Welcome to the class of Web Information Retrieval!

ZHANG Min
z-m@tsinghua.edu.cn
Outline

- What is IR?
- Basic IR procedure
  - Data acquisition
  - Indexing
  - Ranking
    - Term weighting
    - Ranking models
  - System evaluation
Term Weighting

- Not all terms are equally important
- How to select *important / good* keywords?
  - Simplest way: using middle-frequency words according to Zipf’s law
Zipf's Law

- rank * frequency \( \approx \) constant

\[
F(w) = \frac{C}{r(w)^\alpha} \quad \alpha \approx 1, C \approx 0.1
\]

Generalized Zipf's law:

\[
F(w) = \frac{C}{[r(w) + B]^\alpha}
\]

Most useful words (Luhn 57)

Is “too rare” a problem?

Word Rank (by Freq)

Word Freq.

Biggest data structure (stop words)

Applicable in many domains
Term weighting

- **TFIDF**

  \[
  \text{tf} = \text{term frequency} \\
  \quad \text{(the frequency of a term/keyword in a document)}
  \]

  \[
  \text{idf} = \text{inverse document frequency} \\
  \quad \text{(the unevenness of term distribution in the corpus)}
  \]

  \[
  \text{weight}(t, D) = \text{tf}(t, D) \times \text{idf}(t)
  \]

- Some commonly used \(\text{tf} \times \text{idf}\) schemes:

  \[
  \begin{align*}
  \text{tf}(t, D) &= \text{freq}(t, D) & \text{idf}(t) &= \log \left( \frac{N}{n + \alpha} \right) \\
  \text{tf}(t, D) &= \log[\text{freq}(t, D)] & n &= \# \text{ docs containing } t \\
  \text{tf}(t, D) &= \log[\text{freq}(t, D)+1] & N &= \# \text{ docs in the corpus} \\
  \text{tf}(t, D) &= \frac{\text{freq}(t,d)}{\text{Max}[f(t,d)]} & \alpha &= 1, 0.5, \text{ etc} \\
  \end{align*}
  \]

- Sometimes, additional normalizations (e.g. length).
Ranking

The problems underlying ranking
- How is a document/query represented with the selected keywords?
- How are two representations compared?
- (ordered) measure of relevance
Overview of Retrieval Models

- **Relevance**: $\Delta(\text{Rep}(q), \text{Rep}(d))$, $P(r=1|q,d)$, $r \in \{0,1\}$, $P(d \rightarrow q)$ or $P(q \rightarrow d)$
- **Similarity**: $\text{Set theory, Boolean algebra}$
- **Different Boolean model**
  - Boolean model (Ogawa, et al, 91)
  - Fuzzy set model
  - Vector space model (Salton et al., 75)
  - Prob. distr. model (Wong & Yao, 89)
- **Regression Model** (Fox 83)
- **Learn to Rank** (Joachims 02)
  - (Burges et al. 05)
- **Generative Model**
  - Doc generation
  - Query generation
  - Classical prob. Model (Robertson & Sparck Jones, 76)
  - LM approach (Ponte & Croft, 98)
  - (Lafferty & Zhai, 01a)
- **Prob. concept space model** (Wong & Croft, 91)
- **Inference network model** (Turtle & Croft, 91)

**IR fundamental techniques**: set theory, Boolean algebra, Prob. distr. model (Wong & Yao, 89),...
Overview of Retrieval Models

Relevance

\[ \Delta(\text{Rep}(q), \text{Rep}(d)) \]

Probability of Relevance

\[ P(r=1|q,d) \quad r \in \{0,1\} \]

Probabilistic inference

\[ P(d \rightarrow q) \text{ or } P(q \rightarrow d) \]

Different inference system

Doc generation

Query generation

Prob. concept space model

LM approach

Classical prob. Model

Generative Model

Learn to Rank

Regression Model

Different inference system

Inference network model

Boolean model

Fuzzy set model

Vector space model

Prob. distr. model

set theory, Boolean algebra

Different Boolean model

Different rep & similarity

IR fundamental techniques
### Boolean model

- **Boolean model**
  - Document = Logical conjunction of keywords
  - Query = Boolean expression of keywords
    - (AND, OR, NOT, with brackets)
  - \( R(D, Q) = D \rightarrow Q \)

- **Example:**
  
  \[ D = t_1 \land t_2 \land \ldots \land t_n \quad Q = (t_1 \land t_2) \lor (t_3 \land \neg t_4) \]

  We have \( D \rightarrow Q \). Thus \( R(D, Q) = 1 \).

