III.4 Statistical Language Models

1. Basics of Statistical Language Models
2. Query-Likelihood Approaches
3. Smoothing Methods
4. Divergence Approaches
5. Extensions

Based on MRS Chapter 12 and [Zhai 2008]
1. Basics of Statistical Language Models

- Statistical language models (LMs) are **generative models of word sequences** (or, bags of words, sets of words, etc.)

- Application examples:
  - **Speech recognition**, e.g., to select among multiple phonetically similar sentences ("get up at 8 o’clock" vs. "get a potato clock")
  - **Statistical machine translation**, e.g., to select among multiple candidate translations ("logical closing" vs. "logical reasoning")
  - **Information retrieval**, e.g., to rank documents in response to a query

\[
P(\langle \text{hog} \rangle) = 0.1 \times 0.1
\]
\[
P(\langle \text{cat, dog} \rangle) = 0.4 \times 0.9 \times 0.5 \times 0.1
\]
\[
P(\langle \text{dog, dog, hog} \rangle) = 0.5 \times 0.9 \times 0.5 \times 0.9 \times 0.1 \times 0.1
\]
Types of Language Models

• **Unigram LM** based on only **single words** (unigrams), considers **no context**, and assumes **independent generation** of words

\[ P(\langle t_1, \ldots, t_m \rangle) = \prod_{i=1}^{m} P(t_i) \]

• **Bigram LM** conditions on the preceding term

\[ P(\langle t_1, \ldots, t_m \rangle) = P(t_1) \prod_{i=2}^{m} P(t_i|t_{i-1}) \]

• **n-Gram LM** conditions on the preceding \((n-1)\) terms

\[ P(\langle t_1, \ldots, t_m \rangle) = P(t_1) P(t_2|t_1) \ldots \prod_{i=n}^{m} P(t_i|t_{i-n+1} \ldots t_{i-1}) \]
Parameter Estimation

- **Parameters** (e.g., $P(t_i), P(t_i \mid t_{i-1})$) of language model $\theta$ are estimated based on a sample of documents, which are assumed to have been generated by $\theta$

- **Example**: Unigram language models $\theta_{\text{Sports}}$ and $\theta_{\text{Politics}}$ estimated from documents about sports and politics

<table>
<thead>
<tr>
<th>$\theta_{\text{Sports}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>soccer                   : 0.20</td>
</tr>
<tr>
<td>goal                     : 0.15</td>
</tr>
<tr>
<td>tennis                   : 0.10</td>
</tr>
<tr>
<td>player                   : 0.05</td>
</tr>
</tbody>
</table>

\[ \cdots \cdots \quad \text{Sample} \quad \cdots \cdots \]

\[ \cdots \cdots \quad \text{generates} \quad \cdots \cdots \]

<table>
<thead>
<tr>
<th>$\theta_{\text{Politics}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>party                       : 0.20</td>
</tr>
<tr>
<td>debate                      : 0.20</td>
</tr>
<tr>
<td>scandal                     : 0.15</td>
</tr>
<tr>
<td>election                    : 0.05</td>
</tr>
</tbody>
</table>

\[ \cdots \cdots \quad \text{Sample} \quad \cdots \cdots \]

\[ \cdots \cdots \quad \text{generates} \quad \cdots \cdots \]
Probabilistic IR vs. Statistical Language Models

$P[R|d, q]$  

"User finds document $d$ relevant to query $q$"

\[ \propto \frac{P[R|d, q]}{P[\bar{R}|d, q]} \]

Probabilistic IR ranks according to relevance odds

\[ \propto \frac{P[q,d|R]}{P[q,d|\bar{R}]} \]

\[ = \frac{P[q|d,R]}{P[q|d,\bar{R}]} \frac{P[R|d]}{P[R|d]} \]

Statistical LMs rank according to query likelihood
2. Query-Likelihood Approaches

• $P(q|d)$ is the **likelihood that the query was generated** by the language model $\theta_d$ estimated from document $d$

• **Intuition:**
  - User formulates query $q$ by selecting words from a **prototype document**
  - Which document is “closest” to that prototype document
Multi-Bernoulli LM

- Query $q$ is seen as a set of terms and generated from document $d$ by tossing a coin for every word from the vocabulary $V$

$$ P(q|d) = \prod_{t \in q} P(t|d) \times \prod_{t \in V \setminus q} (1 - P(t|d)) $$

$$ \approx \prod_{t \in q} P(t|d) \quad (\text{assuming } |q| << |V|) $$

- [Ponte and Croft ’98] pioneered the use of LMs in IR
Multinomial LM

- Query $q$ is seen as a **bag of terms** and generated from document $d$ by **drawing terms** from the bag of terms corresponding to $d$

\[
P(q|d) = \left( \prod_{|q|} P(t_f(t_1, q) \ldots t_f(t_{|q|}, q)) \right) \prod_{t_i \in q} P(t_i|d)^{t_f(t_i, q)}
\]

\[
\propto \prod_{t_i \in q} P(t_i|d)^{t_f(t_i, q)}
\]

\[
\approx \prod_{t_i \in q} P(t_i|d) \quad \text{(assuming } \forall t_i \in q : t_f(t_i, q) = 1)\]

- **Multinomial LM** is more expressive than **Multi-Bernoulli LM** and therefore **usually preferred**
Multinomial LM (cont’d)

