Day 4: Quantitative methods for comparing texts

Kenneth Benoit

Essex Summer School 2014

July 24, 2013

Documents as vectors

- The idea is that (weighted) features form a vector for each document, and that these vectors can be judged using metrics of similarity
- A document's vector for us is simply (for us) the row of the document-feature matrix

Characteristics of similarity measures

Let A and B be any two documents in a set and $d(A, B)$ be the distance between A and B.

1. $d(x, y) \geq 0$ (the distance between any two points must be non-negative)
2. $d(A, B) = 0$ iff $A = B$ (the distance between two documents must be zero if and only if the two objects are identical)
3. $d(A, B) = d(B, A)$ (distance must be symmetric: A to B is the same distance as from B to A)
4. $d(A, C) \leq d(A, B) + d(B, C)$ (the measure must satisfy the triangle inequality)

Euclidean distance

Between document A and B where j indexes their features, where y_{ij} is the value for feature j of document i

- Euclidean distance is based on the Pythagorean theorem
- Formula

$$\sqrt{\sum_{i=1}^{j} (y_{Aj} - y_{Bj})^2} \tag{1}$$

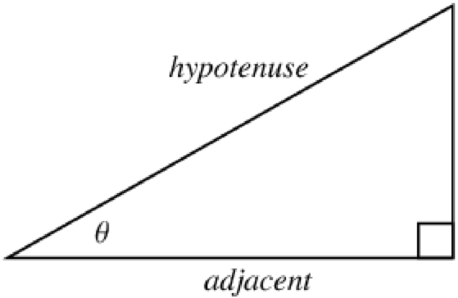
- In vector notation:

$$\|\mathbf{y}_A - \mathbf{y}_B\| \tag{2}$$

- Can be performed for any number of features J (or V as the vocabulary size is sometimes called – the number of columns in of the dfm, same as the number of feature types in the corpus)

Remember Mr. Cosine?

In a right angled triangle, the cosine of an angle θ or $\cos(\theta)$ is the length of the adjacent side divided by the length of the hypotenuse



We can use the vectors to represent the text location in a V-dimensional vector space and compute the angles between them

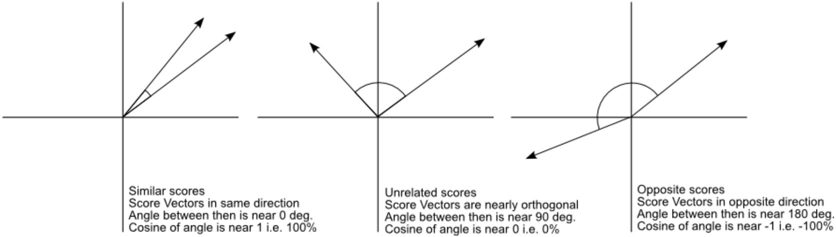
Cosine similarity

- Cosine distance is based on the size of the angle between the vectors
- Formula

$$\frac{\mathbf{y}_A \cdot \mathbf{y}_B}{\|\mathbf{y}_A\| \|\mathbf{y}_B\|} \tag{3}$$

- The $\cdot$ operator is the inner product, or $\sum_j y_{Aj} y_{Bj}$
- The $\|\mathbf{y}_A\|$ is the vector norm of the (vector of) features vector $\mathbf{y}$ for document A, such that $\|\mathbf{y}_A\| = \sqrt{\sum_j y_{Aj}^2}$
- Nice property: independent of document length, because it deals only with the angle of the vectors
- Ranges from -1.0 to 1.0 for term frequencies, or 0 to 1.0 for normalized term frequencies (or tf-idf)

Cosine similarity illustrated



Example text

Hurricane Gilbert swept toward the Dominican Republic Sunday , and the Civil Defense alerted its heavily populated south coast to prepare for high **winds**, heavy **rains** and high seas.

The **storm** was approaching from the southeast with sustained **winds** of 75 mph gusting to 92 mph .

"There is no need for alarm," Civil Defense Director Eugenio Cabral said in a television alert shortly before midnight Saturday .

Cabral said residents of the province of Barahona should closely follow **Gilbert** 's movement .

An estimated 100,000 people live in the province, including 70,000 in the city of Barahona , about 125 miles west of Santo Domingo .

Tropical **Storm Gilbert** formed in the eastern Caribbean and strengthened into a **hurricane** Saturday night

The National **Hurricane** Center in Miami reported its position at 2a.m. Sunday at latitude 16.1 north , longitude 67.5 west, about 140 miles south of Ponce, Puerto Rico, and 200 miles southeast of Santo Domingo.

The National Weather Service in San Juan , Puerto Rico , said **Gilbert** was moving westward at 15 mph with a "broad area of cloudiness and heavy weather" rotating around the center of the **storm**.

The weather service issued a flash flood watch for Puerto Rico and the Virgin Islands until at least 6p.m. Sunday.

Strong **winds** associated with the **Gilbert** brought coastal flooding , strong southeast **winds** and up to 12 feet to Puerto Rico 's south coast.

