In general, we will. . .

Get Texts → Document Term Matrix → Operate → Inference

An expert hospital consultant has written to my hon. Friend...

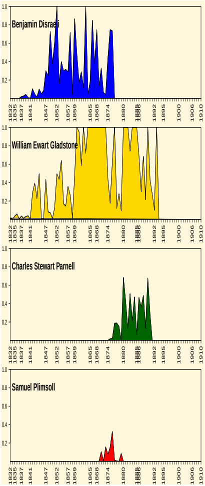
Order. The Minister must be allowed to reply without interruption.

I am grateful to my hon. Friend for her question. I pay tribute to her work with the International Myeloma Foundation...

My constituent, Brian Jago, was fortunate enough to receive a course of Velcade, as a result of which he does not have to...

$$
\begin{array}{c} \\ MP_{001} \\ MP_{002} \\ \\ MP_i \\ \\ MP_{654} \\ MP_{655} \end{array}
\begin{array}{c}
\begin{array}{cccc} a & an & \dots & ze \end{array} \\
\begin{pmatrix}
2 & 0 & \dots & 1 \\
0 & 3 & \dots & 0 \\
\vdots & \vdots & & \vdots \\
\vdots & \vdots & \dots & \vdots \\
\vdots & \vdots & \ddots & \vdots \\
0 & 0 & \dots & 2
\end{pmatrix}
\end{array}
$$

- (dis)similarity
- diversity
- readability
- scale
- classify
- topic model
- burstiness
- sentiment
- ...



I. Defining the Corpus

defn (typically) large set of texts or documents which we wish to analyze.

→ how large? if small enough to read in reasonable time, you should probably just do that.

'structured', in the sense that you know what the documents are, where they begin and end, who authored them etc.

'unstructured data' in sense that what is wanted (e.g. ideological position) may not be directly observable.

may be annotated in sense that metadata —data that is not part of the document itself—is available. Examples include markup, authorship and date information, linguistic tagging (more below)

e.g. court transcripts, legislative records, Twitter feeds, etc.

Sampling

The corpus is made up of the documents within it, but these may be a sample of the total population of documents available.

We sample for reasons of time, resources or (legal) necessity.

e.g. Twitter gives you $\sim 1\%$ of all their tweets, but it would presumably be prohibitively expensive to store 100%.

Often, authors claim to have the universe of cases in their corpus: *all* press releases, *all* treaties, *all* debate speeches.

$\rightarrow$ depending on your philosophical position, you still need to think about sampling error. This is because there exists a superpopulation of populations from which the universe you observed came from.

Random error may not be the only concern: corpus should be representative in some well defined sense for inferences to be meaningful.

II. Reducing Complexity

- language is extraordinarily complex, and involves great subtlety and nuanced interpretation.

but remarkably, we can do very well by simplifying, and representing documents as straightforward mathematical objects.

→ makes the modeling problem much more tractable.

by 'do very well', we mean that much more complicated representations add (almost) nothing to the quality of our inferences, our ability to predict outcomes, and the fit of our models.

NB inevitably, the degree to which one simplifies is dependent on the particular task at hand.

→ there is no 'one best way' to go from texts to numeric data. Good idea to check sensitivity.

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.

From Texts to Numeric Data

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

"PREPROCESSING"

6. operate/model.

‘superfluous’ material: control characters and punctuation

- generally think control characters—non-printing, but cause the document to look different—like `\n` (“new line” in C, Java, or Perl), do not connote much that is of substantive importance.

→ remove them. Same for underlining or **emboldening**.

- punctuation may also be unhelpful

  are `wash`, `wash.`, `wash,`, `wash)` really different words?

But one has to think about the problem at hand...

what to do depends on what language features you are most interested in.

if — the grammatical structure of sentences matters, makes sense to keep most, if not all, punctuation.

e.g. — social media: does use of `!` differ by age group?

but — mostly just interested in coarse features (such as word frequencies); converting most punctuation to whitespace is quick and better than keeping it.

NB — 'dictionaries' can be used to map contractions back to their component parts

e.g. — tell us that `won't` could be `will not`

but — may not be as important as you think.

