Where Are We?

Our fundamental unit of text analysis is the document term matrix.

This is a set of stacked vectors, with each entry in each vector representing the 'amount' of a particular term.

This is could be (re-)weighted in some way (e.g. tfidf).

now cover some fundamental statistical properties of text

and think about how to compare documents, and summarize their content.

Reminder: From Texts to Numeric Data

1. collect raw text in machine readable/electronic form. Decide what constitutes a document.

2. strip away 'superfluous' material: HTML tags, capitalization, punctuation, stop words etc.
3. cut document up into useful elementary pieces: tokenization.
4. add descriptive annotations that preserve context: tagging.
5. map tokens back to common form: lemmatization, stemming.

6. operate/model.

Reminder: From Texts to Numeric Data

1. collect raw text in machine readable/electronic form. Decide what constitutes a document.

"PREPROCESSING"

6. operate/model.

Reminder: Quick Note on Terminology

a type is a unique sequence of characters that are grouped together in some meaningful way. Mostly a word (for us), but might also be a word plus punctuation, or a number etc.

e.g. '`France`', '`American Revolution`', '`1981`'

a token is a particular *instance* of type.

e.g. "`Dog eat dog world`", contains three types, but four tokens (for most purposes).

a term is a type that is part of the system's 'dictionary' (i.e. what the quantitative analysis technique recognizes as a type to be recorded etc). Could be different from the tokens, but often closely related.

e.g. stemmed word like '`treasuri`', which doesn't appear in the document itself.

Lossy Compression

- when we use the vector space model we remove some information and throw it away (e.g. word order, numbers, capitals, stop words).

→ this means we cannot restore the original representation of the data: we have lossy compression.

but presumably, life becomes a lot simpler and the tradeoff is worth it. How much simpler?

e.g. Reuters Corpus Volume 1 (RCV1) (2004) is a benchmark text collection of $\sim$ 800,000 manually coded news stories.

RCV1 has 484,494 types and 197,879,290 tokens (MR&S book, Table 5.1).

	Types	Tokens
rm numbers	473,723	179,158,204
lowercase	391,523	179,158,204
rm 150 stopwords	391,373	94,516,599
stemming	322,383	94,516,599

Heap's Law: Type-Token relationship

So preprocessing 'works' in the sense that it serves to simplify the problem.

but how does the total number of types M, change as total number of tokens T increases ?

Heap's Law: $\boxed{M = kT^b}$

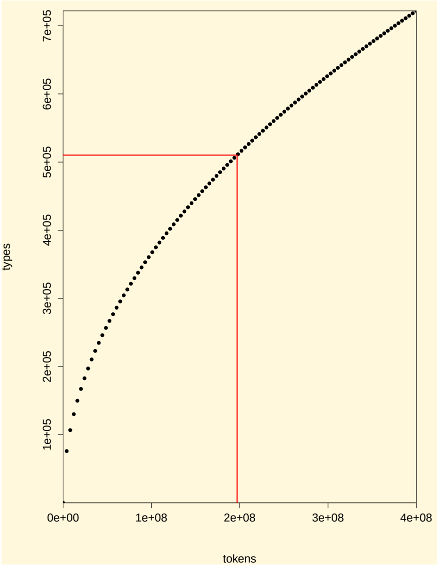
where $k \in (30, 100)$ and $b \in (0.4, 0.6)$ for English.

if we preprocess in different ways, we cause k to be different.

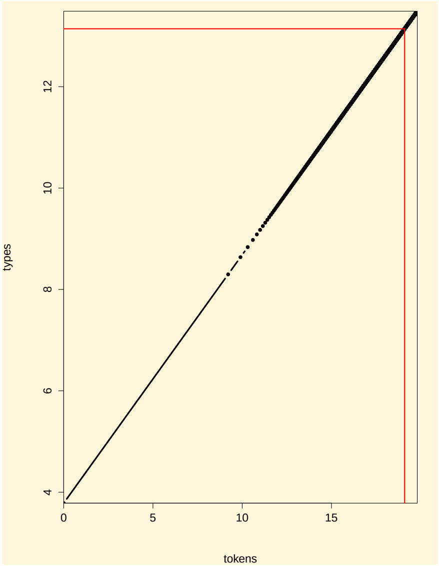
NB number of types increases rapidly at first, then less rapidly. Need to preprocess, especially for long collections!

