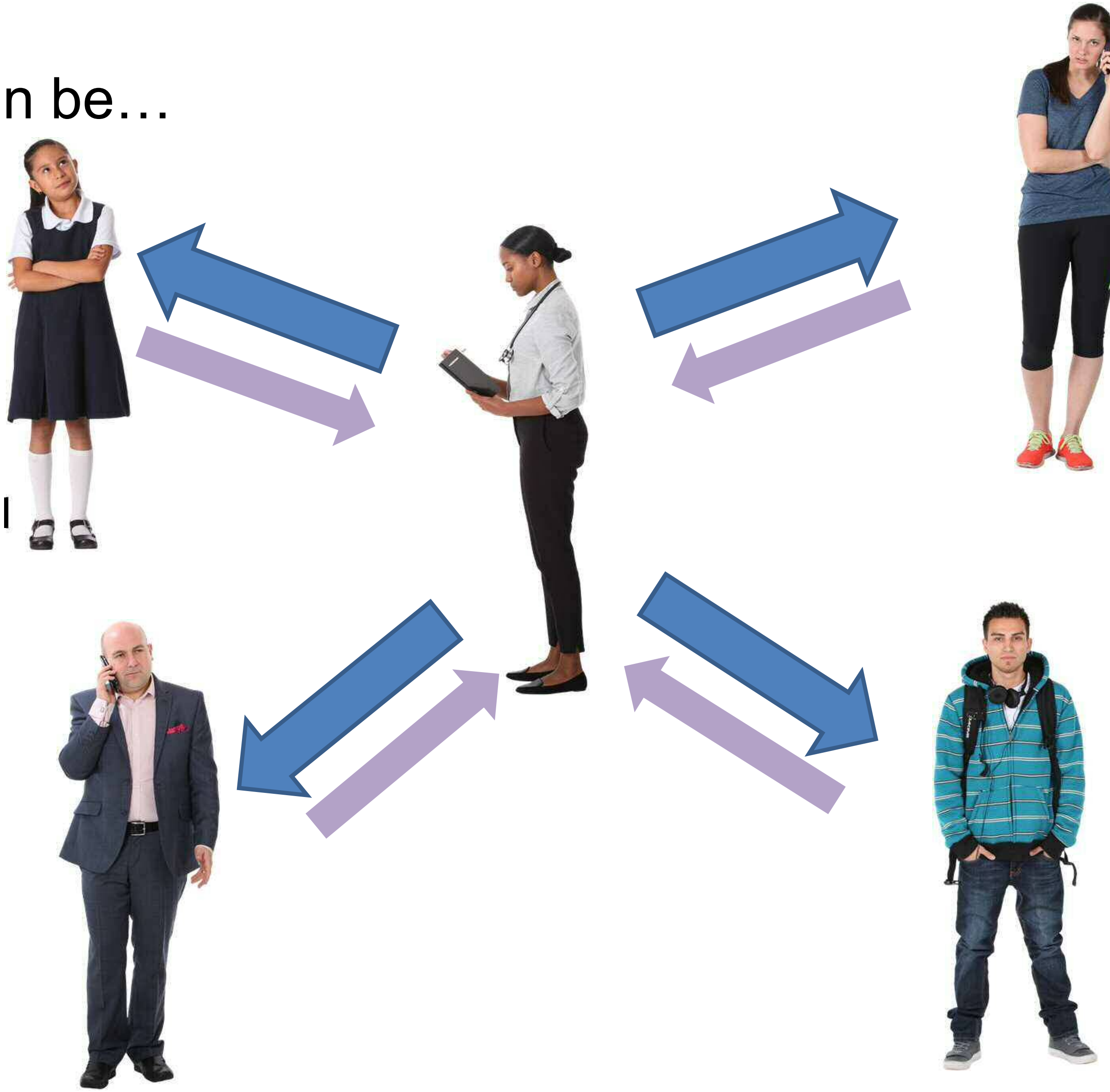


Network graph can be…

Directed,
meaning the edges are
uni-directional

Indicates non-reciprocal
relationships

Although nodes can by
multiply linked to show
reciprocal but unequal
relationships