Example text: selected terms

- Document 1
  Gilbert: 3, hurricane: 2, rains: 1, storm: 2, winds: 2
- Document 2
  Gilbert: 2, hurricane: 1, rains: 0, storm: 1, winds: 2

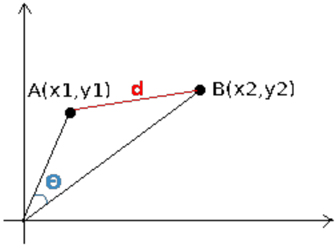
Example text: cosine similarity in R

```
> toyDfm <- matrix(c(3,2,1,2,2, 2,1,0,1,2), nrow=2, byrow=TRUE)
> colnames(toyDfm) <- c("Gilbert", "hurricane", "rain", "storm", "winds")
> rownames(toyDfm) <- c("doc1", "doc2")
> toyDfm
     Gilbert hurricane rain storm winds
doc1       3         2    1     2     2
doc2       2         1    0     1     2
> simil(toyDfm, "cosine")
          doc1
doc2 0.9438798
```

The former measures the similarity of vectors with respect to the origin, while the latter measures the distance between particular points of interest along the vector.

Relationship to Euclidean distance

- Cosine similarity measures the similarity of vectors with respect to the origin
- Euclidean distance measures the distance between particular points of interest along the vector



Relationship to Euclidean distance

- Euclidean distance is $\|\mathbf{y}_A - \mathbf{y}_B\|$
- $\cos(A, B) = \frac{\mathbf{y}_A \cdot \mathbf{y}_B}{\|\mathbf{y}_A\| \|\mathbf{y}_B\|}$

If A and B are normalized to unit length (term proportions instead of frequencies), such that $\|A\|^2 = \|B\|^2 = 1$, then

$$\begin{aligned} \|\mathbf{y}_A - \mathbf{y}_B\|^2 &= (A - B)'(A - B) \\ &= \|A\|^2 + \|B\|^2 - 2\ A'B \\ &= 2(1 - cos(A, B)) \end{aligned}$$

where $(1 - cos(A, B))$ is the complement of the cosine similarity, also known as *cosine distance*

so the Euclidean distance is twice the cosine distance for normalized term vectors

Jacquard coefficient

- Similar to the Cosine similarity
- Formula
$$\frac{\mathbf{y}_A \cdot \mathbf{y}_B}{\|\mathbf{y}_A\| + \|\mathbf{y}_B\| - \mathbf{y}_A \cdot \mathbf{y} y_B} \tag{4}$$
- Ranges from 0 to 1.0
- The $\times$ operator is a ????

Can be made very general for binary features

Example: In the Choi et al paper, they compare vectors of features for (binary) absence or presence – called ("operational taxonomic

Table 1 OTUs Expression of Binary Instances *i* and *j*

j \ *i*	1 (Presence)	0 (Absence)	Sum
1 (Presence)	$a = i \bullet j$	$b = \bar{i} \bullet j$	$a+b$
0 (Absence)	$c = i \bullet \bar{j}$	$d = \bar{i} \bullet \bar{j}$	$c+d$
Sum	$a+c$	$b+d$	$n=a+b+c+d$

units")

- Cosine similarity:

$$s_{\text{cosine}} = \frac{a}{\sqrt{(a+b)(a+c)}} \tag{5}$$

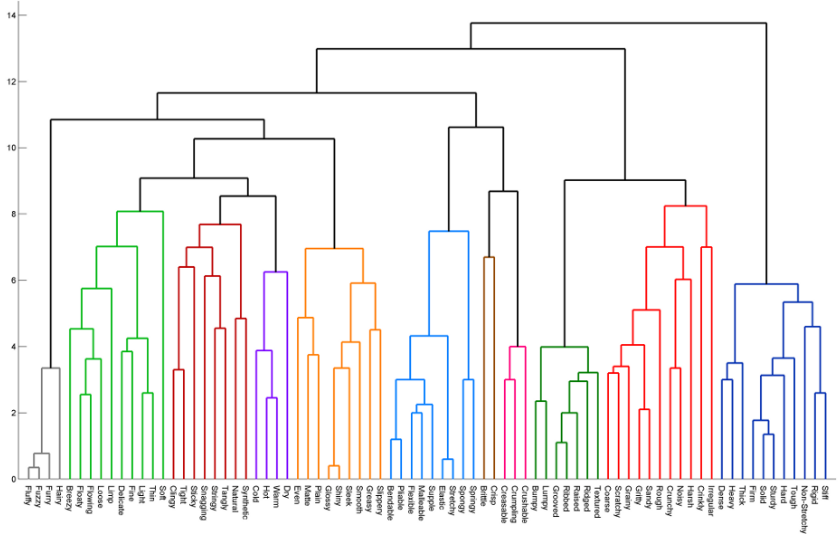
- Jaccard similarity:

$$s_{\text{Jaccard}} = \frac{a}{\sqrt{(a+b+c)}} \tag{6}$$

Typical features

- Normalized term frequency (almost certainly)
- Very common to use tf-idf – if not, similarity is boosted by common words (stop words)
- Not as common to use binary features