$k = 44, b = 0.49, T = 400,000$

RCV1.



RCV1, log-log.



Zipf's Law

Heap's Law tells us about the relationship between tokens and types.

but what about the relationship between the relative frequency of terms in the corpus?

$\rightarrow$ how much more common is the most common term relative to the second common term? What about relative to the the third most common term? And the fourth...

Zipf's Law: corpus frequency of ith most common term is $\boxed{\propto \frac{1}{i}}$

so second most common term is half as common as most common,

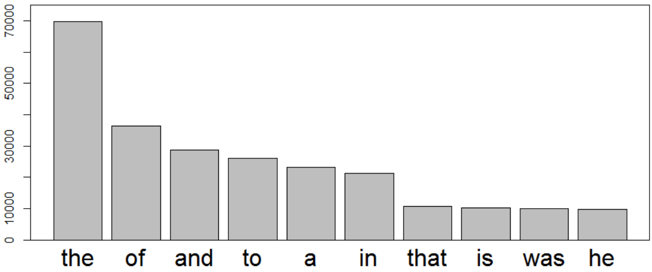
and third most common term is one third as common as most common,

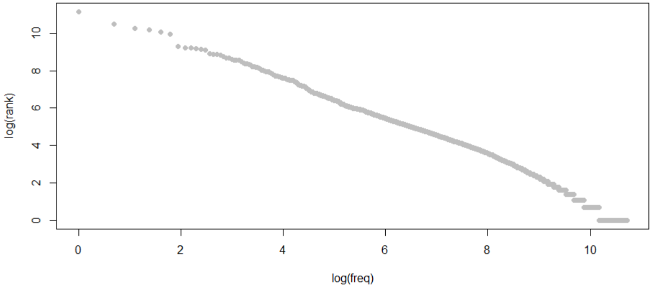
and fourth most common term is one quarter as common as most common,

etc Can rewrite as: corpus frequency of term i as ci^k or $\log(\text{corpus frequency}) = \log c + k \log i$, where i is the rank, $k = -1$.

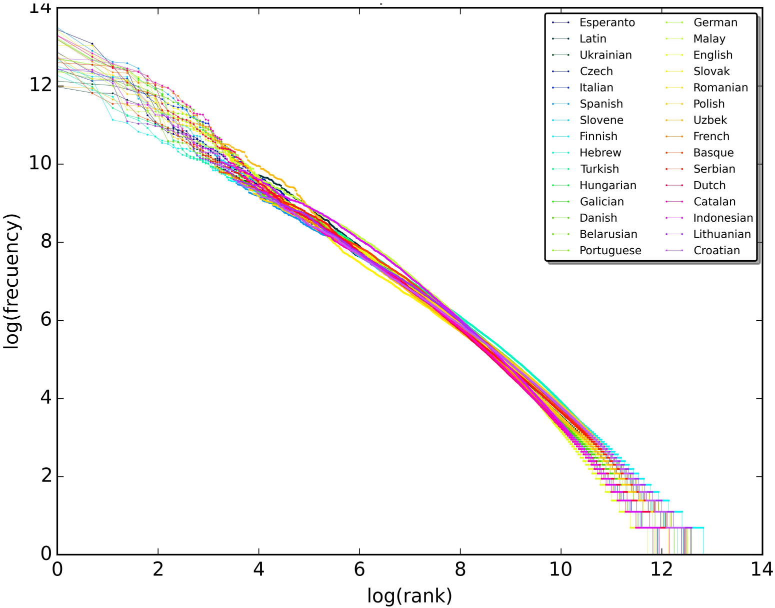
Brown Corpus (1961)

term	freq
the	69836
of	36365
and	28826
to	26126
a	23157
in	21314
that	10777
is	10182
was	9968
he	9801





