Text Processing: Overview

-Applied Multivariate Analysis-

Lecturer: Darren Homrighausen, PhD

Who was the first pope?

Suppose we are having a bar-room debate with our friends about the origins of the papacy

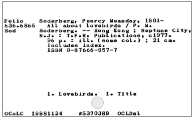
How we would settle this debate has changed radically in the last 20 years.

What we used to do

1. Go to library

2. Card catalog

3. Get metadata



Folio Soderberg, Pearcy Measday, 1901-
636.6865 All about lovebirds / P. M.
Sod Soderberg. -- Hong Kong ; Neptune City,
N.J. : T.F.H. Publications, c1977.
96 p. : ill. (some col.) ; 21 cm.
Includes index.
ISBN 0-87666-957-7

1. Lovebirds. I. Title

OCoLC 19981124 #5370389 OCLDsl

4. Search

5. No book

6. Wait

This was slow and expensive..

What we do now

Information retrieval and representations

How does Google do this?

Information retrieval: given a set of documents (such as webpages, emails, news articles,..), our problem is to retrieve the K most similar documents to a given query (e.g. "who was the first pope?").

The first step is to think of a way of representing these documents.

We want the representation to:

- be both easy to generate from the documents and easy to work with
- highlight important aspects of the documents and suppress unimportant ones

Like always, there is a trade-off between these two ideas

An intuitive first idea


Create account Log in

Article Talk Read Edit View history Search

WIKIPEDIA
The Free Encyclopedia

Main page
Contents
Featured content
Current events
Random article
Donate to Wikipedia
Wikimedia Shop

Interaction
Help
About Wikipedia
Community portal
Recent changes
Contact Wikipedia

Toolbox

Print/export

Languages
Afrikaans
العربية
Aragonés
Asturianu
Беларуская
Беларуская (тарашкевіца)
Bikol Central
Български
Brezhoneg
Català
Česky

List of popes

From Wikipedia, the free encyclopedia

Not to be confused with List of Coptic Orthodox Popes of Alexandria.

For a graphical representation of this list, see List of popes (graphical). For a list of popes by name, see Papal name.

This chronological **list of popes** corresponds to that given in the *Annuario Pontificio* under the heading "I Sommi Pontefici Romani" (The Supreme Pontiffs of Rome), excluding those that are explicitly indicated as antipopes. Published every year by the Roman Curia, the *Annuario Pontificio* attaches no consecutive numbers to the popes, stating that it is impossible to decide which side represented at various times the legitimate succession, in particular regarding Pope Leo VIII, Pope Benedict V and some mid-11th-century popes.[1]

The 2001 edition of the *Annuario Pontificio* introduced "almost 200 corrections to its existing biographies of the popes, from St Peter to John Paul II". The corrections concerned dates, especially in the first two centuries, birthplaces and the family name of one pope.[2]

The term *pope* (Latin: *papa* "father") is used in several Churches to denote their high spiritual leaders (for example Coptic Pope). This title in English usage usually refers to the head of the Roman Catholic Church. The Roman Catholic pope uses various titles by tradition, including *Papa*, *Summus Pontifex*, *Pontifex Maximus*, and *Servus servorum Dei*. Each title has been added by unique historical events and unlike other papal prerogatives, is not incapable of modification.[3]

Hermannus Contractus may have been the first historian to number the popes continuously. His list ends in 1049 with Pope Leo IX as number 154. Several changes were made to the list during the 20th century. Antipope Christopher was considered legitimate for a long time. Pope-elect Stephen was considered legitimate under the name *Stephen II* until the 1961 edition, when his name was erased. Although these changes are no longer controversial, a number of modern lists still include this "first Pope Stephen II". It is probable that this is because they are based on the 1913 edition of the *Catholic Encyclopedia*, which is in the public domain.

A significant number of these popes have been recognized as saints, including 48 out of the first 50 consecutive Popes.

SVMMI PONTIFICES IN HAC BASILICA SEPVLTI

Plaque commemorating the popes buried in St Peter's (their names in Latin and the year of their burial)

Contents [hide]

What if we tried to represent the meaning of documents? For instance, we could take this webpage and record this information

```
beginning.papacy = 37
popes = c('St. Peter','St. Linus',...)
name.origin = 'latin for father'
```

A PROBLEM

This approach is essentially unworkable

While good in terms of the second criteria (highlighting important features), it is terrible in terms of the first (easily generated and used)

This speaks to needing a different representation

Bag-of-words representation

It turns out a very simple minded approach is probably the best developed so far. Take all the words in the document(s) and count how many times they appear and stick this in a long vector (or matrix, if multiple documents).

For example:

```
pope = 154, catholic = 17, vatican = 12, jesus = 2, the = 304,...
```

This is very easy to generate (once we tweak the scripting to ignore certain things).

But is it too much of a reduction?