- **Popular/earliest retrieval model because:**
  - Easy to understand for simple queries.
  - Clean formalism.
  - Reasonably efficient implementations possible for normal queries.
Boolean Models – Problems

- Very rigid: AND means all; OR means any.
- Difficult to express complex user requests.
- Difficult to control the number of documents retrieved.
  - All matched documents will be returned.
- Difficult to rank output.
  - All matched documents logically satisfy the query.
- Difficult to perform relevance feedback.
  - If a document is identified by the user as relevant or irrelevant, how should the query be modified?
Extensions to Boolean model

- $D = \{ ..., (t_i, a_i), ... \}$ i.e. keywords are weighted

- Interpretation:
  
  $D$ is a member of class $t_i$ to degree $a_i$.

  In terms of fuzzy sets:
  
  $\mu_{t_i}(D) = a_i$

- Evaluation:
  
  $R(D, t_i) = \mu_{t_i}(D)$

  $R(D, Q_1 \land Q_2) = \min(R(D, Q_1), R(D, Q_2))$.

  $R(D, Q_1 \lor Q_2) = \max(R(D, Q_1), R(D, Q_2))$.

  $R(D, \neg Q_1) = 1 - R(D, Q_1)$. 
Vector space model

- **Vector space** = all the keywords encountered
  \[ \langle t_1, t_2, t_3, ..., t_n \rangle \]
  Dimension = \( n = |\text{vocabulary}| \)

- **Document**: a weighted vector
  \[ D = \langle a_1, a_2, a_3, ..., a_n \rangle \]
  \( a_i = \text{weight of } t_i \text{ in } D \)

- **Query**: a weighted vector
  \[ Q = \langle b_1, b_2, b_3, ..., b_n \rangle \]
  \( b_i = \text{weight of } t_i \text{ in } Q \)

- **R(D,Q)** = Similarity(D,Q)
Example:

\[ D_1 = 2T_1 + 3T_2 + 5T_3 \]
\[ D_2 = 3T_1 + 7T_2 + T_3 \]
\[ Q = 0T_1 + 0T_2 + 2T_3 \]

- Is \( D_1 \) or \( D_2 \) more similar to \( Q \)?
- How to measure the degree of similarity?
  - Distance? Angle? Projection?
Similarity Measure - Inner Product

- Using vector inner product:
  $$\text{sim}(\mathbf{d}_j, \mathbf{q}) = \mathbf{d}_j \cdot \mathbf{q} = \sum_{i=1}^{t} w_{ij} \cdot w_{iq}$$

  - $w_{ij}$ -- the weight of term $i$ in document $j$
  - $w_{iq}$ -- the weight of term $i$ in the query

- For binary vectors, it’s # of matched query terms in the document (size of intersection).

- For weighted term vectors, it’s the sum of the products of the weights of the matched terms.
Properties of Inner Product

- The inner product is unbounded.

- Favors long documents with a large number of unique terms.

- Measures how many terms matched but not how many terms are not matched.
Cosine Similarity Measure

- **Cosine similarity** measures the cosine of the angle between two vectors.
- Inner product normalized by the vector lengths.

\[
\text{CosSim}(d_j, q) = \frac{\vec{d}_j \cdot \vec{q}}{||\vec{d}_j|| \cdot ||\vec{q}||} = \frac{\sum_{i=1}^{t} (w_{ij} \cdot w_{iq})}{\sqrt{\sum_{i=1}^{t} w_{ij}^2} \cdot \sqrt{\sum_{i=1}^{t} w_{iq}^2}}
\]

\[
D_1 = 2T_1 + 3T_2 + 5T_3 \quad \text{CosSim}(D_1, Q) = \frac{10}{\sqrt{(4+9+25)(0+0+4)}} = 0.81
\]
\[
D_2 = 3T_1 + 7T_2 + 1T_3 \quad \text{CosSim}(D_2, Q) = \frac{2}{\sqrt{(9+49+1)(0+0+4)}} = 0.13
\]
\[
Q = 0T_1 + 0T_2 + 2T_3
\]