‘superfluous’ material: capitalization

Federalist 1

```
The subject speaks its own importance; comprehending in its consequences
nothing less than the existence of the UNION, the safety and welfare of
the parts of which it is composed, the fate of an empire in many respects
the most interesting in the world.
```

- is the one use of ‘`The`’ the same word as the seven uses of ‘`the`’?
- is ‘`UNION`’ the same word as ‘`union`’ and ‘`Union`’ as used elsewhere in this essay?
- yes $\rightarrow$ lowercase (uppercase) everything
- or keep lists (dictionary) of proper nouns, lowercase everything else
- or lowercase words at the beginning of a sentence leave everything else as is

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.

Tokens and tokenization

The text is now 'clean', and we want to pull out the meaningful subunits—the tokens. We will use a tokenizer.

→ usually the tokens are words, but might include numbers or punctuation too.

Common rule for a tokenizer is to use whitespace as the marker.

but given application might require something more subtle

e.g. "`Brown vs Board of Education`" may not be usefully tokenized as '`Brown`', '`vs`', '`Board`', '`of`', '`Education`'

Exceptions and Other Ideas

In some languages, tokenizing is a non-trivial problem because whitespace may not be used:

问世间情是何物，直教生死相许。
天南地北双飞客，老翅几回寒暑。

We may want to deal directly with multiword expressions in some contexts. There are rules which help us identify them relatively quickly and accurately.

e.g. ‘`White House`’ , ‘`traffic light`’

NB these words mean something ‘special’ (and slightly opaque) when combined. Related to idea of collocations: words that appear together more often than we'd predict based on random sampling.

Removing Stop Words

There are certain words that serve as linguistic connectors ('function words') which we can remove.

→ this simplifies our document considerably, with little loss of substantive 'content'. Indeed, search engines often ignore them.

There are many lists available, and we may add to them in an application specific way.

e.g. working with Congressional speech data, '`representative`' might be a stop word.

NB in some specific applications, function word usage **is** important—we'll discuss this when we deal with authorship attribution.

Some stop words

a	about	above	after	again	against	all
am	an	and	any	are	aren't	as
at	be	because	been	before	being	below
between	both	but	by	can't	cannot	could
couldn't	did	didn't	do	does	doesn't	doing
don't	down	during	each	few	for	from
further	had	hadn't	has	hasn't	have	haven't
having	he	he'd	he'll	he's	her	here
here's	hers	herself	him	himself	his	how
how's	i	i'd	i'll	i'm	i've	if
in	into	is	isn't	it	it's	its
itself	let's	me	more	most	mustn't	my
myself	no	nor	not	of	off	on
once	only	or	other	ought	our	ours
ourselves	out	over	own	same	shan't	she
she'd	she'll	she's	should	shouldn't	so	some
such	than	that	that's	the	their	theirs
them	themselves	then	there	there's	these	they
they'd	they'll	they're	they've	this	those	through
to	too	under	until	up	very	was
wasn't	we	we'd	we'll	we're	we've	were
weren't	what	what's	when	when's	where	where's
which	while	who	who's	whom	why	why's
with	won't	would	wouldn't	you	you'd	you'll
you're	you've	your	yours	yourself	yourselves	

Tagging

- **so far** tokens are on even footing—no distinctions drawn between nouns, verbs, nouns acting as subjects, nouns acting as objects, etc.
- **and** for many applications, this information doesn't help very much (e.g. for classification).

- **but** in other applications we may really want to know information about the part-of-speech this word represents. We want to disambiguate in what sense a term is being used.
- **e.g.** in 'events' studies, when we are recording who did what to whom: '`the UK bombing will force ISIS to surrender`'. Here `force` is a verb, not a noun.

→ annotating in this way is called parts-of-speech tagging.