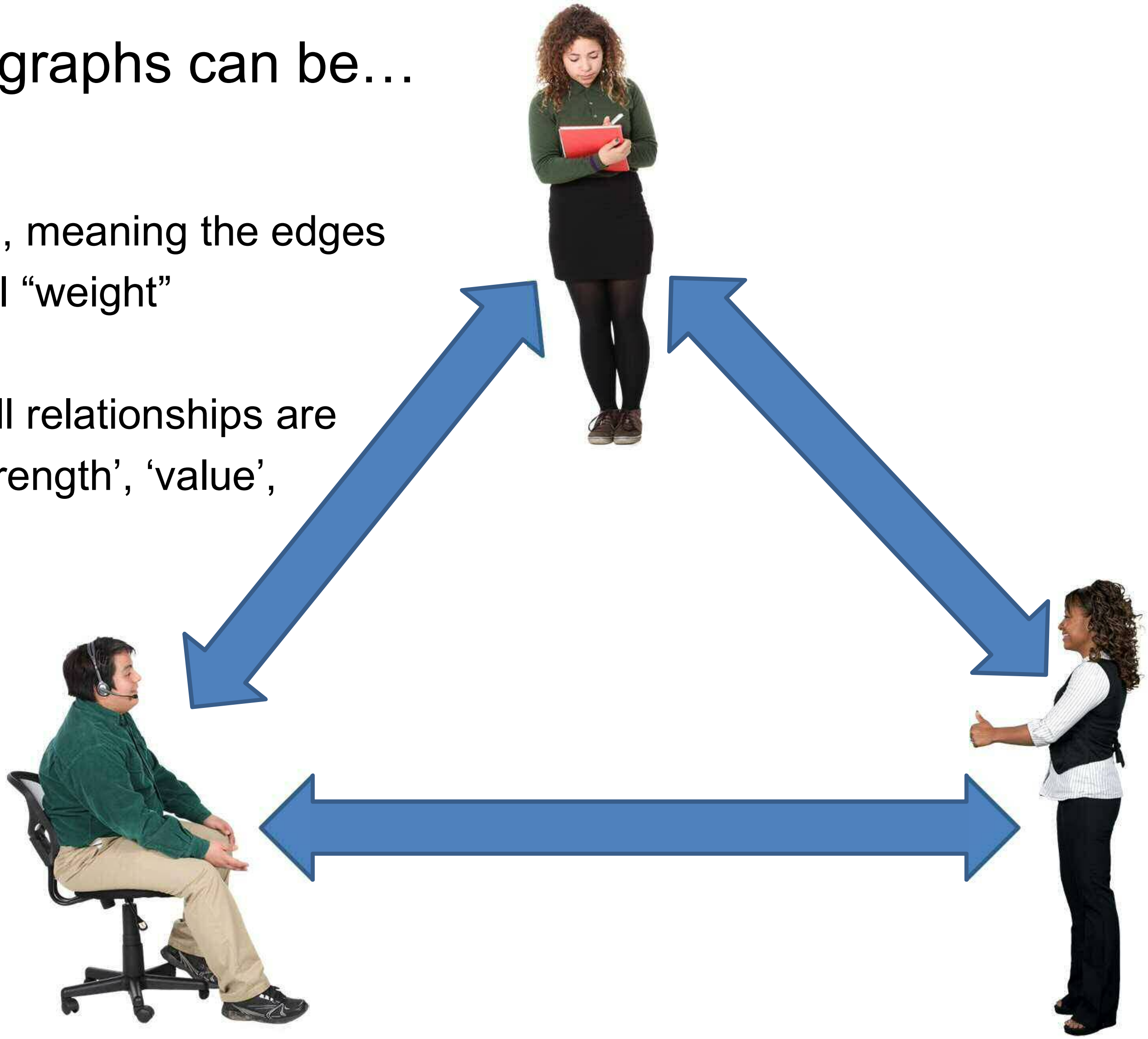


UK Data Service

Network graphs can be…

Unweighted, meaning the edges are all equal “weight”