Other Languages (Wikipedia)



City Populations in US (Gabaix, 1999)

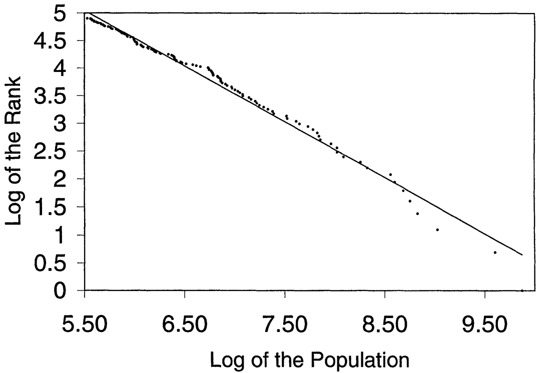


FIGURE I
Log Size versus Log Rank of the 135 largest U. S. Metropolitan Areas in 1991
Source: Statistical Abstract of the United States [1993].

Comparing Texts: Distance

Recall that the vector space model represents a document as a point in the feature space.

i.e. $\mathbf{y}_d \in \mathbb{R}^W$ is a representation of document d.

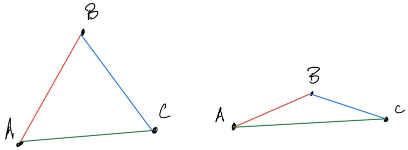
q how 'far' is that document from some other document (in the same space)?

$\rightarrow$ tells us about similarity of documents

Metrics vs Measures

NB not all measures of distance or similarity are metrics. To be a metric, the measure of *distance* between documents i and j, s_{ij} must have certain properties:

1. no negative distances: $s_{ij} \geq 0$
2. distance between documents is zero $\iff$ documents are identical
3. distance between documents is symmetric: $s_{ij} = s_{ji}$
4. measures satisfy triangle inequality. $s_{ik} \leq s_{ij} + s_{jk}$



$\overline{AC} \leq \overline{AB} + \overline{BC}$

Euclidean Distance

The 'ordinary', 'straight line' distance between two points in space.

Recall that $\mathbf{y}_i$ and $\mathbf{y}_j$ are documents,

$$\boxed{\|\mathbf{y}_i - \mathbf{y}_j\| = \sqrt{(\mathbf{y}_i - \mathbf{y}_j) \cdot (\mathbf{y}_i - \mathbf{y}_j)}}$$

e.g. $\mathbf{y}_i = [0.00, 0.00, 1.38, 1.52, 0.00]$ and $\mathbf{y}_j = [0.00, 2.13, 3.24, 0.01, 0.06]$

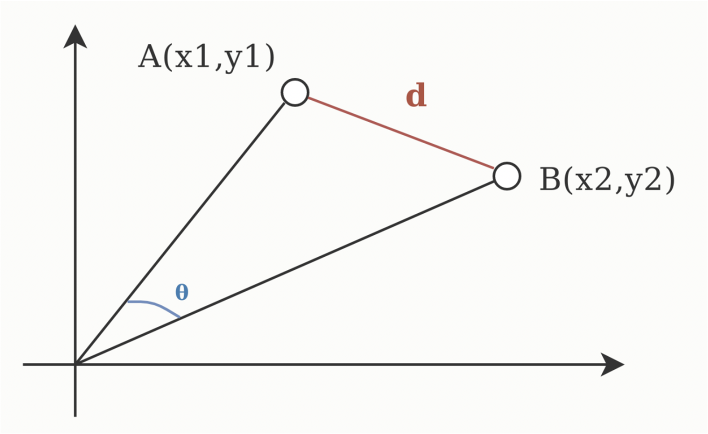
well $(\mathbf{y}_i - \mathbf{y}_j) = [0.00, -2.13, -1.86, 1.51, -0.06]$

and $(\mathbf{y}_i - \mathbf{y}_j) \cdot (\mathbf{y}_i - \mathbf{y}_j) = (0 \times 0) + (-2.13 \times -2.13) + (-1.86 \times -1.86) + (1.51 \times 1.51) + (-0.06 \times -0.06) = 10.2802$

and $\sqrt{(\mathbf{y}_i - \mathbf{y}_j) \cdot (\mathbf{y}_i - \mathbf{y}_j)} = 3.206275$

larger distances imply lower similarity.

