Retrieval Models

闫宏飞
北京大学，信息科学技术学院，计算机系
http://net.pku.edu.cn/~yhf, yhf AT net.pku.edu.cn

Thanks to Bo Peng for sharing many of these slides.

CCF-ADL at Zhengzhou University, June 25-27, 2010
Outline

• Vector Space Model (VSM)
• Latent Semantic Model (LSI)
• Language Model (LM)
Simple flow of retrieval process

Information Need → Representation → Query → Comparison → Retrieved Objects → Evaluation/Feedback

Text Objects → Representation → Indexed Objects
Latent semantic indexing - Wikipedia, the free encyclopedia
Latent Semantic Indexing (LSI) is an indexing and retrieval method that uses a mathematical technique called Singular Value Decomposition (SVD) to identify ...

Benefits of LSI - LSI Timeline - Mathematics of LSI
en.wikipedia.org/wiki/Latent_semantic_indexing - Cached - Similar - ☰ ☱ ☱

Latent semantic analysis - Wikipedia, the free encyclopedia
C0,2-9. http://lsi.research.telcordia.com/lsi/papers/JASIS90.pdf. Original article where the model was first exposed. Michael Berry, S.T. Dumais, ...
Occurrence matrix - Applications - Rank lowering - Derivation
en.wikipedia.org/wiki/Latent_semantic_analysis - Cached - Similar - ☰ ☱

Google Semantically Related Words & Latent Semantic Indexing ...
Google recently strongly promoted the semantic relationships of words in their algorithm.
www.seobook.com/archives/000657.shtml - Cached - Similar - ☰ ☱ ☱

Latent Semantic Indexing
Latent semantic indexing adds an important step to the document indexing process. In addition to recording which keywords a document contains, ...
www.seobook.com/lsi/lisa_definition.htm - Cached - Similar - ☰ ☱ ☱

LSI - Latent Semantic Indexing Web Site
January 12, 2006 podcast interview of Michael W. Berry discussing LSI on the Good Karma Show hosted by Greg Niland (aka GoodROI) at WebmasterRadio.fm ...
www.cs.utk.edu/~lsi/ - Cached - Similar - ☰ ☱ ☱

Latent Semantic Indexing
CCF-ADL at Zhengzhou University,
June 25-27, 2010
Vector Space Model
Documents as vectors

- Each document $j$ can be viewed as a vector, where each term is a dimension, taking values as log-scaled tf.idf.
- So we have a vector space
  - terms are axes
  - docs live in this space
  - High-dimensional space: even after stemming, may have 20,000+ dimensions
Intuition

Postulate: In vector space "close together" documents talk about the same things.

Example: Query-by-example, Free Text query as vector
Cosine similarity

- 向量\(d_1\)和\(d_2\)的“closeness”可以用它们之间的夹角大小来度量
- 具体的，可用cosine of the angle来计算向量相似度.
- 向量按长度归一化

\[
\left| \overrightarrow{d}_j \right| = \sqrt{\sum_{i=1}^{M} w_{i,j}^2} = 1
\]

\[
sim(d_j, d_k) = \frac{\overrightarrow{d}_j \cdot \overrightarrow{d}_k}{\left| \overrightarrow{d}_j \right| \left| \overrightarrow{d}_k \right|} = \frac{\sum_{i=1}^{M} w_{i,j}w_{i,k}}{\sqrt{\sum_{i=1}^{M} w_{i,j}^2} \sqrt{\sum_{i=1}^{M} w_{i,k}^2}}
\]
Latent Semantic Model
Vector Space Model: Pros

• **Automatic** selection of index terms
• **Partial matching** of queries and documents *(dealing with the case where no document contains all search terms)*
• **Ranking** according to **similarity score** *(dealing with large result sets)*
• **Term weighting** schemes *(improves retrieval performance)*
• Various extensions
  – Document clustering
  – Relevance feedback *(modifying query vector)*
• **Geometric** foundation
I guess this page is about a blackberry...?
Problems with Lexical Semantics