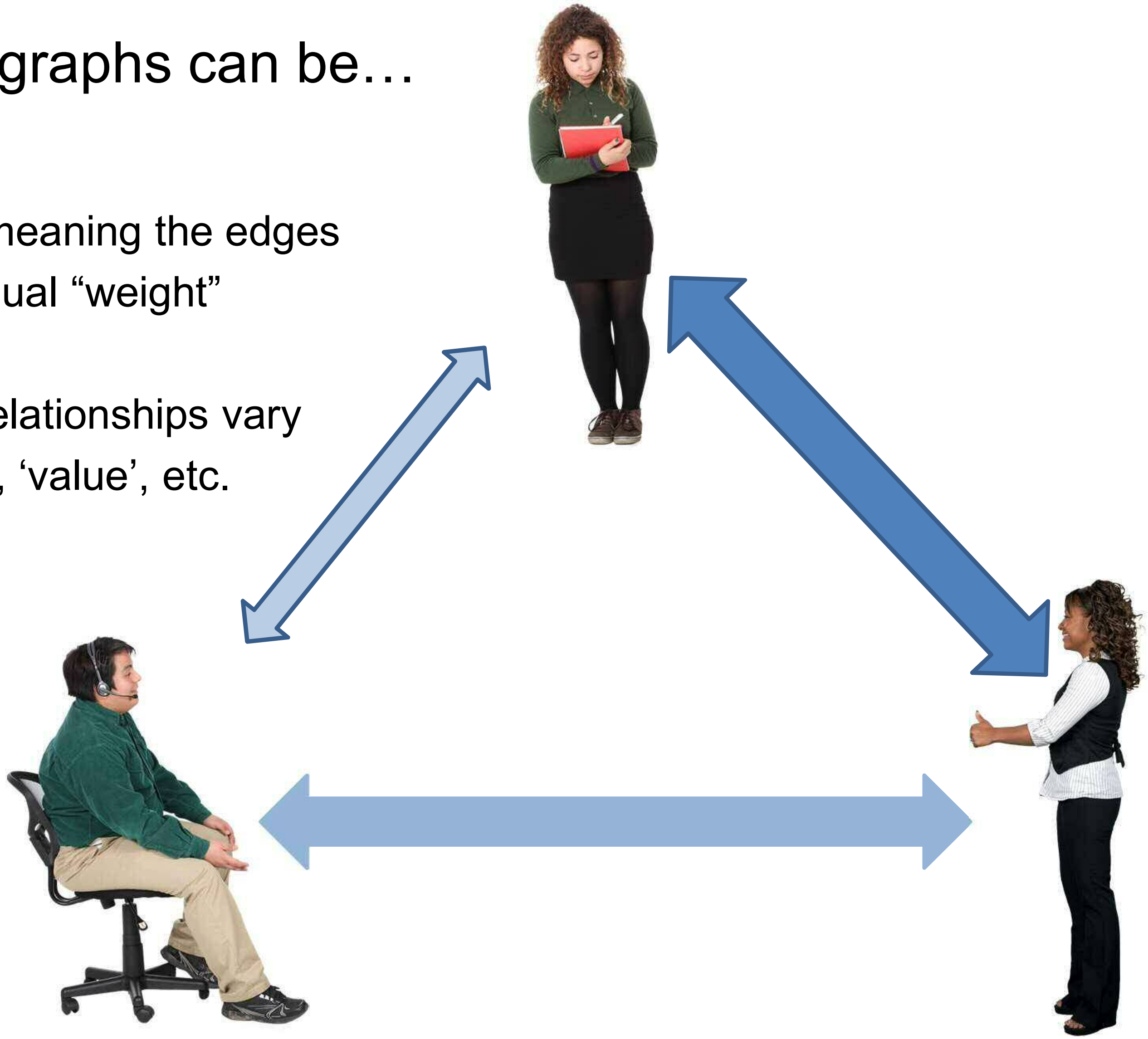
Indicating all relationships are Of equal ‘strength’, ‘value’, etc.



Network graphs can be…

Weighted, meaning the edges have individual "weight"

Indicating relationships vary in 'strength', 'value', etc.