**D_1** is 6 times better than **D_2** using cosine similarity but only 5 times better using inner product.
**Typical VSM weighting formula**

$$
\sum_{t \in Q, D} \frac{1 + \ln(1 + \ln(tf))}{(1 - s) + s \frac{dl}{avdl}} \times qtf \times \ln \frac{N + 1}{df}
$$

where $s$ is an empirical parameter (usually 0.20), and

- $tf$ is the term’s frequency in document
- $qtf$ is the term’s frequency in query
- $N$ is the total number of documents in the collection
- $df$ is the number of documents that contain the term
- $dl$ is the document length, and
- $avdl$ is the average document length.
Comments on Vector Space Models

- Simple, mathematically based approach.
- Provides partial matching and ranked results.
- Tends to work quite well in practice despite obvious weaknesses.
- Allows efficient implementation for large document collections.
Problems with Vector Space Model

- Missing **semantic** information (e.g. word sense).
- Missing **syntactic** information (e.g. phrase structure, word order, proximity information).
- Assumption of **term independence**.

*(not only the problems of VSM, but *The Flaws of Bag-of-Word models*)

- **Lacks the control of a Boolean model** (e.g., *requiring* a term to appear in a document).
  - Given a two-term query “A B”, may prefer a document containing A frequently but not B, over a document that contains both A and B, but both less frequently.
Probabilistic model

The Basic Question

- What is the probability that \textit{THIS} document is relevant to \textit{THIS} query?

Formally…

- 3 random variables:
  - query $Q$, document $D$, relevance $R \in \{0,1\}$
  - Given a particular query $q$, a particular document $d$, $p(R=1|Q=q, D=d) =$?
  - In fact, we only need to compare $P(R=1|Q, D_1)$ with $P(R=1|Q, D_2)$, i.e., rank documents
Probabilistic model: Generative models

- Basic idea
  - Define $P(Q,D|R)$
  - Compute $O(R=1|Q,D)$ using Bayes’ rule

\[
O(R=1|Q,D) = \frac{P(R=1|Q,D)}{P(R=0|Q,D)} = \frac{P(Q,D|R=1)}{P(Q,D|R=0)} \frac{P(R=1)}{P(R=0)}
\]

Ignored for ranking D

IR fundamental techniques 21
Probabilistic model: Generative models

- Basic idea
  - Define $P(Q,D|R)$
  - Compute $O(R=1|Q,D)$ using Bayes’ rule

  \[
  O(R=1|Q,D) = \frac{P(R=1|Q,D)}{P(R=0|Q,D)} = \frac{P(Q,D|R=1)}{P(Q,D|R=0)} \frac{P(R=1)}{P(R=0)}
  \]

  Ignored for ranking $D$

- Special cases
  - **Document “generation”:** $P(Q,D|R) = P(D|Q,R)P(Q|R)$
  - **Query “generation”:** $P(Q,D|R) = P(Q|D,R)P(D|R)$
Document Generation

\[
\frac{P(R=1|Q,D)}{P(R=0|Q,D)} \propto \frac{P(Q,D|R=1)}{P(Q,D|R=0)} \\
= \frac{P(D|Q,R=1)P(Q|R=1)}{P(D|Q,R=0)P(Q|R=0)} \\
\propto \frac{P(D|Q,R=1)}{P(D|Q,R=0)}
\]

