TEXAS McCombs

Text as data

David Puelz

Outline

Big data: text

Text is everywhere. It provides a key source of insight in the modern data science environment.

Examples:

- Product reviews.
- Internet searches.
- Social-media activity for a brand. Company earnings calls.
- Technical documentation.

Big data: text

Annoying reality: text data is **unstructured**.

Compare:

Price	Engine	Cylinders	HP	CityMPG
43755	39014	3.5	6	225
46100	41100	3.5	6	225
36945	33337	3.5	6	265
89765	79978	3.2	6	290
23820	21761	2	4	200
33195	30299	3.2	6	270
26990	24647	2.4	4	200
25940	23508	1.8	4	170

vs.

It is not from the benevolence of the butcher, the brewer, or the baker, that we expect our dinner, but from their regard to their own interest. We address ourselves, not to their humanity but to their self-love, and never talk to them of our own necessities but of their advantages. Nobody but a beggar chooses to depend chiefly upon the benevolence of his fellow-citizens.

Text-analysis goals

Summary/compression:

→ Representing a body of documents as categories
→ Representing a document with a topic or set of topics

Find similar documents to a given document:

→ Showing users items of similar interest
→ Drawing analogies, e.g. from search terms to ads

Classification of documents:

→ Author attribution

Some terminology

Corpus/corpora: body of documents

→ Wikipedia (each page is a document)
→ All tweets on a given date (each tweet is a document)

Dictionary: set of all allowable “words”

→ n-gram: set of n words in a row
→ “White House”: a bi-gram (see Google’s n-gram browser)

Some terminology

NLP: natural language processing

Text pipeline: how unstructured text becomes tidy data

Token versus type:

→ Token: a string with an identified meaning.
→ Type: a higher-order category representing a concept.

Example: "A rose is a rose is a rose." – Sylvia Plath

9 tokens, 4 types ("a", "rose", "is", ".")
Programming analogy: type = class, token = object.

Some terminology

Tokenization: turning a string of symbols into tokens

Metadata: extra information about a document

Examples: author, geo-tag

XML/JSON: two most common file formats for text data. More structured than .txt. Includes metadata

Information in a document

- Metadata
- Words in a document
  - Can't describe everything, but it can get pretty far
- Sentiment
  - Computers may make mistakes in tagging sentiment ..
    Example: I'm having such a great time today at the DMV!
- Main idea/topics
- Grammar/syntax

Tokenization and representation

String of symbols

↓ tokenization

String of tokens

↓ representation

Useful data structure

It ain't over til it's over.

↓

"It", "ain't", "over", "til", "it's", "over", "."

(not: "I", "taint", "overt", "tilits", "over", "."

↓

It: 1

ain't: 1

over: 2

etc

Tokenization: from symbols to tokens

- Tokenization involves many choices of what to do and not do to a raw string of symbols.
- Removing/splitting on white spaces
  - Typically the initial step in tokenization
  - Difficult when words are run together (e.e. cummings poem; "onetwothreefourfive")
- Removing punctuation
  - But be careful, e.g. :-), ->

Tokenization: from symbols to tokens

- Converting everything to lowercase
- Removing “stop” words
  - the, is, a, of...
  - Take care: one person’s trash is another person’s treasure (classifying the federalist papers)
- Dropping numbers and mapping to a common symbol (NUM)
- Stemming: drop suffixes
  - acknowledge, acknowledges, acknowledgement
- Deal with misspellings

Tokenization to data structure ("Bag of words")

- This shows only the words in a document, and nothing about sentence structure or organization.

> *"There is a tide in the affairs of men, which taken at the flood, leads on to fortune. Omitted, all the voyage of their life is bound in shallows and in miseries. On such a full sea are we now afloat. And we must take the current when it serves, or lose our ventures."*

- What the data scientist sees:

tide: 1	flood: 1
affairs: 1	the: 4
men: 1	fortune: 1
we: 2	etc.

Tokenization to data structure ("Bag of words")

Advantage: easy to work with and calculate with

Disadvantage: it destroys other important sources of information (syntax, structure)

We can get surprisingly far with the BoW representation

Two common options for data structures:

$\rightarrow$ Hash table/key-value store (dictionary in Python)
$\rightarrow$ Vectors

Bag of words as a hash table

Sometimes called a key-value store ...