UK Data Service

Nodes – basic processes

['Archibald walked through Manchester with Beryl.']

```
[('Archibald', 'NNP'), ('walked', 'VBD'),
('through', 'IN'), ('Manchester', 'NNP'),
('with', 'IN'), ('Beryl', 'NNP')('.', '.')]
```

['Tariq saw Beryl when she was playing tennis.',]

```
[('Tariq', 'NNP'), ('saw', 'VBD'), ('Beryl', 'NNP'),
('when', 'WRB'), ('she', 'PRP'), ('was', 'VBD'),
('playing', 'VBG'), ('tennis', 'NN'), ('.', '.')]
```

['Archibald shares a house with Beryl and Cerys.']

```
[('Archibald', 'NNP')), ('shares', 'NNS'), ('a',
'DT'), ('house', 'NN'), ('with', 'IN'), ('Beryl',
'NNP'), ('and', 'CC'), ('Cerys', 'NNP') , ('.', '.')]
```

Nodes – basic processes → Named Entity Recognition chunker

['Archibald walked through Manchester with Beryl.']

```
[Tree('S',
[Tree('PERSON', [('Archibald', 'NNP')]), 'walked', 'VBD'),
('through', 'IN'), ('Manchester', 'NNP'), ('with', 'IN'),
Tree('PERSON', [('Beryl', 'NNP')]), ('.', '.')])
```

['Tariq saw Beryl when she was playing tennis.',]

```
Tree('S',
[Tree('PERSON', [('Tariq', 'NNP')]), ('saw', 'VBD'),
Tree('PERSON', [('Beryl', 'NNP')]), ('when', 'WRB'), ('she',
'PRP'), ('was', 'VBD'), ('playing', 'VBG'), ('tennis',
'NN'), ('.', '.')])
```

['Archibald shares a house with Beryl and Cerys.']

```
Tree('S',
[Tree('PERSON', [('Archibald', 'NNP')]), ('shares', 'NNS'),
('a', 'DT'), ('house', 'NN'), ('with', 'IN'),
Tree('PERSON', [('Beryl', 'NNP')]), ('and', 'CC'),
Tree('PERSON', [('Cerys', 'NNP')]), ('.', '.')])
```

Nodes – basics → NE chunker → Extract chunks

['Archibald walked through Manchester with Beryl.']

`['Archibald', 'Beryl']`

['Tariq saw Beryl when she was playing tennis.',]

`['Tariq','Beryl']`

['Archibald shares a house with Beryl and Cerys.']

`['Archibald', 'Beryl', 'Cerys']`

UK Data Service

Nodes – basics → NE chunk → Extract chunks → find unique chunks

['Archibald walked through Manchester with Beryl.']

['Tariq saw Beryl when she was playing tennis.',]

['Archibald shares a house with Beryl and Cerys.']

```
['Archibald',
'Beryl'
'Tariq',
'Cerys']
```

UK Data Service

Edges – basics → NE chunk → Extract chunks

['Archibald walked through Manchester with Beryl.’]

`['Archibald', 'Beryl’]`

['Tariq saw Beryl when she was playing tennis.’,]

`['Tariq','Beryl’]`

['Archibald shares a house with Beryl and Cerys.’]

`['Archibald', 'Beryl', 'Cerys']`

UK Data Service

Edges – basics → NE chunk → Extract chunks → co-occurring pairs

['Archibald walked through Manchester with Beryl.']

```
[('Archibald','Beryl'),
('Beryl', 'Archibald')]
```

['Tariq saw Beryl when she was playing tennis.',]

```
['Tariq','Beryl'),
('Beryl', 'Tariq')]
```

['Archibald shares a house with Beryl and Cerys.']

```
[('Archibald', 'Beryl'),
('Beryl', 'Archibald'),
('Archibald', 'Cerys'),
('Cerys', 'Archibald'),
('Beryl', 'Cerys'),
('Cerys', 'Beryl')]
```

UK Data Service

Edges – basics → NE chunks → co-occurrences→ weights/directed?