Model of relevant docs for Q

Model of non-relevant docs for Q

Assume independent attributes \(A_1...A_k\) .... (generally we take terms as attributes)
Let \(D = d_1...d_k\), where \(d_k \in \{0,1\}\) is the value of attribute \(A_k\) (Similarly \(Q = q_1...q_k\))

\[
\frac{P(R=1|Q,D)}{P(R=0|Q,D)} \propto \prod_{i=1}^{k} \frac{P(A_i = d_i|Q,R=1)}{P(A_i = d_i|Q,R=0)} \\
rank \prod_{i=1}^{k} \frac{P(A_i = 1|Q,R=1)}{1 - P(A_i = 1|Q,R=0)} \cdot \frac{1 - P(A_i = 1|Q,R=0)}{P(A_i = 1|Q,R=0)} \\
\propto \sum_{i=1}^{k} \log \frac{p_i(1-q_i)}{(1-p_i)q_i}
\]

Robertson-Sparck Jones Model

Note: Non-query terms are equally likely to appear in relevant and non-relevant docs

\(p_i = P(A_i = 1|Q, R = 1)\): prob. that term \(A_i\) occurs in a relevant doc

\(q_i = P(A_i = 1|Q, R = 0)\): prob. that term \(A_i\) occurs in a non-relevant doc
Robertson-Sparck Jones Model
(Robertson & Sparck Jones 76)

\[
\log O(R = 1 | Q, D) \approx \sum_{i=1, q_i = 1}^{k} \log \frac{p_i(1 - q_i)}{q_i(1 - p_i)} \quad \text{(RSJ model)}
\]

Two parameters for each term \(A_i\):
\[p_i = P(A_i = 1 | Q, R = 1): \text{prob. that term } A_i \text{ occurs in a relevant doc} \]
\[q_i = P(A_i = 1 | Q, R = 0): \text{prob. that term } A_i \text{ occurs in a non-relevant doc} \]

How to estimate parameters (probabilities)?
Suppose we have relevance judgments,

\[
\hat{p}_i = \frac{\#(\text{rel. doc with } A_i) + 0.5}{\#(\text{rel. doc}) + 1} \quad \hat{q}_i = \frac{\#(\text{nonrel. doc with } A_i) + 0.5}{\#(\text{nonrel. doc}) + 1}
\]

“+0.5” and “+1” can be justified by Bayesian estimation
RSJ Model: No Relevance Info
(Croft & Harper 79)

\[ \log O(R = 1 | Q, D) \approx \sum_{i=1, q_i=1}^k \log \frac{p_i(1 - q_i)}{q_i(1 - p_i)} \]  
\text{ (RSJ model)}

How to estimate parameters (probabilities)?

Suppose we do not have relevance judgments,
- We will assume \( p_i \) to be a constant
- Estimate \( q_i \) by assuming all documents to be non-relevant

\[ \log O(R = 1 | Q, D) \approx \sum_{i=1, q_i=1}^k \log \frac{N - n_i + 0.5}{n_i + 0.5} \]

\( \text{IDF}' = \log \frac{N - n_i}{n_i} \)

\( N \): # documents in collection
\( n_i \): # documents in which term \( A_i \) occurs
RSJ Model: Summary

- The most important classic prob. IR model
- Use only term presence/absence, thus also referred to as Binary Independence Model
- Essentially Naïve Bayes for doc ranking
- Most natural for relevance/pseudo feedback
- When without relevance judgments, the model parameters must be estimated in an ad hoc way
- Performance isn’t as good as tuned VS model
- Many improvements …
Improvements -- BM25/Okapi Approximation
(Robertson et al. 94)

- Idea: Approximate \( p(R=1|Q,D) \) with a simpler function that share similar properties

- Observations:
  - \( \log O(R=1|Q,D) \) is a sum of term weights \( W_i \)
  - Adding TF \( (d_i \) is not only binary value 0/1)
    - \( W_i = 0, \text{ if } TF_i = 0 \)
    - \( W_i \) increases monotonically with \( TF_i \)
    - \( W_i \) has an asymptotic limit
  - Adding document length
    - “Carefully” penalize long doc
  - Adding query TF

\[
\log O(R = 1|Q, D) \approx \sum_{i=1, q_i=1}^{k} \log \frac{p_i(1-q_i)}{q_i(1-p_i)}
\]
The most famous ranking function in the doc generation branch – BM25 series