Is Euclidean Distance what we really want?



Cosine Similarity

$$c_{ij} = \boxed{\frac{\mathbf{y}_i \cdot \mathbf{y}_j}{\|\mathbf{y}_i\| \, \|\mathbf{y}_j\|}}$$

→ we have a measure of similarity, which (since $\mathbf{y}_i$ and $\mathbf{y}_j$ are non-negative) must be between 0 and 1.

If $\mathbf{y}_i$ and $\mathbf{y}_j$ are vectors, c_{ij} is the cosine of the angle between them.

and document length is controlled for.

so intuitively, cosine similarity captures some notion of relative 'direction' (e.g. style or topics in the document) rather than 'magnitude' (distance from origin). Is the Pearson correlation between two vectors that have been demeaned.

Example

$$\mathbf{y}_i = [2.3, 4.3];\ \mathbf{y}_j = [3.9, 2.1]$$

then $\mathbf{y}_i \cdot \mathbf{y}_j = [2.3 \times 3.9] + [4.3 \times 2.1] = 18.$

and $||\mathbf{y}_i|| = \sqrt{2.3^2 + 4.3^2} = 4.88;\ ||\mathbf{y}_j|| = \sqrt{3.9^2 + 2.1^2} = 4.43$

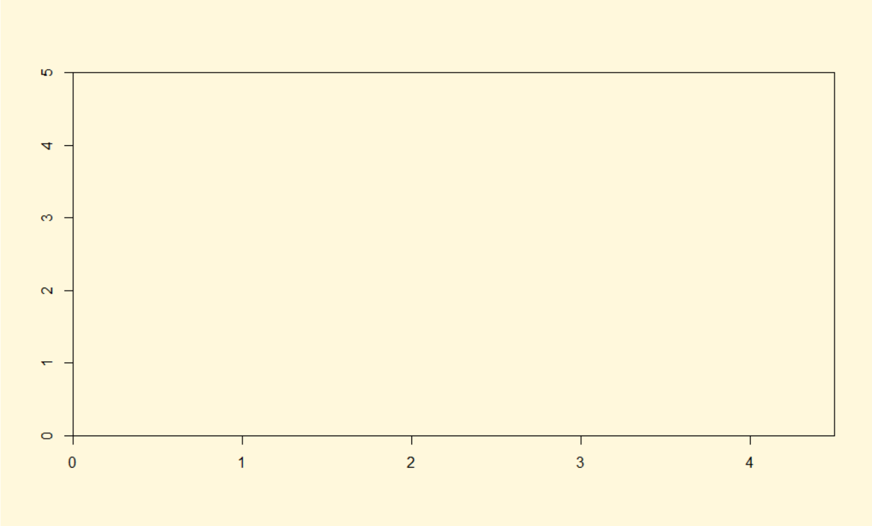
so $c_{ij} = \frac{18}{4.88 \times 4.43} = 0.83.$