['Archibald walked through Manchester with Beryl.’]

```
[('Archibald', 'Beryl’, 1),
('Beryl', 'Archibald', 1)]
```

['Tariq saw Beryl when she was playing tennis.’,]

```
['Tariq','Beryl', 0.5),
('Beryl', 'Tariq’, 0.1)]
```

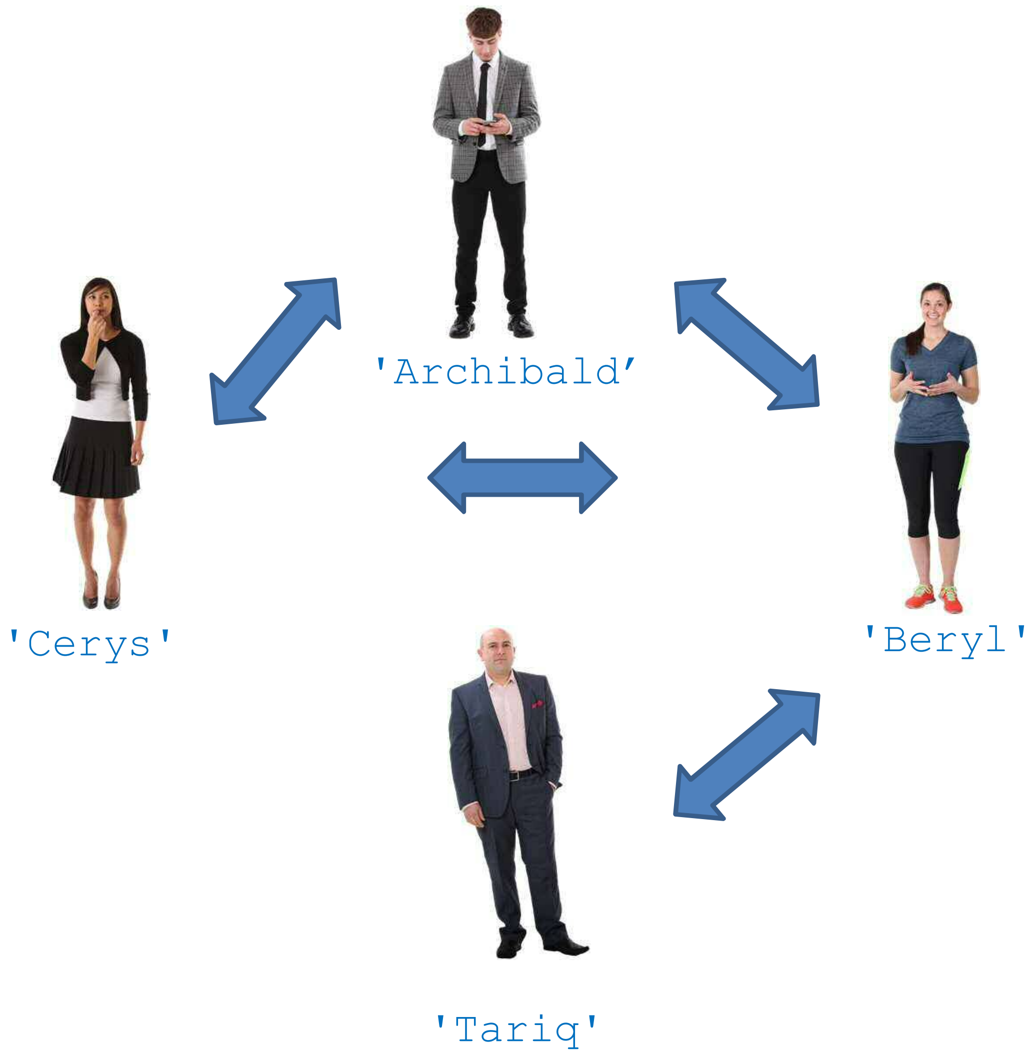
['Archibald shares a house with Beryl and Cerys.’]

```
[('Archibald', 'Beryl’, 20),
('Beryl', 'Archibald’, 20)),
('Archibald', 'Cerys’, 20)),
('Cerys', 'Archibald’, 20)),
('Beryl', 'Cerys’, 20)),
('Cerys', 'Beryl’, 20))]
```