Penn POS Tagger

Number	Tag	Description
1.	CC	Coordinating conjunction
2.	CD	Cardinal number
3.	DT	Determiner
4.	EX	Existential *there*
5.	FW	Foreign word
6.	IN	Preposition or subordinating conjunction
7.	JJ	Adjective
8.	JJR	Adjective, comparative
9.	JJS	Adjective, superlative
10.	LS	List item marker
11.	MD	Modal
12.	NN	Noun, singular or mass
13.	NNS	Noun, plural
14.	NNP	Proper noun, singular
15.	NNPS	Proper noun, plural
16.	PDT	Predeterminer
17.	POS	Possessive ending
18.	PRP	Personal pronoun
19.	PRP$	Possessive pronoun
20.	RB	Adverb
21.	RBR	Adverb, comparative
22.	RBS	Adverb, superlative
23.	RP	Particle
24.	SYM	Symbol
25.	TO	*to*
26.	UH	Interjection
27.	VB	Verb, base form
28.	VBD	Verb, past tense
29.	VBG	Verb, gerund or present participle
30.	VBN	Verb, past participle
31.	VBP	Verb, non-3rd person singular present
32.	VBZ	Verb, 3rd person singular present
33.	WDT	Wh-determiner
34.	WP	Wh-pronoun
35.	WP$	Possessive wh-pronoun
36.	WRB	Wh-adverb

Stemming and Lemmatization

Documents may use different forms of words (‘`jumped`’, ‘`jumping`’, ‘`jump`’), or words which are similar in concept (‘`bureaucratic`’, ‘`bureaucrat`’, ‘`bureaucratization`’) as if they are different tokens.

→ we can simplify considerably by mapping these variants (back) to the same word.

- Stemming does this using a crude (heuristic) which just ‘chops off’ the affixes. It returns a stem which might not be a dictionary word.
- Lemmatization does this using a vocabulary, parts of speech context and mapping rules. It returns a word in the dictionary: a lemma.

e.g. For the word “studies” stemming would return “studi”. Lemmatization would return “study”.

Stemming

Though technically incorrect, 'stemming' and 'lemmatization' often used interchangeably.

For small examples, one can use a 'look up' table: table listing what a given realization of a word should be mapped to.

btw we sometimes use 'equivalency classes' meaning that an internal thesaurus maps different words back to the same type of word: e.g. '`rightwing`' and '`republican`' to '`conservative`'.

In practice, need something faster (and cruder), e.g. a Porter Stemmer algorithm using the `Snowball` compiler.

Snowball examples

Original Word		Stemmed Word
abolish	$\mapsto$	`abolish`
abolished	$\mapsto$	`abolish`
abolishing	$\mapsto$	`abolish`
abolition	$\mapsto$	`abolit`
abortion	$\mapsto$	`abort`
abortions	$\mapsto$	`abort`
abortive	$\mapsto$	`abort`
treasure	$\mapsto$	`treasure`
treasured	$\mapsto$	`treasure`
treasures	$\mapsto$	`treasure`
treasuring	$\mapsto$	`treasure`
treasury	$\mapsto$	`treasuri`

NYT

```
Emergency measures adopted for Beijing’s first ‘‘red alert" over air
pollution left millions of schoolchildren cooped up at home, forced
motorists off the roads and shut down factories across the region on
Tuesday, but they failed to dispel the toxic air that shrouded the Chinese
capital in a soupy, metallic haze.
```

marked up

```
Emergenc y  measur es  adopt ed  for Beij ing  s first red alert over
air pollut ion  left million s  of schoolchildren coop ed  up at home,
forc ed  motorist s  off the road s  and shut down factor ies  across
the region on Tuesday, but they fail ed  to dispel the toxic air that
shroud ed  the Chines e  capit al  in a soupy, metal lic  haze.
```

NYT

Emergency measures adopted for Beijing's first \red alert" over air pollution left millions of schoolchildren cooped up at home, forced motorists off the roads and shut down factories across the region on Tuesday, but they failed to dispel the toxic air that shrouded the Chinese capital in a soupy, metallic haze.

Stemmed

Emergenc measur adopt for Beij s first red alert over air pollut left million of schoolchildren coop up at home forc motorist off the road and shut down factori across the region on Tuesdai but thei fail to dispel the toxic air that shroud the Chines capit in a soupi metal haze.

We Don't Care about Word Order

We have now preprocessed our texts.

Generally, we are willing to ignore the order of the words in a document. This considerably simplifies things. And we do (almost) as well without that information as when we retain it.