Bag-of-words representation

Idea: By itself "pope" can mean different things

But, we can learn from the other words in the document

- Words like 'football', 'NFL', 'lineman', and 'arizona' suggest the wrong type of pope
- Words like 'pontiff', 'vatican', 'catholic', and 'italy' suggest the right type of pope
- Words like 'cardinal' are not informative

Counting words

Recall problem: given a query and a set of documents, find the K documents most similar to the query

Countings words:

1. Make a list of all the words present in the documents and the query
2. Index the words $w = 1, \ldots, W$
   (for example, in alphabetical order)
3. Index the documents $d = 1, \ldots, D$
   (just pick some order)
4. For each document d, count how many times each word w is used (can be, and most likely is, zero), and call this count X_{dw}. The vector $X_d = (X_{d1}, \ldots, X_{dW})^\top$ gives the word counts for the d^{th} document
5. Lastly, do the same thing for the query $Y = (Y_1, \ldots, Y_W)^\top$ and Y_w is the count for word w in the query

Simple example

Documents:

$d = 1$: "This statistics class is classy'
$d = 2$: "statistics say this statistics class has no class'

Query:

"classy statistics class"

	this	statistics	class	classy	is	has	no	say
X_1	1	1	1	1	1	0	0	0
X_2	1	2	2	0	0	1	1	1
Y	0	1	1	1	0	0	0	0

This is known as a document-term matrix

Distances and similarity measures

We represented each document X_d and query Y in a convenient vector format. Now, how do we measure similarity?

Measures of distance between W-dimensional vectors X and Y:

- The ℓ_2 or Euclidean distance is

$$||X - Y||_2 = \sqrt{\sum_{w=1}^{W} (X_w - Y_w)^2}$$

- The ℓ_1 or Manhattan distance is

$$||X - Y||_2 = \sum_{w=1}^{W} |X_w - Y_w|$$

There are many others

Basic idea: Find the K vectors X that are 'closest'

Bigger example

Documents: Suppose we have 8 Wikipedia articles, 4 about the TMNT (Leonardo, Raphael, Michelangelo, and Donatello) and about the 4 renaissance artists of the same name

1 2 3 4 5 6 7 8

Query: "Raphael is cool but rude, Michelangelo is a party dude!"

Potential problems

What are the potential problems with performing this query?

Potential problems

What are the potential problems with performing this query?

- Unequal document sizes (example: TMNT Michelangel is 3330 words vs 6524 words for the artist). Also, the query is only 7 words long.
- Stemming
- Punctuation
- Common words ('raphael' occurs 70 times in TMNT article and 144 in the artist's article) provide little discrimination

Distances

If we don't account for any of these problems, we get the following subset of the document-term matrix along with the distance to the query

	but	cool	dude	party	michelangelo	raphael	rude	dist
doc 1	19	0	0	0	4	24	0	309.5
doc 2	8	1	0	0	7	45	1	185.2
doc 3	7	0	4	3	77	23	0	331.0
doc 4	2	0	0	0	4	11	0	220.2
doc 5	17	0	0	0	9	6	0	928.5
doc 6	36	0	0	0	17	101	0	646.5
doc 7	10	0	0	0	159	2	0	527.3
doc 8	2	0	0	0	0	0	0	196.1
query	1	1	1	1	1	1	1	0.0

1. Raphael the Turtle
2. Donatello the Artist
3. Michelangelo the Turtle

Distances

If we don't account for any of these problems, we get the following subset of the document-term matrix along with the distance to the query

	but	cool	dude	party	michelangelo	raphael	rude	dist
doc 1	19	0	0	0	4	24	0	309.5
doc 2	8	1	0	0	7	45	1	185.2
doc 3	7	0	4	3	77	23	0	331.0
doc 4	2	0	0	0	4	11	0	220.2
doc 5	17	0	0	0	9	6	0	928.5
doc 6	36	0	0	0	17	101	0	646.5
doc 7	10	0	0	0	159	2	0	527.3
doc 8	2	0	0	0	0	0	0	196.1
query	1	1	1	1	1	1	1	0.0

1. Raphael the Turtle
2. Donatello the Artist
3. Michelangelo the Turtle

Varying document lengths and normalization

Different documents have different lengths. Total word counts:

doc 1	doc 2	doc 3	doc 4	doc 5	doc 6	doc 7	doc 8	query
3114	1976	3330	2143	8962	6524	4618	1766	7

Note that the documents have quite different lengths

We should normalize them in some way

- Document length: Divide X by its sum

$$X \leftarrow X / \sum_{w=1}^{W} X_w$$

- ℓ_2 length: Divide X by its Euclidean length

$$X \leftarrow X / ||X||_2$$

Back to our example

```
    dist/doclen  dist/l2len
doc 1 0.3852650 1.373041
doc 2 0.3777634 1.321871
doc 3 0.3781194 1.319048
doc 4 0.3887862 1.393433
doc 5 0.3906030 1.404972
doc 6 0.3820197 1.349070
doc 7 0.3812202 1.324758
doc 8 0.3935327 1.411486
query 0.0000000 0.000000
```

Great!

So far, we've dealt with the varying document lengths. What about some words being more helpful than others?

Common words, especially, are not going to help find relevant documents

How do we deal with common words?

