Classic Info Retrieval continued

foundational techniques for text documents

Last time:

- the information retrieval process

information need => query => results => relevance

- modeling documents and queries
- the Boolean model

Today:

- ranking and the vector model
- extending models, techniques for modern search

Simple Model with Ranking

- Document: *bag* of terms - count occurrences
- Query: *set* of terms
- Satisfying: OR model
- Ranking: numerical score measuring degree to which document satisfies query

  some choices:
  - one point for each query term in document
  - ➢one point for ***each occurrence*** of a query term in document
- Documents returned in sorted list by decreasing score

Simple Model: example

Doc 1: "Computers have brought the world to our fingertips. We will try to understand at a basic level the science -- old and new -- underlying this new Computational Universe. Our quest takes us on a broad sweep of scientific knowledge and related technologies… Ultimately, this study makes us look anew at ourselves -- our genome; language; music; "knowledge"; and, above all, the mystery of our intelligence. (cos 116 description)

Frequencies:

science 1; knowledge 2; principles 0; engineering 0

Doc 2: "An introduction to computer science in the context of scientific, engineering, and commercial applications. The goal of the course is to teach basic principles and practical issues, while at the same time preparing students to use computers effectively for applications in computer science …" (cos 126 description)

Frequencies:

science 2; knowledge 0; principles 1; engineering 1

Generalize Simple Model:
The Vector Model

- Have a *lexicon* (aka *dictionary*) of all terms appearing in the collection of documents
  - *m* terms in all, number 1, …, *m*
- Document: an *m*-dimensional vector
  - i^{th} entry of the vector is a real-valued *weight* (importance of) term *i* in the document
- Query: an *m*-dimensional vector
  - The i^{th} entry of the vector is a real-valued *weight* (importance of) term *i* in the query

Vector Model: Satisfying & Ranking

- Satisfying:
  - Each document is scored as to the degree it satisfies query (higher better)
  - there is no inherent notion of satisfying
  - typically doc satisfies query if score is > threshold
- Ranking:
  - Documents are returned in sorted list decreasing by score:
    - Include only highest *n* documents, some *n?*

Where get dictionary of *t* terms?

- Pre-determined dictionary.
  - How sure get all terms?
- Build lexicon when collect documents
  - What if collection dynamic: add terms?

How compute score

Calculate a vector function of the document vector and the query vector

Choices:

1. distance between the vectors:

$\text{Dist}(\boldsymbol{d},\boldsymbol{q}) = \sqrt{(\Sigma^t_{i=1}(\boldsymbol{d}_i - \boldsymbol{q}_i)^2)}$

- Is *dissimilarity* measure
- Not normalized: Dist ranges [0, inf.)
- Fix: use $e^{-\textbf{Dist}}$ with range (0,1]
- Is it the right sense of difference?

How compute score, continued

2. angle between the vectors:

Dot product: $\boldsymbol{d \bullet q} = \Sigma^t_{i=1}(\boldsymbol{d}_i * \boldsymbol{q}_i)$

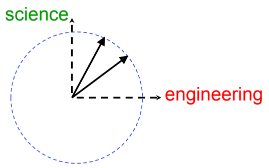
- Is *similarity* measure
- Not normalized: dot product ranges (-inf., inf.)
- Fix: use normalized dot product, range [-1,1]

  $\boldsymbol{(d \bullet q) / (|d|*|q|)}$ where $|\boldsymbol{v}| = \sqrt{\Sigma^t_{i=1}(\boldsymbol{v}_i^2)}$

  the length of $\boldsymbol{v}$
- In practice vector components are non-negative so range is [0,1]
- This most commonly used function for score

Normalizing vectors

- If use unit vectors, $\boldsymbol{d} / |\boldsymbol{d}|$ and $\boldsymbol{v} / |\boldsymbol{v}|$ some of issues go away



The Simple Model as a Vector Model

- Document: an *m*-dimensional vector
  - i^{th} entry of the vector is the number of times term *i* appears in the document
- Query: an *m*-dimensional vector
  - The i^{th} entry of the vector is 1 if term *i* in the query, 0 otherwise
- Vector function: dot product

How compute weights d_i and q_i?

First:

observations about this model?