UK Data Service

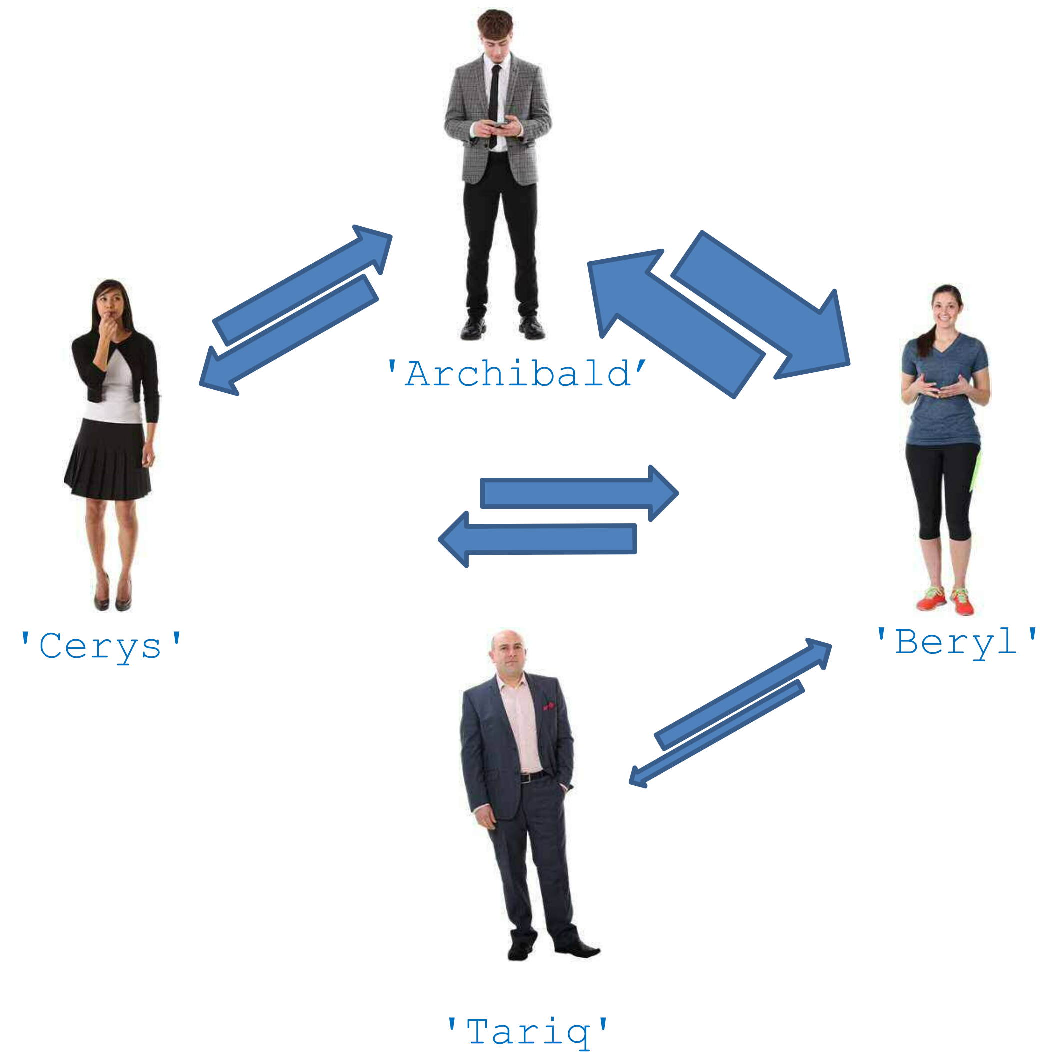
Populated a network graph with extracted nodes and edges

Undirected
Unweighted



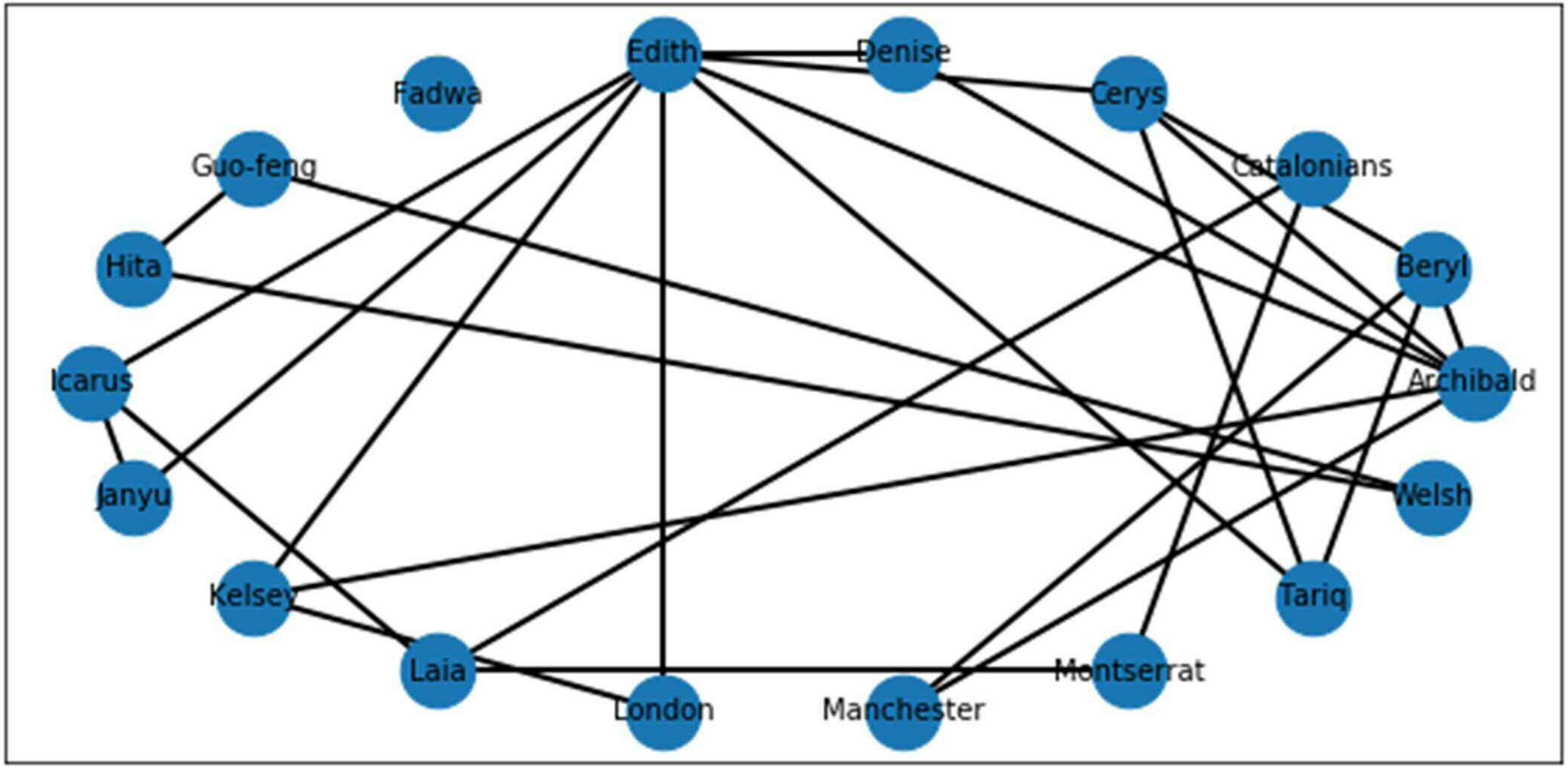
Populated a network graph with extracted nodes and edges