Uses for similarity measures: Clustering



Other used for similarity measures

- Used extensively in information retrieval
- Summmary measures of how far apart two texts are – but be careful exactly how you define "features"
- Some but not many applications in social sciences to measure substantive similarity — scaling models are generally preferred

Edit distances

- Edit distance refers to the number of operations required to transform one string into another
- Common edit distance: the Levenshtein distance
- Example: the Levenshtein distance between "kitten" and "sitting" is 3
  - kitten → sitten (substitution of "s" for "k")
  - sitten → sittin (substitution of "i" for "e")
  - sittin → sitting (insertion of "g" at the end).
- Not common, as at a textual level this is hard to implement and possibly meaningless

Detecting "keywords": Constructing the association table

	Class A	**Class B**	Total
Word	*a*	*b*	*a*+*b*
~ Word	*c*	*d*	*c*+*d*
Total	*a*+*c*	*b*+*d*	*N* = *a*+*b*+*c*+*d*

Pearson's chi-squared statistic

$$\chi^2 = \sum \frac{(observed - expected)^2}{expected} = \sum_{i=1}^{k} \frac{(Y_i - np_i)^2}{np_i}$$

$$d.f. = k - 1$$

Chi-squared test of independence

Basic intuition: if the two variables were independent of each other, the relative proportions should be similar to the marginal distributions.

E.g. a word would occur at equal relative frequencies in each subset of a corpus

Since we have two margins, we need to calculate the proportion as:

$$\hat{p}_{word,subset} = \hat{p}_{word} \times \hat{p}_{subset}$$

Generally:

$$\text{Expected Frequency} = \frac{r}{N} \cdot \frac{c}{N} \cdot n = \frac{rc}{N}$$

where r and c refer to row and column marginals

Quantifying Uncertainty

- Critical if we really want to compare texts
- Question: How?
    - Make parametric assumptions about the data-generating process. For instance, we could model feature counts according to a Poisson distribution.
    - Use a sampling procedure and obtain averages from the samples. For instance we could sample 100-word sequences, compute reliability, and look at the spread of the readability measures from the samples
    - Bootstrapping: a generalized resampling method

Bootstrapping

- *Bootstrapping* refers to repeated resampling of data points with replacement
- Used to estimate the error variance (i.e. the standard error) of an estimate when the sampling distribution is unknown (or cannot be safely assumed)
- Robust in the absence of parametric assumptions
- Useful for some quantities for which there is no known sampling distribution, such as computing the standard error of a median

Bootstrapping illustrated

```
> ## illustrate bootstrap sampling
> # using sample to generate a permutation of the sequence 1:10
> sample(10)
 [1]  6  1  2  4  5  7  9  3 10  8
> # bootstrap sample from the same sequence
> sample(10, replace=T)
 [1]  3  3 10  7  5  3  9  8  7  6
> # boostrap sample from the same sequence with probabilities that
> # favor the numbers 1-5
> prob1 <- c(rep(.15, 5), rep(.05, 5))
> prob1
 [1] 0.15 0.15 0.15 0.15 0.15 0.05 0.05 0.05 0.05 0.05
> sample(10, replace=T, prob=prob1)
 [1] 10  4  7  6  5  2  9  5  1  5
```

Bootstrapping the standard error of the median

Using loops:

```
bs <- NULL
for (i in 1:100) {
 bs[i] <- median(sample(spending, replace=TRUE))
}
quantile(bs, c(.025, .5, .975))
median(spending)
```

Bootstrapping the standard error of the median

Using `lapply` and `sapply`:

```
resamples <- lapply(1:100, function(i) sample(spending, replace=TRUE))
bs <- sapply(resamples, median)
quantile(bs, c(.025, .5, .975))
```

Bootstrapping the standard error of the median

Using a user-defined function:

```
b.median <- function(data, n) {
    resamples <- lapply(1:n, function(i) sample(data, replace=T))
    sapply(resamples, median)
    std.err <- sqrt(var(r.median))
    list(std.err=std.err, resamples=resamples, medians=r.median)
}
summary(b.median(spending, 10))
summary(b.median(spending, 100))
summary(b.median(spending, 400))
median(spending)
```

Bootstrapping the standard error of the median

Using R's **boot** library:

```
library(boot)
samplemedian <- function(x, d) return(median(x[d]))
quantile(boot(spending, samplemedian, R=10)$t, c(.025, .5, .975))
quantile(boot(spending, samplemedian, R=100)$t, c(.025, .5, .975))
quantile(boot(spending, samplemedian, R=400)$t, c(.025, .5, .975))
```

Note: There is a good reference on using `boot()` from
`http://www.mayin.org/ajayshah/KB/R/documents/boot.html`

Guidelines for bootstrapping text

- Bootstrap by resampling tokens.
  Advantage: This is easily done from the document-feature matrix.
  Disadvantage: Ignores the natural units into which text is grouped, such as sentences
- Bootstrap by resampling sentences.
  Advantage: Produces more meaningful (potentially readable) texts, more faithful to data-generating process.
  Disadvantage: More complicated, cannot be done from dfm, must segment the text into sentences and construct a new dfm for each resample.
- Other options:
  - paragraphs
  - pages
  - chapters
  - stratified: words within sentences or paragraphs