- **Final: BM25** – achieving top performances in TREC

\[
\sum_{T \in Q} w^{(1)} \frac{(k_1 + 1)tf}{K + tf} \frac{(k_3 + 1)qtf}{k_3 + qtf}
\]

\[
w^{(1)} = \log \frac{(r + 0.5)/(R - r + 0.5)}{(n - r + 0.5)/(N - n - R + r + 0.5)}
\]

\[
K = k_1((1 - b) + b \frac{|d|}{avdl})
\]

<table>
<thead>
<tr>
<th>Are the documents relevant to the term?</th>
<th>1 = Yes (Relevant)</th>
<th>0 = No (Non-Relevant)</th>
<th>Collection-wide Incidence</th>
</tr>
</thead>
<tbody>
<tr>
<td>Is the term present in the documents?</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 = Yes (Present)</td>
<td>r</td>
<td>n - r</td>
<td>n</td>
</tr>
<tr>
<td>0 = No (Absent)</td>
<td>R - r</td>
<td>N - n - R + r</td>
<td>N - n</td>
</tr>
</tbody>
</table>

- tf: the count of word T in the document d,
- qtf: the count of word T in the query q,
- |d|: the length of document d,
- avdl: the average document length of the collection,
- \(k_1\) (1.0 to 2.0), \(b\) (usually 0.75) and \(k_3\) (0 to 1000) : constants.
Query Generation Model

\[ O(R = 1 | Q, D) \propto \frac{P(Q, D | R = 1)}{P(Q, D | R = 0)} \]
\[ = \frac{P(Q | D, R = 1)P(D | R = 1)}{P(Q | D, R = 0)P(D | R = 0)} \]
\[ \propto P(Q | D, R = 1) \frac{P(D | R = 1)}{P(D | R = 0)} \]

(Assume \( P(Q | D, R = 0) \approx P(Q | R = 0) \))

Query likelihood \( p(q| \theta_d) \)  \hspace{1cm} Document prior

Assuming uniform prior, we have \( O(R = 1 | Q, D) \propto P(Q | D, R = 1) \)

Now, the question is how to compute \( P(Q | D, R = 1) \)?

Generally involves two steps:
1. estimate a language model based on \( D \)
2. compute the query likelihood according to the estimated model

Leading to the so-called “Language Modeling Approach” …
What is a Statistical LM?

- A probability distribution over word sequences
  - \( p(\text{"Today is Friday"}) \approx 0.001 \)
  - \( p(\text{"Today Friday is"}) \approx 0.0000000000000001 \)
  - \( p(\text{"The eigenvalue is positive"}) \approx 0.00001 \)

- Context-dependent!

- Can also be regarded as a probabilistic mechanism for “generating” text, thus also called a “generative” model
The Simplest Language Model
(Unigram Model)

- Generate a piece of text by generating each word independently
- Thus, \( p(w_1 \ w_2 \ ... \ \ w_n) = p(w_1)p(w_2)\ldots p(w_n) \)
- Parameters: \( \{p(w_i)\} \) \( p(w_1) + \ldots + p(w_N) = 1 \)
  - (\( N \) is voc. size)
- Essentially a multinomial distribution over words
- A piece of text can be regarded as a sample drawn according to this word distribution
Text Generation with Unigram LM

(Unigram) Language Model $\theta$

$$p(w|\theta)$$

Sampling → Document

### Topic 1: Text mining
- Text mining 0.2
- Mining 0.1
- Association 0.01
- Clustering 0.02
- Food 0.00001

### Topic 2: Health
- Food 0.25
- Nutrition 0.1
- Healthy 0.05
- Diet 0.02

Text mining paper

Food nutrition paper

IR fundamental techniques 32
**Estimation of Unigram LM**

(Unigram) Language Model $\theta$

$$p(w|\theta) = ?$$

**Estimation**

Document

- text ?
- mining ?
- association ?
- database ?
- query ?
- ...?

---

A "text mining paper"

(totol #words=100)

- text 10
- mining 5
- association 3
- database 3
- algorithm 2
- ...?
- query 1
- efficient 1
Language Models for Retrieval

Query = “data mining algorithms”

Which model would most likely have generated this query?