• **Polysemy**: 词通常有 *multitude of meanings* 和不同用法。Vector Space Model 不能区分同一个词的不同含义，即 *ambiguity*.

\[
\text{sim}_{\text{true}}(d, q) < \cos(\angle(d, q))
\]

• **Synonymy**: 不同的 *terms* 可能具有 *identical or a similar meaning*. Vector Space Model 里不能表达词之间的 *associations*.

\[
\text{sim}_{\text{true}}(d, q) > \cos(\angle(d, q))
\]
Issues in the VSM

• terms之间的独立性假设
  – 有些terms更可能在一起出现
    • 同义词，相关词汇，拼写错误，etc.
  – 根据上下文，terms可能有不同的含义

• term-document矩阵维度很高

对每篇文档/每个词，真的有那么多重要的特征？
Singular Value Decomposition

- 对term-document矩阵作奇异值分解 *Singular Value Decomposition*
  - r, 矩阵的 rank
  - Σ, singular values 的对角阵（按降序排列）
  - D, T, 具有正交的单位长度列向量 (TT’=I, DD’=I)

\[ W_{td} = T \Sigma D^T \]

\[ \text{WW}^T \text{的特征值} \]

\[ \text{W}^TW \text{和WW}^T \text{的特征向量} \]
Singular Values

- $\Sigma$ gives an ordering to the dimensions
  - 值下降非常快
  - 尾部的singular values at 代表 "noise"
  - 在low-value dimensions截止可以减少 noise，提高性能
Low-rank Approximation

\[ w_{td} \approx T \]

\[ T \]
Latent Semantic Indexing (LSI)

• Perform a **low-rank approximation of term-document matrix** (typical rank 100-300)

• General idea
  
  – Map documents (*and* terms) to a **low-dimensional representation**.
  
  – Design a mapping such that the low-dimensional space reflects **semantic associations** (latent semantic space).
  
  – Compute document similarity based on the **inner product** in this **latent semantic space**
What it is

- From the original term-document matrix $A_r$, we calculate its approximation $A_k$.
- In $A_k$, each row corresponds to a term, and each column corresponds to a document.
- The difference is that documents are in a new space, its dimensions $k << r$.
- How to compare two terms?
  \[ A_k A_k^T = T \Sigma D^T D \Sigma^T T^T = (T \Sigma)(T \Sigma)^T \]
- How to compare two documents?
  \[ A_k^T A_k = D \Sigma^T T^T T \Sigma D^T = (\Sigma D^T)(\Sigma D) \]
- How to compare a term and a document?
  \[ A_k[i,j] \]

CCF-ADL at Zhengzhou University,
June 25-27, 2010
LSI Term matrix T

• T matrix
  – 每个term在LSI space的向量
  – 原始matrix: terms向量是d-dimensional，T中要小很多
  – Dimensions是在相同文档中倾向于与这个词“同现”的一组terms
    • synonyms, contextually-related words, variant endings
  – (TΣ) 用来计算term相似度
Document matrix D

• D matrix
  – In the LSI space, the document representation
  – The T vectors have the same dimensionality
  – The product $(\Sigma D^T)$ is used to calculate the document similarity
  – It is used to calculate the similarity between a query and a document
Retrieval with LSI

• LSI检索过程:
  – 查询映射/投影到LSI的$D^T$空间，称为“folded in“：
  – $W=TS\Sigma D^T$，若$q$投影到$D^T$中后为$q'$，则有
    
    $q = T\Sigma q'^T$

    – 既有$q' = (\Sigma^{-1}T^{-1}q)^T = qT\Sigma^{-1}$

    – Folded in 既为 document/query vector 乘上$T\Sigma^{-1}$

    – 文档集的文档向量为$\Sigma D^T$

    – 两者通过dot-product计算相似度
Improved Retrieval with LSI

• 性能提升来自...
  – 去除了 noise
  – 不需要 stem terms (variants will co-occur)
  – 不需要 stop list
  – 没有速度和空间上的改进, though...
### C =

<table>
<thead>
<tr>
<th></th>
<th>$d_1$</th>
<th>$d_2$</th>
<th>$d_3$</th>
<th>$d_4$</th>
<th>$d_5$</th>
<th>$d_6$</th>
</tr>
</thead>
<tbody>
<tr>
<td>ship</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>boat</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>ocean</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>voyage</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>trip</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

### $T_r =$

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>ship</td>
<td>-0.44</td>
<td>-0.30</td>
<td>0.57</td>
<td>0.58</td>
<td>0.25</td>
</tr>
<tr>
<td>boat</td>
<td>-0.13</td>
<td>-0.33</td>
<td>-0.59</td>
<td>0.00</td>
<td>0.73</td>
</tr>
<tr>
<td>ocean</td>
<td>-0.48</td>
<td>-0.51</td>
<td>-0.37</td>
<td>0.00</td>
<td>-0.61</td>
</tr>
<tr>
<td>voyage</td>
<td>-0.70</td>
<td>0.35</td>
<td>0.15</td>
<td>-0.58</td>
<td>0.16</td>
</tr>
<tr>
<td>trip</td>
<td>-0.26</td>
<td>0.65</td>
<td>-0.41</td>
<td>0.58</td>
<td>-0.09</td>
</tr>
</tbody>
</table>

### $\Sigma_r =$

<table>
<thead>
<tr>
<th></th>
<th>2.16</th>
<th>0.00</th>
<th>0.00</th>
<th>0.00</th>
<th>0.00</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.00</td>
<td>1.59</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>0.00</td>
<td>0.00</td>
<td>1.28</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>1.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.39</td>
<td>0.00</td>
</tr>
</tbody>
</table>
\[ D_r^T = \begin{array}{cccccc}
1 & -0.75 & -0.28 & -0.20 & -0.45 & -0.33 & -0.12 \\
2 & -0.29 & -0.53 & -0.19 & 0.63 & 0.22 & 0.41 \\
3 & 0.28 & -0.75 & 0.45 & -0.20 & 0.12 & -0.33 \\
4 & 0.00 & 0.00 & 0.58 & 0.00 & -0.58 & 0.58 \\
5 & -0.53 & 0.29 & 0.63 & 0.19 & 0.41 & -0.22 \\
\end{array} \]

\[ \Sigma_2 = \begin{array}{cccccc}
2.16 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\
0.00 & 1.59 & 0.00 & 0.00 & 0.00 & 0.00 \\
0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\
0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\
0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\
\end{array} \]

\[ \Sigma_2 D_2^T = \begin{array}{cccccc}
1 & -1.62 & -0.60 & -0.44 & -0.97 & -0.70 & -0.26 \\
2 & -0.46 & -0.84 & -0.30 & 1.00 & 0.35 & 0.65 \\
3 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\
4 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\
5 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\
\end{array} \]
Example

- Map into 2-dimension space

<table>
<thead>
<tr>
<th></th>
<th>$d_1$</th>
<th>$d_2$</th>
<th>$d_3$</th>
<th>$d_4$</th>
<th>$d_5$</th>
<th>$d_6$</th>
</tr>
</thead>
<tbody>
<tr>
<td>ship</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>boat</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>ocean</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>voyage</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>trip</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>$d_1$</th>
<th>$d_2$</th>
<th>$d_3$</th>
<th>$d_4$</th>
<th>$d_5$</th>
<th>$d_6$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-1.62</td>
<td>-0.60</td>
<td>-0.44</td>
<td>-0.97</td>
<td>-0.70</td>
<td>-0.26</td>
</tr>
<tr>
<td>2</td>
<td>-0.46</td>
<td>-0.84</td>
<td>-0.30</td>
<td>1.00</td>
<td>0.35</td>
<td>0.65</td>
</tr>
<tr>
<td>3</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>4</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>5</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
</tbody>
</table>
Latent Semantic Analysis

