Query

- Which plays of Shakespeare contain the words *Brutus AND Caesar* but NOT *Calpurnia*?
- Could grep all of Shakespeare's plays for *Brutus* and *Caesar* then strip out lines containing *Calpurnia*?
  - Slow (for large corpora)
  - *NOT* is hard to do
  - Other operations (e.g., find the *Romans NEAR countrymen*) not feasible

Term-document incidence

<table>
<thead>
<tr>
<th>Term</th>
<th>Antony and Cleopatra</th>
<th>Julius Caesar</th>
<th>The Tempest</th>
<th>Hamlet</th>
<th>Othello</th>
<th>Macbeth</th>
</tr>
</thead>
<tbody>
<tr>
<td>Antony</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Brutus</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Caesar</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Calpurnia</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Cleopatra</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>mercy</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>verger</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Incidence vectors

- So we have a 0/1 vector for each term.
- To answer query: take the vectors for *Brutus, Caesar* and *Calpurnia* (complemented) \( \Rightarrow \) bitwise *AND*.
- \[ 110100 \text{ AND } 110111 \text{ AND } 101111 = 100100. \]

Answers to query

- Antony and Cleopatra, Act III, Scene ii
  - Agrippa [aside to DOMITIUS ENOBARBUS]: Why, Enobarbus, When Antony found Julius Caesar dead, We shed almost to roaring; and he wept When at Philippi he found Brutus slain.

- Hamlet, Act III, Scene ii
  - Lord Polonius: I did enact Julius Caesar I was killed in the Capit. Brutus killed me.

Bigger corpora

- Consider \( n = 1 \text{M} \) documents, each with about 1K terms.
- Avg 6 bytes/term incl spaces/punctuation
  - 6GB of data.
- Say there are \( m = 500 \text{K distinct} \) terms among these.
Can’t build the matrix

- 500K x 1M matrix has half-a-trillion 0’s and 1’s.
- But it has no more than one billion 1’s.
- matrix is extremely sparse.
- What’s a better representation?

Inverted index

- Documents are parsed to extract words and these are saved with the document ID.

Documents are parsed to extract words and these are saved with the document ID.

I did enact Julius Caesar I was killed ’t the Capitol; Brutus killed me.

So let it be with Caesar. The noble Brutus hath told you Caesar was ambitious

After all documents have been parsed the inverted file is sorted by terms

Multiple term entries in a single document are merged and frequency information added

Issues with index we just built

- How do we process a query?
- What terms in a doc do we index?
  - All words or only “important” ones?
- Stopword list: terms that are so common that they’re ignored for indexing.
  - e.g., the, a, an, of, to ...
  - language-specific.

Issues in what to index

- Cooper’s vs. Cooper vs. Cooper.
- Full-text vs. full text vs. [full, text] vs. fulltext.
- Accents: résumé vs. resume.

Cooper’s concordance of Wordsworth was published in 1911. The applications of full-text retrieval are legion: they include résumé scanning, litigation support and searching published journals on-line.
Punctuation
- Ne’er: use language-specific, handcrafted “locale” to normalize.
- State-of-the-art: break up hyphenated sequence.
- U.S.A. vs. USA - use locale.
- a.out

Numbers
- 3/12/91
- Mar. 12, 1991
- 55 B.C.
- B-52
- 100.2.86.144
  - Generally, don’t index as text
  - Creation dates for docs

Case folding
- Reduce all letters to lower case
  - exception: upper case in mid-sentence
    - e.g., General Motors
    - Fed vs. fed
    - SAIL vs. sail

Thesauri and soundex
- Handle synonyms and homonyms
  - Hand-constructed equivalence classes
    - e.g., car = automobile
    - your ≠ you’re
  - Index such equivalences, or expand query?
    - More later ...

Spell correction
- Look for all words within (say) edit distance 3 (Insert/Delete/Replace) at query time
  - e.g., Alanis Morisette
- Spell correction is expensive and slows the query (up to a factor of 100)
  - Invoke only when index returns zero matches?
  - What if docs contain mis-spellings?