(Ponte & Croft 98)
Ranking Docs by Query Likelihood

- Document ranking based on *query likelihood*

\[
\log p(q \mid d) = \sum_i \log p(w_i \mid d) \\
\text{where, } \ q = w_1w_2\ldots w_n
\]

- Retrieval problem \( \approx \) Estimation of \( p(w_i \mid d) \)

**Document language model**
How to Estimate $p(w \mid d)$?

- Simplest solution: Maximum Likelihood Estimator
  - $P(w \mid d) = \text{relative frequency of word } w \text{ in } d$
  - What if a word doesn’t appear in the text? $P(w \mid d) = 0$

- In general, what probability should we give a word that has not been observed?
  - “smoothing”
Language Model Smoothing (Illustration)

Max. Likelihood Estimate

\[ p_{ML}(w) = \frac{\text{count of } w}{\text{count of all words}} \]

All smoothing methods try to

- Discount the probability of words seen in a document
- Re-allocate the extra counts so that unseen words will have a non-zero count

For more information, ref. to: Lafferty, J. and Zhai, C., 2003.
Smoothing & TF-IDF Weighting

- A general smoothing schema: using a reference model

\[
p(w \mid d) = \begin{cases} 
p_{\text{seen}}(w \mid d) & \text{if } w \text{ is seen in } d \\
\alpha_d p(w \mid C) & \text{otherwise} \end{cases}
\]

Discounted ML estimate
Collection language model

- Plug in the general smoothing scheme to the query likelihood retrieval formula, we obtain

\[
\log p(q \mid d) = \sum_{\substack{w_i \in d \\
\text{if } w_i \in q}} \left[ \log \frac{p_{\text{seen}}(w_i \mid d)}{\alpha_d p(w_i \mid C)} \right] + n \log \alpha_d + \sum_i \log p(w_i \mid C)
\]

TF weighting
Doc length normalization
(long doc is expected to have a smaller \(\alpha_d\))
IDF weighting
Ignore for ranking

- Smoothing with \(p(w/C) \approx \text{TF-IDF + length norm.}\)
Derivation of the Query Likelihood Retrieval Formula

Retrieval formula using the general smoothing scheme

\[
p(w | d) = \begin{cases} p_{\text{Seen}}(w | d) & \text{if } w \text{ is seen in } d \\ \alpha_d p(w | C) & \text{otherwise} \end{cases}
\]

Discounted ML estimate

\[
\alpha_d = 1 - \frac{\sum_{w \text{ is seen}} p_{\text{Seen}}(w | d)}{\sum_{w \text{ is unseen}} p(w | C)}
\]

Reference language model

Retrieval formula using the general smoothing scheme

\[
p(q | d) = \sum_{w \in V, c(w,q)>0} c(w,q) \log p(w | d)
\]

\[
= \sum_{w \in V, c(w,d)>0, c(w,q)>0} c(w,q) \log p_{\text{Seen}}(w | d) + \sum_{w \in V, c(w,q)>0, c(w,d)=0} c(w,q) \log \alpha_d p(w | C)
\]

\[
= \sum_{w \in V, c(w,d)>0} c(w,q) \log p_{\text{Seen}}(w | d) + \sum_{w \in V, c(w,q)>0, c(w,d)>0} c(w,q) \log \alpha_d p(w | C) - \sum_{w \in V, c(w,q)>0, c(w,d)>0} c(w,q) \log \alpha_d p(w | C)
\]

Key rewriting step

Similar rewritings are very common when using LMs for IR…
Homework

How many ways can you find to prevent your website being indexed by Search Engine?

Submission deadline: by March 8.
on http://learn.tsinghua.edu.cn
Robots.txt

- Protocol for giving spiders ("robots") limited access to a website, originally from 1994
  - www.robotstxt.org/orig.html

- Website announces its request on what can(not) be crawled
  - For a URL, create a file URL/robots.txt
  - This file specifies access restrictions

- Example: # robots.txt
  
  User-agent: *
  Disallow: /cyberworld/map/ # This is an infinite virtual URL space

  # Cybermapper knows where to go.
  User-agent: cybermapper
  Disallow: