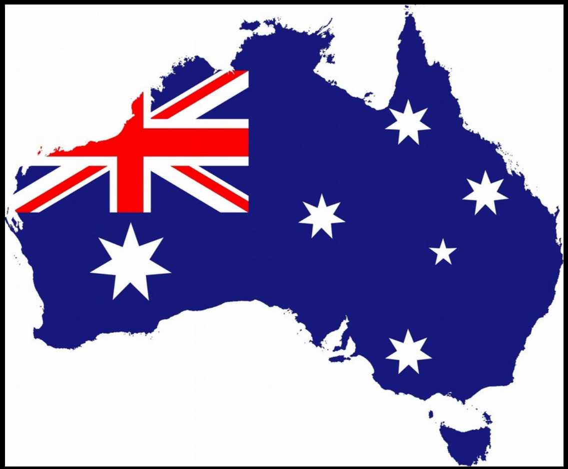
Movie review classification

- Major NLP task: sentiment classification
  - Work out the sentiment behind a piece
  - Some problems about target:
    - God damnit I dropped my iPhone
    - How does the author feel?
    - … about their iPhone
- We're going to be crude: +ve and -ve only
  - You must have an opinion!



Experimental setup

- Read in the corpus
- This is going to contain many words! $|V| \rightarrow$ a lot
- Let's get just the *k* most frequent words
  - 2000 is a nice number to start with

- Based on these, write a feature extractor that includes only popular words
  - Use set() to remove dupes

Test feature extraction

```
>>> print( document_features( movie_reviews.words( 'pos/cv957
_8737.txt' )))
{'contains(waste)': False, 'contains(lot)': False, …}
```

- So far, so good
- Let's evaluate our system
  - Build a test/train split
  - Train a classifier
  - Evaluate on the test split
- Use the .accuracy() method

What words indicate review type?

- We can look at the most informative features
- This gives words that indicate +ve/-ve reviews
- Most Informative Features

```
contains(outstanding) = True     pos : neg   =   11.1 : 1.0
contains(seagal) = True          neg : pos   =    7.7 : 1.0
contains(wonderfully) = True     pos : neg   =    6.8 : 1.0
contains(damon) = True           pos : neg   =    5.9 : 1.0
contains(wasted) = True          neg : pos   =    5.8 : 1.0
```

- Oh dear, Steven…
- What happens if we select a lot more words as features?

Data mining: N-grams

- A secondary use of this classifier:
  - Finding indicative patterns in the data

- “Seagal” could be any Seagal (probably isn't..)

- What if we knew which one it was?
  - New feature extraction code
  - Find all n-grams, for 1...k n-grams

Data mining: n-grams

```
from nltk.util import ngrams
.. everygrams
```

- See: http://www.nltk.org/api/nltk.html
- The goal is to convert:
  - [you, ai, n't, no, muslim, bruv]
  - [you, ai], [ai, n't], [n't, no], [no, muslim], [muslim, bruv]