NB we are treating a document as a $\boxed{\text{bag-of-words}}$ (BOW).

btw, we keep multiplicity—i.e. multiple uses of same token

e.g. "`The leading Republican presidential candidate has said Muslims should be banned from entering the US.`"

→ "`lead republican presidenti candid said muslim ban enter us`"

= "`us lead said candid presidenti ban muslim republican enter`"

Could we retain Word Order?

- **for** some applications, retaining word order is very important.
- **e.g.** we have a large number of multiword expressions or named entities like ‘`Bill Gates`’
- **e.g.** we think some important subtlety of expression is lost: negation perhaps— “`I want coffee, not tea`” might be interpreted very differently without word order.

→ can use *n*-grams, which are (sometimes contiguous) sequences of two (bigrams) or three (trigrams) tokens. This makes computations considerably more complex.

original/some pre-processing

```
a military patrol boat rescued three of the kayakers on general carrera lake and a
helicopter lifted out the other three the chilean army said
```

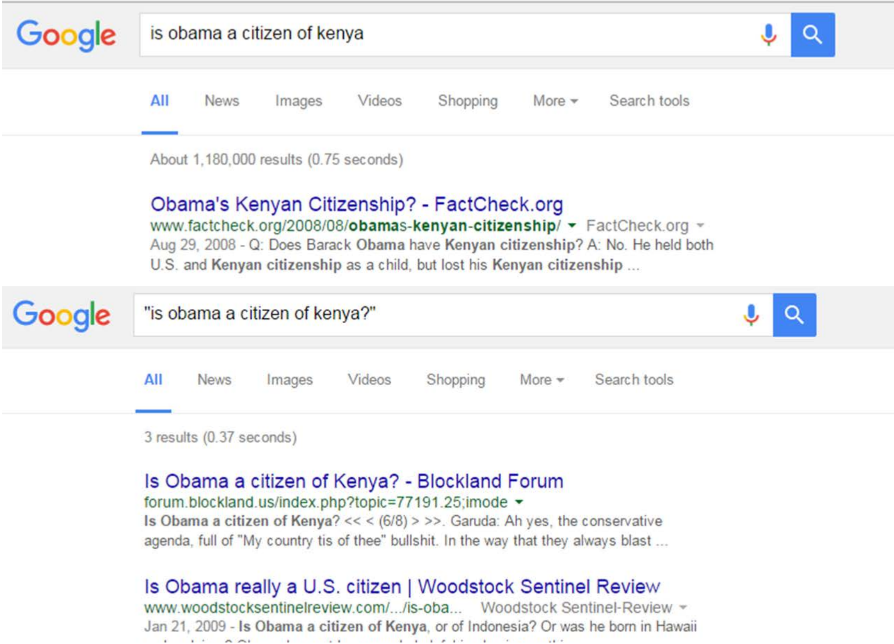
bigrams

```
"a military" "military patrol" "patrol boat" "boat rescued" "rescued three" "three of"
"of the" "the kayakers" "kayakers on" "on general" "general carrera" "carrera lake"
"lake and" "and a" "a helicopter" "helicopter lifted" "lifted out" "out the" "the
other" "other three" "three the" "the chilean" "chilean army" "army said"
```

trigrams

```
"a military patrol" "military patrol boat" "patrol boat rescued" "boat rescued three"
"rescued three of" "three of the" "of the kayakers" "the kayakers on" "kayakers on
general" "on general carrera" "general carrera lake" "carrera lake and" "lake and a"
"and a helicopter" "a helicopter lifted" "helicopter lifted out" "lifted out the" "out
the other" "the other three" "other three the" "three the chilean" "the chilean army"
"chilean army said"
```

Very similar documents may not share short n-grams



III. Vector Space Model

We can think about a document as being a collection of W features (tokens, words etc)

if each feature can be placed on the real line, then the document can be thought of as a point $\mathbb{R}^W$.

e.g. "`Bob goes home`" can be thought of a vector in 3 dimensions: one corresponds to how '`Bob`'-ish it is, one corresponds to how '`goes`'-ish it is, one corresponds to how '`home`'-ish it is.