Intuition: Words that do not appear very often should help us discriminate better, as they are by implication very specific to that document

To deal with common words we could just keep track of a list of words like ‘the’, ‘this’, ‘that’, etc. and exclude them from our representation.

This is both too crude and time consuming

Common words and IDF weighting

Inverse document frequency (IDF) weighting is smarter and more efficient

- For each word, w, let n_w be the number of documents that contain this word
- The, for each vector X_d and Y, multiply the w^{th} component by
$$IDF(w) = \log(D/n_w)$$

Note that if a word appears in every document, then it gets a weight of zero.

If $n_w < D$, then $\log(D/n_w) > 0$. In particular, if $D >> n_w$, then D/n_w is also large (example: $D = 100$, $n_w = 1$, $IDF(w) \approx 4.6$)

Putting it all together

Think of the document-term matrix

	word 1	word 2	...	word W
doc 1				
doc 2				
$\vdots$				
doc D				

- Normalization scales each *row* by something (divides a row vector X by its sum or ℓ_2 norm)
- IDF weighting scales each *column* by something (multiplies the w^{th} column by $IDF(w)$)
- We can use both, just normalize first and then perform IDF

Back to our example

```
                    dist/doclen/IDF
doc 1 (tmnt leo)              0.623
doc 2 (tmnt rap)              0.622
doc 3 (tmnt mic)              0.620
doc 4 (tmnt don)              0.623
doc 5 (real leo)              0.622
doc 6 (real rap)              0.622
doc 7 (real mic)              0.622
doc 8 (real don)              0.624
query (tmnt leo)              0.000
```

What happened?

```
 [1] "---"         "----"        "---x"        "--dead"      "--x"         "-foot"
 [7] "-part"       "-year-old"   "abandoned"   "abbeville"   "abbey"       "abilities"
[13] "ability"     "able"        "abode"       "about"       "above"       "abrams"
[19] "abroad"      "abruptly"    "absence"     "absent"      "absolute"    "absorbed"
[25] "absorbing"   "abstemious"  "absurd"      "abundance"   "abundantly"  "abuse"
[31] "academic"    "academies"   "academy"     "accademia"   "accent"      "accept"
[37] "acceptance"  "accepted"    "accepting"   "accident"    "acclaimed"   "acclimate"
[43] "accompany"   "accomplish"  "accordance"  "according"   "account"     "accounts"
```

Stemming

Having words ‘connect’, ‘connects’, ‘connected’, ‘connecting’, ‘connection’, etc. in our representation is extraneous. Stemming reduces all of these to a single stem word ‘connect’

Can a simple list of rules provide perfect stemming? No: ‘relate’ vs. ‘relativity’ or ‘sand’ and ‘sander’ or ‘wand’ and ‘wander’ or ‘man’ and ‘many’ or...

Stemming also depends on the language. It is easier in English than in:

- German (fusional or agglomerative language) e.g.
  Hubschrauberlandeplatz = helicopter landing pad
- Turkisk (agglutinative language) e.g.
  Turklestiremedigimizlerdensinizdir = maybe you are one of those whom we were not able to Turkify

Stemming

Before

```
 [1] "---"         "----"        "---x"        "--dead"      "--x"         "-foot"
 [7] "-part"       "-year-old"   "abandoned"   "abbeville"   "abbey"       "abilities"
[13] "ability"     "able"        "abode"       "about"       "above"       "abrams"
[19] "abroad"      "abruptly"    "absence"     "absent"      "absolute"    "absorbed"
[25] "absorbing"   "abstemious"  "absurd"      "abundance"   "abundantly"  "abuse"
[31] "academic"    "academies"   "academy"     "accademia"   "accent"      "accept"
[37] "acceptance"  "accepted"    "accepting"   "accident"    "acclaimed"   "acclimate"
[43] "accompany"   "accomplish"  "accordance"  "according"   "account"     "accounts"
```

After

```
 [1] "---"        "----"       "---x"       "--dead"     "--x"        "-foot"
 [7] "-part"      "-year-old"  "abandon"    "abbevill"   "abbey"      "abil"
[13] "abl"        "abod"       "abov"       "abram"      "abroad"     "abrupt"
[19] "absenc"     "absent"     "absolut"    "absorb"     "abstemi"    "absurd"
[25] "abund"      "abus"       "academ"     "academi"    "accademia"  "accent"
[31] "accept"     "accid"      "acclaim"    "acclim"     "accommod"   "accompani"
[37] "accomplish" "accord"     "account"    "accumul"    "accur"      "accus"
[43] "achiev"     "ackerman"   "acknowledg" "acolyt"     "acquir"     "act"
```

Back to our example, after stemming

1

2

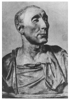
3

4

5

6

7



8

Query: "Raphael is cool but rude, Michelangelo is a party dude!"

	dist/doclen/IDF
doc 1 (tmnt leo)	0.965
doc 2 (tmnt rap)	0.870
doc 3 (tmnt mic)	0.867
doc 4 (tmnt don)	0.971
doc 5 (real leo)	0.927
doc 6 (real rap)	0.971
doc 7 (real mic)	0.954
doc 8 (real don)	0.930
query	0.000