Vector model: Observations

- Have matrix of terms by documents

  ⇒Can use linear algebra
- Queries and documents are the same

  ⇒Can compare documents same way
  - Clustering documents
- Document with only some of query terms can score higher than document with all query terms

How compute weights

- Vector model *could* have weights assigned by human intervention
  - may add meta-information
  - User setting query weights might make sense
    - User decides importance of terms in own search
  - Humans setting document weights?
    - Who? Billions+ of documents
- Return to model of documents as bag of words – calculate weights
  - Function mapping bag of words to vector

Calculations on board:

- General notation:
  - w_{jd} is the weight of term *j* in document *d*
  - $freq_{jd}$ is the # of times term *j* appears in doc *d*
  - $freq_{jC}$ is the # of times term *j* appears in the collection C of documents (collection frequency)
  - *$length_d$ is the total number of occurrences of terms in document d (word length)*
  - n_j = # docs containing term j
  - *N* = number of docs in collection

Summary weight calculation

- General notation:
  - w_{jd} is the weight of term *j* in document *d*
  - $freq_{jd}$ is the # of times term *j* appears in doc *d*
  - n_j = # docs containing term j
  - *N* = number of docs in collection
- Classic *tf-idf* definition of weight, normalized:

$$u_{jd} = freq_{jd} \; * \; \log(N/ n_j)$$

$$w_{jd} = \frac{u_{jd}}{(\Sigma^t_{i=1}(u_{id}^2))^{1/2}}$$

Weight of query components?

- Set of terms, ***some choices***:
  1. w_{jq} = 0 or 1
  2. $w_{jq} = freq_{jq} \; * \; \log(N/ n_j)$
     $= 0 \text{ or } \log(N/ n_j)$
- Bag of terms
  - Analyze like document

    Some queries are prose expressions of ***information need***

Do we want idf term in both document weights and query weights?

Vector Model example

Doc 1: "Computers have brought the world to our fingertips. We will try to understand at a basic level the science -- old and new -- underlying this new Computational Universe. Our quest takes us on a broad sweep of scientific knowledge and related technologies… Ultimately, this study makes us look anew at ourselves -- our genome; language; music; "knowledge"; and, above all, the mystery of our intelligence. (cos 116 description)

Frequencies:

science 1; knowledge 2; principles 0; engineering 0

Doc 2: "An introduction to computer science in the context of scientific, engineering, and commercial applications. The goal of the course is to teach basic principles and practical issues, while at the same time preparing students to use computers effectively for applications in computer science …" (cos 126 description)

Frequencies:

science 2; knowledge 0; principles 1; engineering 1

Vector model example cont.

- Consider the 5 100-level and 200-level COS courses as the collection (109, 217, 226)
- Only other appearance of our 4 words is "science" once in 109 description.
- idf:

  science ln(5/3) = .51

  engineering, principles, knowledge:

  ln(5/1) = 1.6

Term by Doc. Table: *$freq_{jd}$ * $log(N/n_j)$*

	Doc 1	Doc 2
science	.51	1.02
engineering		1.6
principles		1.6
knowledge	3.2	

Unnormalized dot product for query: *science*, *engineering*, *knowledge*, *principles* using 0/1 query vector

- Doc 1: 3.71
- Doc 2: 4.22

- If documents have about same vector length, this right ratio for normalized (cosine) score

Additional ways to calculate document weights

- Dampen frequency effect:
  w_{jd} = 1+ log ($freq_{jd}$) if $freq_{jd}$ > 0; 0 otherwise
- *Use smoothing term to dampen effect:*
  W_{jd} = a + (1-a) $freq_{jd}$ / max_p ($freq_{pd}$)
  - *a is typically .4 or .5*
  - *Can multiply second term by idf*
- *Effects for long documents (Section 6.4.4)*

Classic IR models - Taxonomy

Well-specified models:

✓Boolean
✓Vector

- Probabilistic
  - based on probabilistic model of words in documents

Probabilistic Model Brief Overview

- Binary Independence Model
  - Original model
  - Chapter 11
- Language Model
  - More commonly used
  - Chapter 12

Key probability ideas

- P(A) probability event A occurs
- P(A,B) probability both A and B occur
- P(A|B) probability A occurs given B occurs

- P(A,B) = P(A|B)*P(B) = P(B|A)*P(A)

Language Model

- Is probability measure over strings over some “vocabulary”
- Want to estimate the model of a document
- Want to rank using this estimate
- Assume terms independent

  $P(t1, t2) = P(t1) * P(t2)$

25

Ranking function: main idea

- Rank using P(doc|query) = P(d|q)
- $P(d|q) = P(q|d)* P(d)/P(q)$
- Estimate P(d|q)
  - Don’t care about numerical accuracy
  - Care that preserves ranking order!
- Assume uniform distribution P(d) => ignore
- P(q) constant for give query => ignore
- Left with rank ≈ P(q|d)
- Estimate $P(q|d) = \Pi_{t \text{ in } q} P(t|d)$ for terms t

  $= \Pi_{t \text{ in } q} (freq_{td} / length_d)$

26

Extending classic information retrieval for today’s possibilities

27

Ranking

- What intuitive criteria?