Features will typically be the *n*-gram (mostly unigram) frequencies of the tokens in the document, or some function of those frequencies.

e.g. '`the cat sat on the mat`' becomes `(2,1,1,1,1)` if we define the dimensions as `(the, cat, sat, on, mat)` and use simple counts.

Notation and Terminology

$d = 1, \ldots, D$ indexes documents in the corpus

$w = 1, \ldots, W$ indexes features found in documents

$\mathbf{y}_d \in \mathbb{R}^W$ is a representation of document d in a particular feature space

so each document is now a vector, with each entry representing the frequency of a particular token or feature. . .

$\rightarrow$ stacking those vectors on top of each other gives the document term matrix (DTM) or the document feature matrix (DFM).

$\rightarrow$ taking the transpose of the DTM gives the term document matrix (TDM) or feature document matrix (FDM).

partial DTM from Roosevelt's Inaugural Addresses

```
                features
docs                     american expect induct presid will
  1933-Roosevelt                2      1      1      1   12
  1937-Roosevelt                4      0      0      2   16
  1941-Roosevelt                4      0      0      1    4
  1945-Roosevelt                1      0      0      1    7
```

partial TDM from Roosevelt's Inaugural Addresses

```
          docs
features   1933-Roosevelt 1937-Roosevelt 1941-Roosevelt 1945-Roosevelt
  american              2              4              4              1
  expect                1              0              0              0
  induct                1              0              0              0
  presid                1              2              1              1
  will                 12             16              4              7
```

IV. Weighting

To this point, we have been constructing the document vectors as counts. More formally, this is term frequency, since it simply records the number of occurrences of a given term.

but this implies that all words are of 'equal importance'. This is a problem in some domains

e.g. almost every article in political science will mention 'politics', but that suggests they are all more similar than they really are (and makes it hard to find 'different' ones).

so we may want to do something that throws certain feature relationships into starker relief.

along with term frequency, we may want to consider document frequency: the number of documents in which this word appears.

Introducing tf-idf

- tf_{dw}, term frequency: number of times word w appears in document d
- df_w, document frequency: number of documents in the collection of documents that contain word w

- $\ln \frac{|D|}{df_w}$, inverse document frequency: (natural) log of the total size of the corpus $|D|$ divided by the number of documents in the collection of documents that contain word w. When the word is common in the corpus, this will be a small number. When the word is rare, this will be a large number.

$$\boxed{tf_{dw} \cdot \ln \frac{|D|}{df_w}\text{, term frequency-inverse document frequency: tf-idf.}}$$

tf-idf

$$\boxed{tf_{dw} \cdot \ln \frac{|D|}{df_w}\text{, term frequency-inverse document frequency: tf-idf.}}$$

→ when a word is common in a given document, but rare in the corpus as whole, this means tf is high and idf is high. So presence of that word is indicative of difference, and it is weighted up.

but if word is common in a given document, and common in the corpus, tf is high, but idf are low. So term is weighted down, and filtered out.

and very low for words occurring in every document: least discriminative words.

Example: FDR corpus

FDR used ‘`will`’ 12 times in his 1933 speech. So, tf=12.

and in his 4 speeches (our corpus), he used it (at least once) in *every* speech. So, $|D| = 4$ and $df = 4$

so the idf is $\ln \frac{|D|}{df} = \ln\left(\frac{4}{4}\right) = 0$

→ tf-idf=0 for ‘`will`’ in 1933.

but he used ‘`expect`’ once in 1933, and he didn’t use it any other speech.

so idf is $\ln \frac{|D|}{df} = \ln\left(\frac{4}{1}\right) = 1.38$

→ tf-idf=1.38 for ‘`expect`’ in 1933.

→ ‘`expect`’ helps us discriminate better than ‘`will`’.

Notes on a DTM

the way we construct the DTM— including order/nature of pre-processing—is application specific.

NB DTM tends to be sparse: contains lots of (mostly) zeros.

- partly a consequence of language itself: people say things in idiosyncratic ways.
- partly a consequence of reweighting: taking log(1).

in some applications, we might remove sparse terms—tokens that occur in very few docs.

NB there are efficient ways to store and manipulate sparse matrices.