- **Latent semantic space**: illustrating example

courtesy of Susan Dumais
Empirical evidence

• Experiments on TREC 1/2/3 — Dumais
• Precision at or above median TREC precision
  — Top scorer on almost 20% of TREC topics
• Slightly better on average than straight vector spaces
• Effect of dimensionality:

<table>
<thead>
<tr>
<th>Dimensions</th>
<th>Precision</th>
</tr>
</thead>
<tbody>
<tr>
<td>250</td>
<td>0.367</td>
</tr>
<tr>
<td>300</td>
<td>0.371</td>
</tr>
<tr>
<td>346</td>
<td>0.374</td>
</tr>
</tbody>
</table>
LSI has many other applications

• 在很多场合，我们都有 feature-object matrix.
  - 矩阵是高维，有大量冗余，从而能使用 low-rank approximation.
  - 比如文本检索，the terms 是 features，the docs 是 objects. \(\rightarrow\) Latent Semantic Index
  - 比如 opinions 和 users … \(\rightarrow\)
  - 数据不全 (e.g., users’ opinions), 可以在低维空间里恢复.

• Powerful general analytical technique
Language Models
IR based on Language Model (LM)

- 通常的search方法：猜测作者写相关文档时使用的词，形成query
- The LM approach directly exploits that idea!
Formal Language (Model)

• 传统的生成模型 *generative model*: 产生 *strings*
  – Finite state machines or regular grammars, etc.

• Example:

\[
\text{(I wish)}^* \\
\]

(I wish) *
Stochastic Language Models

- Models *probability* of generating strings in the language (commonly all strings over alphabet $\sum$)

<table>
<thead>
<tr>
<th>Probability</th>
<th>Word</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.2</td>
<td>the</td>
</tr>
<tr>
<td>0.1</td>
<td>a</td>
</tr>
<tr>
<td>0.01</td>
<td>man</td>
</tr>
<tr>
<td>0.01</td>
<td>woman</td>
</tr>
<tr>
<td>0.03</td>
<td>said</td>
</tr>
<tr>
<td>0.02</td>
<td>likes</td>
</tr>
</tbody>
</table>

...  

```
the                  man                  likes                  the                  woman
___                  ___                  ___                    ___                    ___
0.2                  0.01                 0.02                   0.2                  0.01
```

Multiply

$P(s \mid M) = 0.00000008$
Stochastic Language Models

- Model *probability* of generating any string

<table>
<thead>
<tr>
<th>Model M1</th>
<th>Model M2</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.2 the</td>
<td>0.2 the</td>
</tr>
<tr>
<td>0.01 class</td>
<td>0.0001 class</td>
</tr>
<tr>
<td>0.0001 sayst</td>
<td>0.03 sayst</td>
</tr>
<tr>
<td>0.0001 pleaseth</td>
<td>0.02 pleaseth</td>
</tr>
<tr>
<td>0.0001 yon</td>
<td>0.1 yon</td>
</tr>
<tr>
<td>0.0005 maiden</td>
<td>0.01 maiden</td>
</tr>
<tr>
<td>0.01 woman</td>
<td>0.0001 woman</td>
</tr>
</tbody>
</table>

\[
P(s|M2) > P(s|M1)
\]
Stochastic Language Models

- 用来生成文本的统计模型
  
  - Probability distribution over strings in a given language

\[
P(\bullet \bullet \bullet \bullet|\ M) = P(\bullet|\ M) \\
P(\bullet|\ M, \bullet) \\
P(\bullet \bullet|\ M, \bullet) \\
P(\bullet \bullet \bullet \bullet|\ M, \bullet \bullet \bullet)
\]
Unigram and higher-order models

\[ P(\bullet \bullet \bullet \bullet \bullet) = P(\bullet)P(\bullet | \bullet)\ P(\bullet | \bullet \bullet \bullet \bullet)P(\bullet | \bullet \bullet \bullet \bullet \bullet) \]