Graphically

$$\mathbf{y_i} = [2.3, 4.3];\ \mathbf{y_j} = [3.9, 2.1]$$

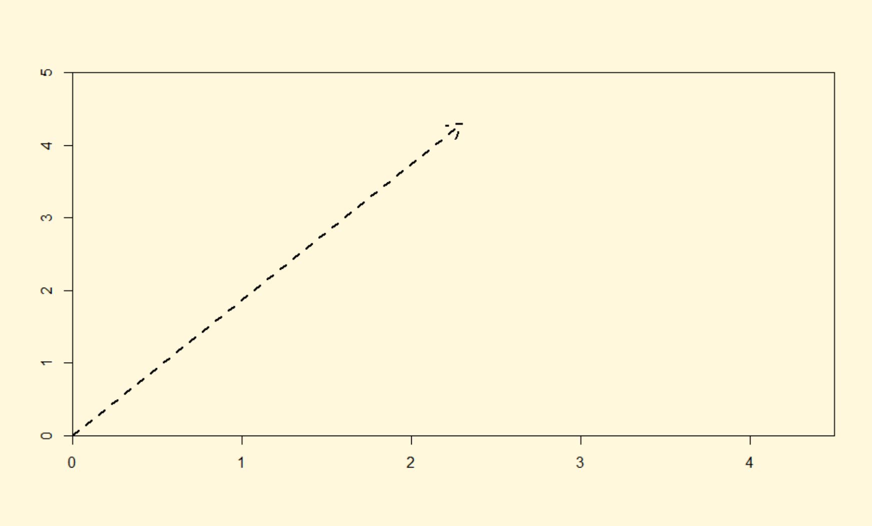
Graphically

$$\mathbf{y_i} = [2.3, 4.3]; \ \mathbf{y_j} = [3.9, 2.1]$$



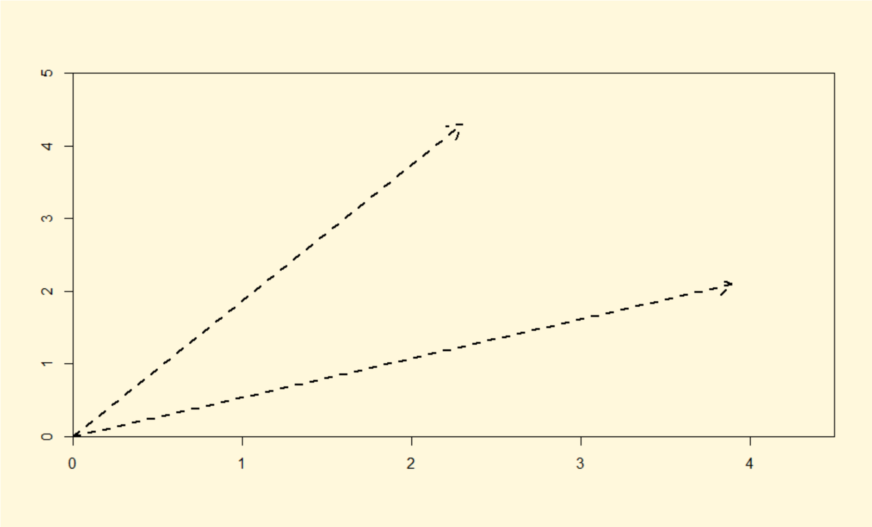
Graphically

$\mathbf{y_i} = [2.3, 4.3]$; $\mathbf{y_j} = [3.9, 2.1]$



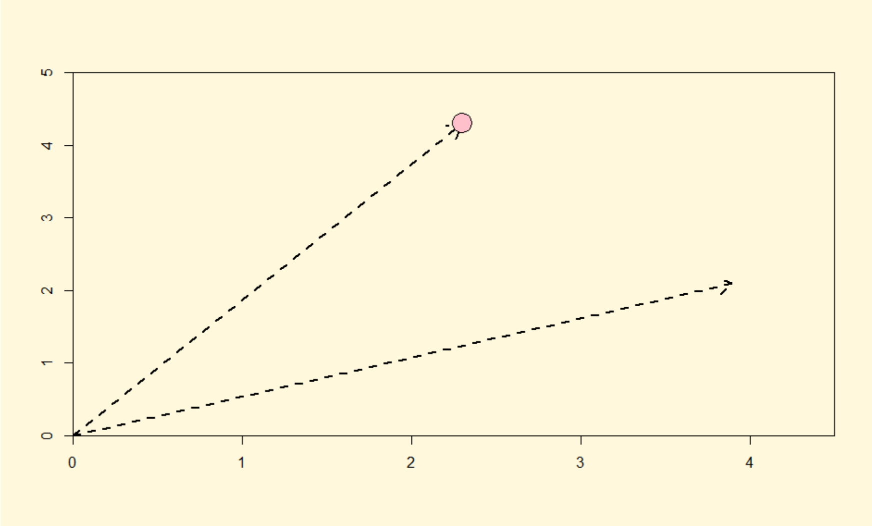
Graphically

$\mathbf{y_i} = [2.3, 4.3]$; $\mathbf{y_j} = [3.9, 2.1]$



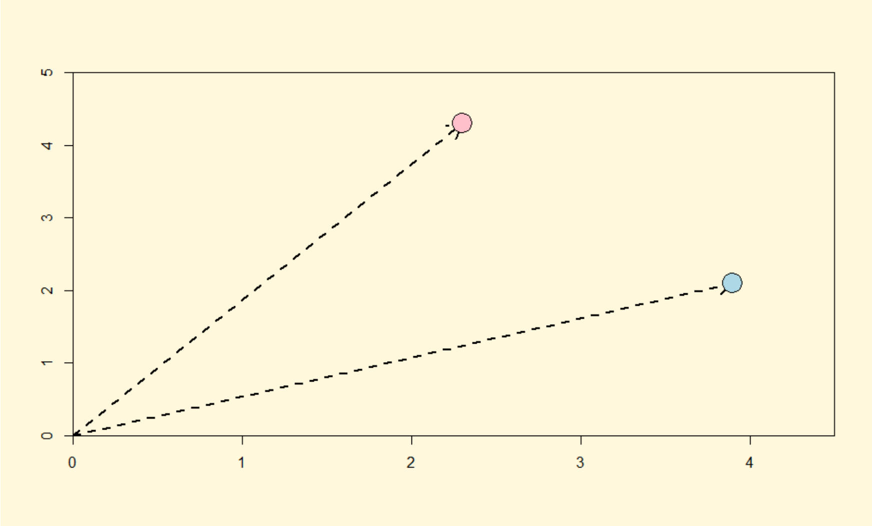