Lemmatization
- Reduce inflectional/variant forms to base form
  - E.g.,
    - am, are, is → be
    - car, cars, car’s, cars’ → car
  - the boy’s cars are different colors → the boy car be different color
Stemming

- Reduce terms to their “roots” before indexing
  - language dependent
  - e.g., *automate(s), automatic, automation* all reduced to *automat.*

  for example compressed and compression are both accepted as equivalent to compress.

  for example compres and compres are both accept as equval to compres.

Porter’s algorithm

- Commonest algorithm for stemming English
- Conventions + 5 phases of reductions
  - phases applied sequentially
  - each phase consists of a set of commands
  - sample convention: *Of the rules in a compound command, select the one that applies to the longest suffix.*
- Porter’s stemmer available: [http//www.sims.berkeley.edu/~hearst/irbook/porter.html](http://www.sims.berkeley.edu/~hearst/irbook/porter.html)

Typical rules in Porter

- *sses → ss*
- *ies → i*
- *ational → ate*
- *tional → tion*

Beyond term search

- What about phrases?
- Proximity: Find *Gates NEAR Microsoft.*
  - Need index to capture position information in docs.
- Zones in documents: Find documents with *(author = Ullman) AND (text contains *automata).*

Evidence accumulation

- 1 vs. 0 occurrence of a search term
  - 2 vs. 1 occurrence
  - 3 vs. 2 occurrences, etc.
- Need term frequency information in docs

Ranking search results

- Boolean queries give inclusion or exclusion of docs.
- Need to measure proximity from query to each doc.
- Whether docs presented to user are singletons, or a group of docs covering various aspects of the query.
Test Corpora

**TABLE 4.1 Common Test Corpora**

<table>
<thead>
<tr>
<th>Collection</th>
<th>#Docs</th>
<th>#Qrys</th>
<th>Size (MB)</th>
<th>Term/Doc</th>
<th>Q-D NBAns</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADE</td>
<td>83</td>
<td>35</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CET</td>
<td>2303</td>
<td>14</td>
<td>2</td>
<td>400</td>
<td>&gt;10,000</td>
</tr>
<tr>
<td>CACM</td>
<td>3034</td>
<td>66</td>
<td>3</td>
<td>14.5</td>
<td></td>
</tr>
<tr>
<td>CIIR</td>
<td>1460</td>
<td>112</td>
<td>2</td>
<td>46.5</td>
<td></td>
</tr>
<tr>
<td>Goldhill</td>
<td>1493</td>
<td>200</td>
<td>2</td>
<td>12.1</td>
<td></td>
</tr>
<tr>
<td>LISA</td>
<td>3072</td>
<td>52</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Medline</td>
<td>1331</td>
<td>30</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SPLE</td>
<td>11,459</td>
<td>93</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CSHYBB</td>
<td>34,356</td>
<td>116</td>
<td>498</td>
<td>250</td>
<td>16,140</td>
</tr>
<tr>
<td>Reuters</td>
<td>21,578</td>
<td>672</td>
<td>28</td>
<td>134</td>
<td></td>
</tr>
<tr>
<td>TREC</td>
<td>760,009</td>
<td>2000</td>
<td>2000</td>
<td>0.3543</td>
<td>&gt;10,000</td>
</tr>
</tbody>
</table>

Standard relevance benchmarks

- **TREC** - National Institute of Standards and Testing (NIST) has run large IR testbed for many years
- Reuters and other benchmark sets used
- “Retrieval tasks” specified
  - sometimes as queries
- Human experts mark, for each query and for each doc, “Relevant” or “Not relevant”
  - or at least for subset that some system returned

Sample TREC query

**Sample TREC query**

*Topic:* Tobacco company advertising and the young

**Description:** A document will provide information on what is a widely held opinion that the tobacco industry aims to advertise for the young. Nominations: A relevant document must report on tobacco company advertising and its relation to young people. A relevant document can address either side of the question: (1) Do tobacco companies consciously target the young, or (2) As the tobacco industry argues, is this an erroneous public perception. The “youth” may be identified as youth, children, adolescents, teenagers, high school students, and college students.