- **Unigram** Language Models
  \[ P(\bullet)P(\bullet)P(\bullet)P(\bullet) \]

- **Bigram** (generally, \(n\)-gram) Language Models
  \[ P(\bullet)P(\bullet | \bullet)P(\bullet | \bullet) \]

- **Other Language Models**
  - Grammar-based models (PCFGs), etc.
  - Probably not the first thing to try in IR
The fundamental problem of LMs

• 模型 $M$ 是不知道的
  – 只有代表这个模型的样例文本
    $$P(\cdots | M(\cdots \cdots \cdots))$$

• 从样例文本中来估计Model
• 然后计算观察到的文本概率
Using Language Models in IR

• 每篇文档对应一个 model
• 按 $P(d \mid q)$ 对文档排序
• $P(d \mid q) = P(q \mid d) \times P(d) / P(q)$
  – $P(q)$ is the same for all documents, so ignore
  – $P(d)$ [the prior] is often treated as the same for all $d$
    • But we could use criteria like authority, length, genre
      – $P(q \mid d)$ is the probability of $q$ given $d$’s model
• Very general formal approach
Language Models for IR

• Language Modeling Approaches
  – 为 query generation process 建模
  – 文档排序：对一个query作为由文档模型产生的随机样本而被观察到的概率 the probability that a query would be observed as a random sample from the respective document model
    • Multinomial approach

\[ P(q|M_d) = \prod_{w \in V} P(w|M_d)^{tf_w} \]
Retrieval based on probabilistic LM

- 把query的产生当作一个随机过程
- 方法
  - 为每个文档Infer a language model.
  - Estimate the probability: 估计每个文档模型产生这个query的概率
  - Rank: 按这个概率对文档排序.
  - 通常使用Unigram model
Query generation probability (1)

- 排序公式

\[
p(Q, d) = p(d)p(Q | d)
\approx p(d)p(Q | M_d)
\]

- 用最大似然估计:

\[
\hat{p}(Q | M_d) = \prod_{t \in Q} \hat{p}_{ml}(t | M_d)
\]

\[
= \prod_{t \in Q} \frac{tf_{(t,d)}}{dl_d}
\]

**Unigram assumption:**

Given a particular language model, the query terms occur independently

- \(M_d\): language model of document \(d\)
- \(tf_{(t,d)}\): raw tf of term \(t\) in document \(d\)
- \(dl_d\): total number of tokens in document \(d\)
Insufficient data

• Zero probability $p(t | M_d) = 0$
  – 一个文档里没有query中的某个term时…

• General approach
  – 没有出现文档中的term按它出现在collection中的概率来代替.
  – If $tf_{(t,d)} = 0$, $p(t | M_d) = \frac{cf_t}{cs}$

$cf_t$: raw count of term t in the collection
$cs$: raw collection size(total number of tokens in the collection)
Insufficient data

• Zero probabilities spell disaster
  – 使用平滑：smooth probabilities
    • Discount nonzero probabilities
    • Give some probability mass to unseen things
    • 有很多方法，如adding 1, ½ or ε to counts, Dirichlet priors, discounting, and interpolation
    • [See FSNLP ch. 6 if you want more]
  – 使用混合模型：use a mixture between the document multinomial and the collection multinomial distribution
Mixture model

- $P(w | d) = \lambda P_{mle}(w | M_d) + (1 - \lambda)P_{mle}(w | M_c)$

- 参数$\lambda$很重要
  - $\lambda$ 值高，使得查询成为“conjunctive-like” — 适合短查询
  - $\lambda$ 值低更适合长查询
  - 调整$\lambda$ 来优化性能
    - 比如使得它与文档长度相关 (cf. Dirichlet prior or Witten-Bell smoothing)
Basic mixture model summary