Graphically

$\mathbf{y_i} = [2.3, 4.3]$; $\mathbf{y_j} = [3.9, 2.1]$



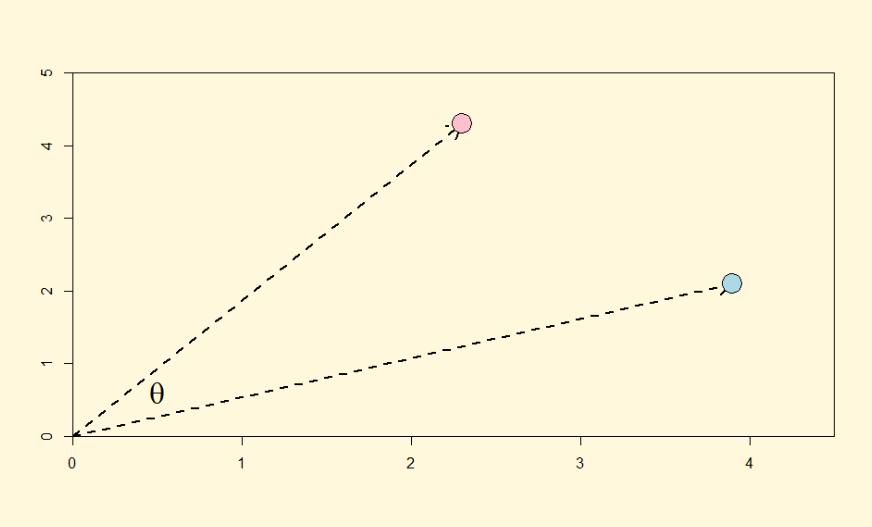
Graphically

$\mathbf{y_i} = [2.3, 4.3]$; $\mathbf{y_j} = [3.9, 2.1]$

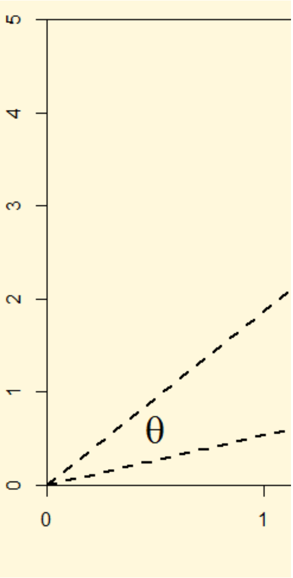


Graphically

$\mathbf{y_i} = [2.3, 4.3]$; $\mathbf{y_j} = [3.9, 2.1]$



Algebra



$\rightarrow$ $\cos\theta = \frac{\mathbf{y}_i \cdot \mathbf{y}_j}{||\mathbf{y}_i||||\mathbf{y}_j||} = 0.83$