28

Enhanced document model

- First model: set of terms
  - term in/not in document
- Next: bag of terms
  - know frequency of terms in document
- Now: sequence of terms + additional properties of terms
  - sequence gives you where term in doc
    - derive relative position of multiple query terms
  - Special use? (e.g. in title, font, …)
    - most require “mark-up”: tags, meta-data, etc.

29

HTML mark-up example

```
<h2> <font color="#A52A2A"> Communication </font></h2>
This course will be essentially ``paperless''. All assignments
   will be posted <i>only</i> on the course Web site.
   ``Handouts'' and copies of any transparencies used in
   class will be posted on the course Web site as well.
   Important announcements on all aspects of the course will
   be made on the <a href="announce.html">
   Announcements</a>  page. <b>Students are responsible
   for monitoring the postings under  ``Announcements''. </b>
   Schedule changes will be made on the on-line <a
   href="schedule.html"> schedule page</a>. and announced
   under ``Announcements''. The only paper we will
   exchange is your solutions to the problem sets, which we
   will grade and hand back, the exam questions and your
   responses, and your project reports.
```

30

yields

Communication

This course will be essentially ``paperless''. All assignments will be posted *only* on the course Web site (see Schedule and Readings). ``Handouts'' and copies of any transparencies used in class will be posted on the course Web site as well. Important announcements on all aspects of the course will be made on the Announcements page. **Students are responsible for monitoring the postings under ``Announcements''.** Schedule changes will be made on the on-line schedule page. and announced under ``Announcements''. The only paper we will exchange is your solutions to the problem sets, which we will grade and hand back, the exam questions and your responses, and your project reports.

Enhanced document model: restate

"sequence of terms + properties of terms"

⇓ WHY?

"set of (term, properties) pairs"

Properties:

- for each distinct term
  - Frequency of term in doc
    - Vector model of classic IR
- for individual occurrence of each term
  - Where in doc.
  - properties of use

Model

- Document: set of (term,properties) pairs
- Query: sequence of terms
  - Can make more complicated
- Satisfying: AND model
  - relax if no document contains all?
- Ranking: wide open function
  - info beyond documents and query ?

Data Structure for Collection

- for each document, keep list of:
  - terms appearing
    - aggregate properties of term
      e.g. frequency
    - positions at which each term occurs
      - attributes for each occurrence of term
- keep summary information for documents

Data Structure for Collection: Invert

- for each term, keep list of:
  - documents in which it appears
    - positions at which it occurs in each doc.
      - attributes for each occurrence
- keep summary information for documents
- keep summary information for terms

Inverted Index for Collection

- for each term, keep POSTINGS LIST of:
  - each document in which it appears
    - each position at which it occurs in doc.
      - attributes for each occurrence

POSTING

- Core structure used by query evaluation and document ranking algorithms

Index structure

term$_1$:(doc ID (position, attributes)
(position, attributes),
…
(position, attributes))
(doc ID (position, attributes)
(position, attributes),
…
(position, attributes))
…
term$_2$:(doc ID (position, attributes)
(position, attributes),
…
(position, attributes))
…

Models have seen

Model	Document	Query	Satisfy
Boolean	set of terms	Boolean expression over terms	evaluate boolean expression
Vector dictionary of *t* terms	*t*-dimensional vector	*t*-dimensional vector	vector measure of similarity Doc.s ranked by score
Extended	set of pairs (term, properties)	sequence of terms	Boolean AND Doc.s ranked; flexible scoring algorithm

Query models advantages

- Boolean
  - - No ranking in pure
  - + Get power of Boolean Algebra:
    - expressiveness
    - optimization of query forms
- Vector
  - + Query and document look the same
  - + Power of linear algebra
  - - No requirement all terms present in pure

Query models advantages

- Extended
  - + could use full Boolean Algebra to define satisfying documents
  - - query and document not same
  - ranking arbitrary function of document and query
    - - computational cost?