- Keys: words
- Values: counts in the document
- Easy to add new entry to hash table
- Useful for open-ended vocabulary
- Generally preferred for storage/manipulation

```
tide: 1          flood: 1
affairs: 1       the: 4
men: 1           fortune: 1
we: 2            ...
```

Bag of words as a vector

- Each index in a vector corresponds to a word; each entry is word count
- Can be difficult to update if a new word is encountered
- Easier to do math with
- Two examples of document vectors

tide	affairs	the	. . .	bacteria	cars	ain't	over	til
1	1	4	. . .	0	0	0	0	0
0	0	0	. . .	0	0	1	2	2

(most entries not shown)

Document-term matrix (DTM)

N = number of documents (rows)

D = size of dictionary (columns)

X_{ij} = number of times term j appears in document i.

The document-term matrix X for a corpus puts each document's vector as a row of the matrix:

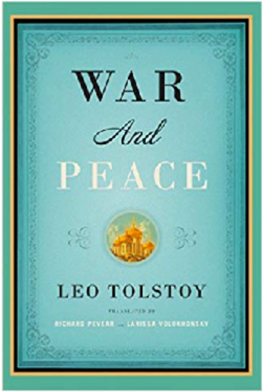
```
(X11 X12 ... X1D)
(X21 X22 ... X2D)
(X31 X32 ... X3D)
      . . .
(XN1 XN2 ... XND)
```

What's next?

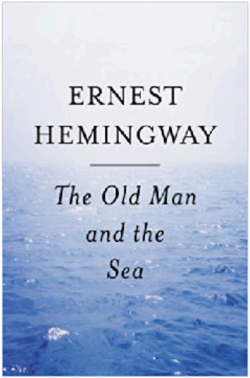
You have your documents tokenized and stored in a data structure, what else might you do with it?

Term frequency (TF) weighting

Some documents are longer than others. Maybe we don't care about raw word counts – only frequency of an occurrence of each word.



VS



Term frequency (TF) weighting

Some documents are longer than others. Maybe we don't care about raw word counts – only frequency of an occurrence of each word.

$\rightarrow$ In this case, we might normalize the document term matrix row to sum to 1 along each row:

$$\text{TF}_i = \frac{X_{ij}}{\sum_{j=1}^{D} X_{ij}} \quad \text{“term frequency”}$$

Inverse-document frequency (IDF) weighting

Similarly, some words occur frequently across all documents but aren't that interesting or useful.

Ex: "Brisket" in a corpus of documents about Texas BBQ. "Rome" in a corpus of travel narratives about Rome. "Congress" in a corpus of political news stories.

The rarer/more specific words might be much more helpful for a given NLP task (classification, summary, matching, etc).

Specificity is inversely proportional to how common a word is across the whole corpus.

Inverse-document frequency (IDF) weighting

IDF weights measure the specificity of a term:

$$\mathrm{IDF}_j = \log(1 + \frac{N}{M_j})$$

where $M_j = \sum_{i=1}^{N} \mathbb{1}(X_{ij} > 0) =$ # of docs where term j appears.

So, weights correspond to the columns/terms, and are inversely proportional to the frequency of appearance across documents

TF-IDF weights

We can combine these to define TF-IDF weights!

$$\begin{aligned}\tilde{X}_{ij} &= \text{TF-IDF}_{ij} \\ &= \text{TF}_{ij} \cdot \text{IDF}_j \\ &= \frac{X_{ij}}{\sum_{j=1}^{D} X_{ij}} \cdot \log(1 + \frac{N}{M_j})\end{aligned}$$

We then use the TF-IDF weights instead of the actual document-term matrix. Words that are frequent in a document but rare in the whole corpus get high TF-IDF weights.

Ok, now what?

- Compare documents (rows of the matrix)
- Compare words (columns of the matrix)
- Cluster documents
- Find low-dimensional summaries, e.g. via PCA
- Classify documents, etc.

Comparing documents

Now, we come back to measuring distance!

A standard measure of similarity is cosine similarity. Intuitively, two documents are similar if their vectors point in the same direction.
(Recall that each row of our document-term matrix is a vector x_i)

$$
\begin{aligned}
\mathrm{sim}(x_1, x_2) &= \frac{x_1 \cdot x_2}{\|x_1\| \cdot \|x_2\|} \\
&= \frac{\sum_{j=1}^{D} x_{1j} x_{2j}}{\left(\sum_{j=1}^{D} x_{1j}^2\right)^{1/2} \cdot \left(\sum_{j=1}^{D} x_{2j}^2\right)^{1/2}} \\
&= \text{cosine of angle between } x_1 \text{ and } x_2
\end{aligned}
$$

From cosine similarity to distance

Cosine similarity can also be used to define cosine distance between two nonnegative vectors.

$$\mathrm{dist}(x_1, x_2) = 1 - \mathrm{sim}(x_1, x_2)$$

As long as we can measure distances, we can cluster documents!