Precision and recall

- **Precision**: fraction of retrieved docs that are relevant = Pr(relevant|retrieved)
- **Recall**: fraction of relevant docs that are retrieved = Pr(retrieved|relevant)

<table>
<thead>
<tr>
<th>Relevant</th>
<th>Not Relevant</th>
</tr>
</thead>
<tbody>
<tr>
<td>Retrieved</td>
<td>tp, fp</td>
</tr>
<tr>
<td>Not Retrieved</td>
<td>fn, tn</td>
</tr>
</tbody>
</table>

- Precision $P = \frac{tp}{tp + fp}$
- Recall $R = \frac{tp}{tp + fn}$

**Precision/Recall**

- Can get high recall (but low precision) by retrieving all docs on all queries!
- Recall is a non-decreasing function of the number of docs retrieved
  - Precision usually decreases (in a good system)
- Difficulties in using precision/recall
  - Binary relevance
  - Should average over large corpus/query ensembles
  - Need human relevance judgements
  - Heavily skewed by corpus/authorship
A combined measure: F

- Combined measure that assesses this tradeoff is F measure (weighted harmonic mean):
  \[ F = \frac{1}{\frac{\alpha}{P} + (1-\alpha)\frac{1}{R}} = \frac{\beta^2 + 1}{\beta^2 P + R} \]

- People usually use balanced F_1 measure
  - i.e., with \( \beta = 1 \) or \( \alpha = \frac{1}{2} \)
  - Harmonic mean is conservative average
- See CJ van Rijsbergen, *Information Retrieval*

Evaluation

- There are various other measures
  - Precision at fixed recall
    - This is perhaps the most appropriate thing for web search: all people want to know is how many good matches there are in the first one or two pages of results
  - 11-point interpolated average precision
    - The standard measure in the TREC competitions: Take the precision at 11 levels of recall varying from 0 to 1 by tenths of the documents, using interpolation (the value for 0 is always interpolated!), and average them

Ranking models in IR

- Key idea:
  - We wish to return in order the documents most likely to be useful to the searcher
- To do this, we want to know which documents best satisfy a query
  - An obvious idea is that if a document talks about a topic more then it is a better match
  - A query should then just specify terms that are relevant to the information need, without requiring that all of them must be present
- Document relevant if it has a lot of the terms

Binary term presence matrices

- Record whether a document contains a word: document is binary vector in \( \{0,1\}^n \)
- Idea: Query satisfaction = overlap measure:
  \[ |X \cap Y| \]
Overlap matching

- What are the problems with the overlap measure?
- It doesn’t consider:
  - Term frequency in document
  - Term scarcity in collection
    - (How many documents mention term?)
  - Length of documents

Many Overlap Measures

<table>
<thead>
<tr>
<th>Measure</th>
<th>Formula</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simple matching (coordination level match)</td>
<td>$</td>
</tr>
<tr>
<td>Dice’s Coefficient</td>
<td>$2</td>
</tr>
<tr>
<td>Jaccard’s Coefficient</td>
<td>$</td>
</tr>
<tr>
<td>Cosine Coefficient</td>
<td>$</td>
</tr>
<tr>
<td>Overlap Coefficient</td>
<td>$\min(</td>
</tr>
</tbody>
</table>

Documents as vectors

- Each doc $j$ can be viewed as a vector of tf values, one component for each term
- So we have a vector space
  - terms are axes
  - docs live in this space
  - even with stemming, may have 20,000+ dimensions
- (The corpus of documents gives us a matrix, which we could also view as a vector space in which words live – transposable data)

Documents in 3D Space

Assumption: Documents that are “close together” in space are similar in meaning.

The vector space model

Query as vector:

- Regard query as short document
- Return the docs, ranked by distance to the query
- Easy to compute, since both query & docs are vectors.

- Developed in the SMART system (Salton, c. 1970) and standardly used by TREC participants and web IR systems

Vector Representation

- Documents & Queries represented as vectors.
- Position 1 corresponds to term 1, ...position t to term t
- The weight of the term is stored in each position
  $D_i = w_{d_i1}, w_{d_i2}, ..., w_{d_it}$
  $Q = w_{q1}, w_{q2}, ..., w_{qt}$
  $w = 0$ if a term is absent

- Vector distance measure used to rank retrieved documents
Documents in 3D Space

Documents that are close to query (measured using vector-space metric) => returned first.