Directed
Weighted



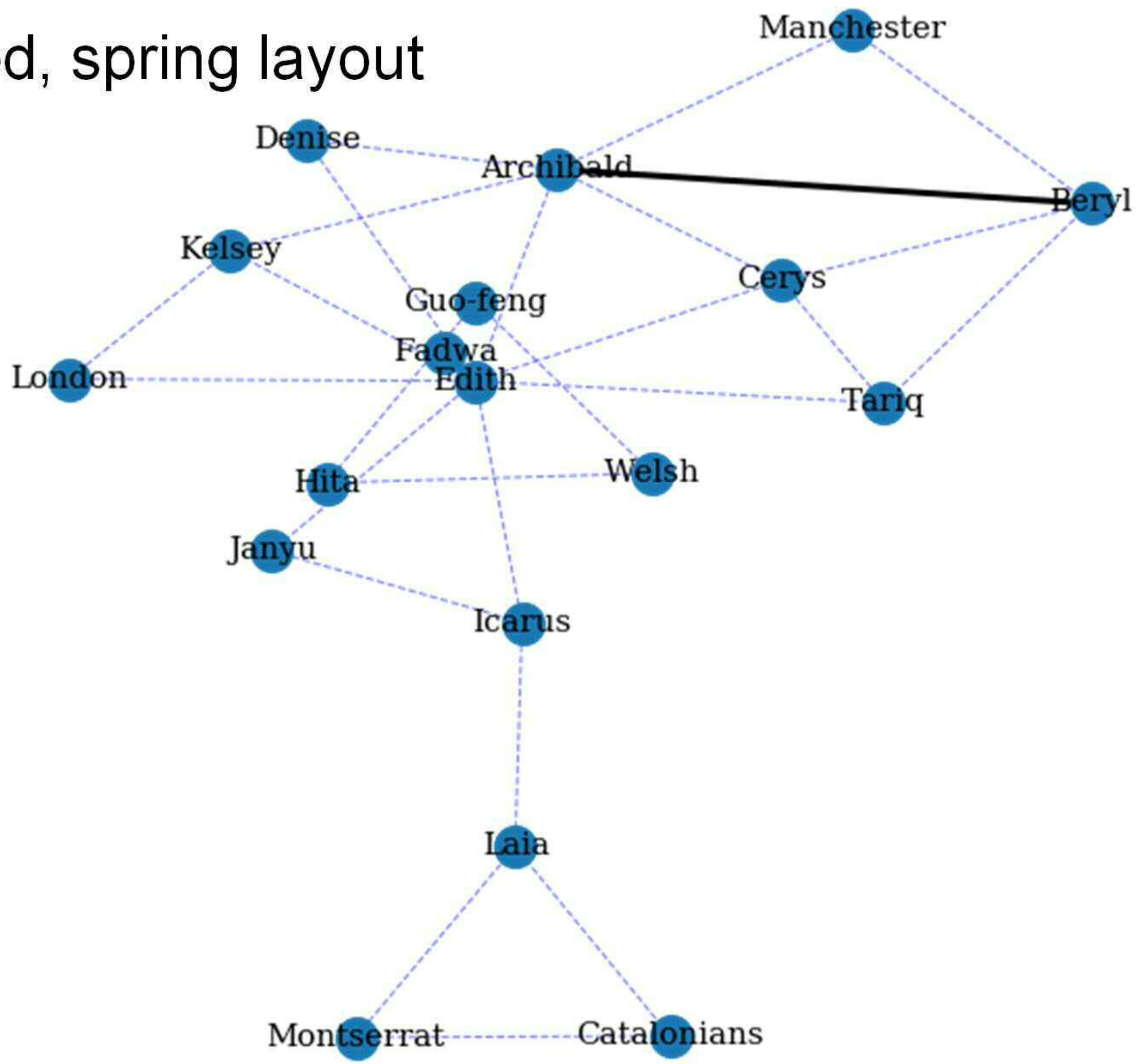
UK Data Service

Undirected, unweighted, circular layout



UK Data Service

Weighted, undirected, spring layout



UK Data Service

Links to code, python packages and resources

- https://github.com/UKDataServiceOpen/text-mining/tree/master/code
- nltk (Natural Language Toolkit) https://www.nltk.org/book/ch01.html
- nltk.corpus http://www.nltk.org/howto/corpus.html
- spaCy https://nlpforhackers.io/complete-guide-to-spacy/
- Semantic vectors package https://github.com/semanticvectors/semanticvectors/wiki
- Geometry and Meaning, by Dominic Widdows https://web.stanford.edu/group/cslipublications/cslipublications/site/1575864487.shtml
- Networkx python package https://networkx.github.io/documentation/stable/reference/index.html

Questions

Dr. J. Kasmire

julia.kasmire@manchester.ac.uk

@JKasmireComplex

UKDS

@UKDataService

UKDataService