- General formulation of the LM for IR

\[ p(Q, d) = p(d) \prod_{t \in Q} ((1 - \lambda) p(t) + \lambda p(t | M_d)) \]

- General language model
- Individual-document model
Example

- Document collection (2 documents)
  - d₁: Xerox reports a profit but revenue is down
  - d₂: Lucent narrows quarter loss but revenue decreases further

- Model: MLE unigram from documents; λ = ½

- Query: revenue down
  - P(Q|d₁)
    - = [(1/8 + 2/16)/2] x [(1/8 + 1/16)/2]
    - = 1/8 x 3/32 = 3/256
  - P(Q|d₂)
    - = [(1/8 + 2/16)/2] x [(0 + 1/16)/2]
    - = 1/8 x 1/32 = 1/256

- Ranking: d₁ > d₂
Alternative Models of Text Generation

\[ P(M | \text{Searcher}) \quad P(\text{Query} | M) \]

\[ P(M | \text{Writer}) \quad P(\text{Doc} | M) \]

Is this the same model?
Retrieval Using Language Models

- Query likelihood (1)
- Document likelihood (2)
- Model comparison (3)
Query Likelihood

• \( P(Q|D_m) \)
• 主要问题是估计文档模型
  – i.e. smoothing techniques instead of tf.idf weights
• 检索效果不错
  – e.g. UMass, BBN, Twente, CMU
• 问题：处理relevance feedback, query expansion, structured queries困难
Document Likelihood

- 按 $P(D|R)/P(D|NR)$ 排序
  - $P(w|R)$ is estimated by $P(w|Q_m)$, $Q_m$ is the query or relevance model
  - $P(w|NR)$ is estimated by collection probabilities $P(w)$

- 问题是估计 relevance model
  - Treat query as generated by mixture of topic and background
  - Estimate relevance model from related documents (query expansion)
  - Relevance feedback is easily incorporated

- Good retrieval results
  - e.g. UMass at SIGIR 01
  - inconsistent with heterogeneous document collections
Model Comparison

• 估计query和document模型，进行模型比较
• KL divergence \( D(Q_m || D_m) \)
  \[
  D(Q_m || D_m) = \sum_{x \in X} Q_m(x) \log \frac{Q_m(x)}{D_m(x)}
  \]

• 取得了较前两方法更好的效果
Language models: pro & con

• Novel way of looking at the problem of text retrieval based on probabilistic language modeling
  – Conceptually simple and explanatory
  – Formal mathematical model
  – Natural use of collection statistics, not heuristics (almost…)

• LMs provide effective retrieval and can be improved to the extent that the following conditions can be met
  – language models are accurate representations of the data.
  – Users have some sense of term distribution.
Comparison With Vector Space

- 和传统的tf.idf models有一定联系:
  - (unscaled) term frequency is directly in model
  - the probabilities do length normalization of term frequencies
  - the effect of doing a mixture with overall collection frequencies is a little like idf: terms rare in the general collection but common in some documents will have a greater influence on the ranking
Comparison With Vector Space

• 相似点
  – Term weights based on frequency
  – Terms often used as if they were independent
  – Inverse document/collection frequency used
  – Some form of length normalization used

• 不同点
  – Based on probability rather than similarity
  – Intuitions are probabilistic rather than geometric
  – Details of use of document length and term, document, and collection frequency differ
Summary

- Latent Semantic Indexing
  - singular value decomposition
  - Matrix Low-rank Approximation

- Language Model
  - Generative model
  - smooth probabilities
  - Mixture model

\[
P(q|M_d) = \prod_{w \in V} P(w|M_d)^{tf_w}
\]

\[
p(Q,d) = p(d) \prod_{t \in Q} ((1 - \lambda) p(t) + \lambda p(t | M_d))
\]