and $\theta = \arccos\left(\frac{\mathbf{y}_i \cdot \mathbf{y}_j}{||\mathbf{y}_i||||\mathbf{y}_j||}\right)$.

so $\theta = \arccos(0.83) = 0.59$

$\rightarrow$ $0.59 \times \frac{180}{\pi} = 33.80°$.

Looks about right.

1983 General Election Manifestos

- Labour manifesto as 'longest suicide note in history'
- unilateral nuclear disarmament, withdrawal from the EEC, abolition of the Lords, re-nationalisation

- Conservative manifesto promised trade union curbs, deflation etc.

$$\boxed{c_{ij} \approx 0.70}$$

1997 General Election Manifestos

- Conservative manifesto promised continuation of moderate Major years.

- 'New Labour' and 'Third Way'
- committed to Conservative spending plans (for next two years), no income tax rises.

$$\boxed{c_{ij} \approx 0.90}$$

Animals at the Zoo

- we can produce a cosine dissimilarity measure via $1 - c_{ij}$ (though not a metric)

but there are a large number of other distance measures on offer:

Jaccard: size of the intersection of the two documents (number of common words between the documents) divided by the size of the union of the two documents (total number of unique words in docs).

Manhattan: known as 'taxicab' distance or 'city block' distance. Absolute difference between coordinates: $||\mathbf{y}_i - \mathbf{y}_j|| = \sum |\mathbf{y}_i - \mathbf{y}_j|$. As we go from $\mathbf{y}_i$ to $\mathbf{y}_j$, have to do so at right angles: travel along, turn 90° and then up (or down), then turn 90° and go along, turn 90° etc.

Canberra: weighted version of Manhattan distance. $\sum \frac{|\mathbf{y}_i - \mathbf{y}_j|}{|\mathbf{y}_i| + |\mathbf{y}_j|}$

Minowski: generalized version of Euclidean and Manhattan. $\left(\sum |\mathbf{y}_i - \mathbf{y}_j|^c\right)^{\frac{1}{c}}$. If c is 1, this is Manhattan. If c is 2, this is Euclidean.

etc

Edit Distance

Governments routinely ask citizens, firms and other stake-holders for feedback on proposed rules: this is given as comments.

We may want to know how much different rules in different agencies at different times change in response to consultation: look at *edit distance*.

see Rashin (2019) for more elaborate ideas

Edit Distance: An Example

original, s_1	`one million dollar limit for licensees in easternmost Florida`
final, s_2	`one billion dollar limit for licensees in southern Florida`

Suppose we can do three things:

1. insert a character into a string
2. delete a character from a string
3. replace a character in a string by another character

The smallest number of operations taking us from s_1 to s_2 is the **Levenshtein distance** between those strings.

Levenshtein in Action

$s_1 =$ easternmost

$s_2 =$ southern

1 delete m, delete o, delete s, delete t. → eastern

2 insert h. → easthern

3 replace e, a and s with s, o and u. → southern.

How many operations? $4 + 1 + 3 = 8$. **Levenshtein distance** is 8.

Notes on Edit Distances

Calculating the optimal (number of) steps is not trivial: famous dynamic programming problem.

If all the operations have similar weight the distance is symmetric : same whether you are going $s_1 \rightarrow s_2$ or $s_2 \rightarrow s_1$.

In applications we might choose to give types of edit operations different weights

e.g. could imagine it's easier to go from `million` to `millions` than from `million` to `billion`, even though these both require one operation.

Different types of edit distances allow different types of operations.