Documents in 3D Space

Document Space has High Dimensionality

- What happens beyond 2 or 3 dimensions?
  - Similarity still has to do with the number of shared tokens.
  - More terms -> harder to understand which subsets of words are shared among similar documents.

- We will look in detail at ranking methods
  - One approach to handling high dimensionality: Clustering

Word Frequency

- Which word is more indicative of document similarity?
  - ‘book,’ or ‘Rumplestiltskin’?
  - Need to consider “document frequency”---how frequently the word appears in doc collection.

- Which doc is a better match for the query “Kangaroo”?
  - One with a single mention of Kangaroos… or a doc that mentions it 10 times?
  - Need to consider “term frequency”---how many times the word appears in the current document.

TF x IDF

\[ w_{ik} = tf_{ik} \times \log\left(\frac{N}{n_k}\right) \]

- \( T_i \) = term \( k \) in document \( D_j \)
- \( tf_{ik} \) = frequency of term \( T_i \) in document \( D_j \)
- \( idf_i = \text{inverse document frequency of term } T_i \text{ in } C \)
- \( N \) = total number of documents in the collection \( C \)
- \( n_k \) = the number of documents in \( C \) that contain \( T_i \)

\[ idf_i = \log\left(\frac{N}{n_k}\right) \]

Inverse Document Frequency

- IDF provides high values for rare words and low values for common words

\[
\begin{align*}
\log\left(\frac{10000}{10000}\right) &= 0 \\
\log\left(\frac{10000}{5000}\right) &= 0.301 \\
\log\left(\frac{10000}{20}\right) &= 2.698 \\
\log\left(\frac{10000}{1}\right) &= 4
\end{align*}
\]

TF-IDF normalization

- Normalize the term weights
  - so longer docs not given more weight (fairness)
  - force all values to fall within a certain range: \([0, 1]\)

\[
w_{ik} = \frac{tf_{ik} \log\left(\frac{N}{n_k}\right)}{\sqrt{\sum_{i=1}^{n}(tf_{ik})^2 \log\left(\frac{N}{n_k}\right)^2}}
\]
Vector space similarity
(use the weights to compare the documents)

Now, the similarity of two documents is:

\[ \text{sim}(D_i, D_j) = \sum_{k=1}^{n} w_{ik} \cdot w_{jk} \]

This is also called the cosine, or normalized inner product.
(Normalization was done when weighting the terms.)

What’s Cosine anyway?

One of the basic trigonometric functions encountered in trigonometry.
Let \( \theta \) be an angle measured counterclockwise from the x-axis along the arc of the unit circle. Then \( \cos(\theta) \) is the horizontal coordinate of the arc endpoint. As a result of this definition, the cosine function is periodic with period \( 2\pi \).

From http://mathworld.wolfram.com/Cosine.html

Cosine Detail (degrees)

Computing Cosine Similarity Scores

<table>
<thead>
<tr>
<th>Document</th>
<th>Query</th>
</tr>
</thead>
<tbody>
<tr>
<td>( D_1 )</td>
<td>( Q )</td>
</tr>
<tr>
<td>( D_2 )</td>
<td>( Q )</td>
</tr>
<tr>
<td>( D_3 )</td>
<td>( Q )</td>
</tr>
</tbody>
</table>

\[ \cos \alpha_1 = 0.74 \]
\[ \cos \alpha_2 = 0.98 \]

To Think About

- How does this ranking algorithm behave?
- Make a set of hypothetical documents consisting of terms and their weights
- Create some hypothetical queries
- How are the documents ranked, depending on the weights of their terms and the queries’ terms?

Computing a similarity score

Say we have query vector \( Q = (0.4, 0.8) \)
Also, document \( D_1 = (0.2, 0.7) \)
What does their similarity comparison yield?

\[ \text{sim}(Q, D_1) = \frac{(0.4 \times 0.2) + (0.8 \times 0.7)}{\sqrt{(0.4)^2 + (0.8)^2} \times \sqrt{(0.2)^2 + (0.7)^2}}} \]
\[ = \frac{0.64}{\sqrt{0.42}} = 0.98 \]
## Summary: Why use vector spaces?

- User’s query treated as a (very) short document.
- Query → a vector in the same space as the docs.
- Easily measure each doc’s proximity to query.
- Natural measure of scores/ranking
  - No longer Boolean.