• Maximum-likelihood estimate for parameters $P(t_i|d)$

$$P(t_i|d) = \frac{tf(t_i, d)}{|d|}$$

is prone to overfitting and leads to

• bias in favor of **short documents** / against **long documents**

• conjunctive query semantics, i.e., query can not be generated from language models of documents that miss one of the query terms
3. Smoothing

- Smoothing methods avoid *overfitting* to the sample (often: one document) and are *essential* for LMs to work in practice
  - Laplace smoothing (cf. Chapter III.3)
  - Absolute discounting
  - Jelinek-Mercer smoothing
  - Dirichlet smoothing
  - Good-Turing smoothing
  - Katz’s back-off model
  - …

- **Choice** of smoothing method and *parameter setting* still mostly “*black art*” (or empirical, i.e., based on training data)
Jelinek-Mercer Smoothing

• Uses a **linear combination** (mixture) of document language model $\theta_d$ and **document-collection language model** $\theta_D$

\[
P(t|d) = \lambda \frac{tf(t, d)}{|d|} + (1 - \lambda) \frac{tf(t, D)}{|D|}
\]

with document $D$ as concatenation of entire document collection

• Parameter $\lambda$ can be tuned by **cross-validation** with held-out data
  • divide set of relevant $(q, d)$ pairs into $n$ partitions
  • build LM on the pairs from $n-1$ partitions
  • choose $\lambda$ to maximize precision (or recall or F1) on held-out partition
  • iterate with different choice of $n^{th}$ partition and average

• Parameter $\lambda$ can be made **document- or term-dependent**
Jelinek-Mercer Smoothing vs. TF*IDF

\[ P(q|d) = \prod_{t \in q} P(t|d) \]

\[ = \prod_{t \in q} \left( \lambda \frac{tf(t,d)}{|d|} + (1 - \lambda) \frac{tf(t,D)}{|D|} \right) \]

\[ \propto \sum_{t \in q} \log \left( \lambda \frac{tf(t,d)}{|d|} + (1 - \lambda) \frac{tf(t,D)}{|D|} \right) \]

\[ \propto \sum_{t \in q} \log \left( 1 + \frac{\lambda}{1-\lambda} \frac{tf(t,d)}{|d|} \frac{|D|}{tf(t,D)} \right) \]

\[ \sim tf \quad \sim idf \]

- (Jelinek-Mercer) smoothing has **effect similar to IDF weighting**
- Jelinek-Mercer smoothing leads to a **TF*IDF-style model**
Dirichlet-Prior Smoothing

- **Uses Bayesian estimation** with a conjugate Dirichlet prior instead of the Maximum-Likelihood Estimation

\[ P(t|d) = \frac{tf(t, d) + \alpha \frac{tf(t, D)}{|D|}}{|d| + \alpha} \]

- **Intuition**: Document \(d\) is **extended by \(\alpha\) terms** generated by the document-collection language model

- **Parameter \(\alpha\)** usually set as multiple of **average document length**
Dirichlet Smoothing vs. Jelinek-Mercer Smoothing

\[ P(t|d) = \lambda \frac{tf(t,d)}{|d|} + (1 - \lambda) \frac{tf(t,D)}{|D|} \]

\[ = \frac{|d|}{|d|+\alpha} \frac{tf(t,d)}{|d|} + \frac{\alpha}{|d|+\alpha} \frac{tf(t,D)}{|D|} \quad \text{(set } \lambda = \frac{|d|}{|d|+\alpha}) \]

\[ = \frac{tf(t,d) + \alpha \frac{tf(t,D)}{|D|}}{|d|+\alpha} \]

- Jelinek-Mercer smoothing with document-dependent \( \lambda \)
  becomes a special case of Dirichlet smoothing
4. Divergence Approaches

- Query-likelihood approaches see query as a sample from a LM.
- Query expansion, relevance feedback, etc. are difficult to express as query-likelihood approaches, since they would require tinkering with the sample (i.e., the query) and more fine-grained control than adding/removing terms.
Kullback-Leibler Divergence

- Kullback-Leibler divergence (aka. information gain or relative entropy) is an information-theoretic non-symmetric measure of distance between probability distributions

\[ D(\theta_q || \theta_d) = \sum_{t \in V} P(t | \theta_q) \log \frac{P(t | \theta_q)}{P(t | \theta_d)} \]

- Example:

<table>
<thead>
<tr>
<th>(\theta_q)</th>
<th>apple : 0.50</th>
<th>muffin : 0.50</th>
</tr>
</thead>
</table>

| \(\theta_d\) | apple : 0.25 | muffin : 0.25 | recipe : 0.10 | water : 0.10 | sugar : 0.30 |

\[ D(\theta_q || \theta_d) = P(\text{apple} | \theta_q) \log \frac{P(\text{apple} | \theta_q)}{P(\text{apple} | \theta_d)} + P(\text{muffin} | \theta_q) \log \frac{P(\text{muffin} | \theta_q)}{P(\text{muffin} | \theta_d)} \]

\[ = 0.50 \log \frac{0.50}{0.25} + 0.50 \log \frac{0.50}{0.25} \]

\[ = 1.00 \]
Relevance Feedback LM

• [Zhai and Lafferty ’01] re-estimate query language model as

\[ P(t|\theta'_q) = (1 - \alpha) P(t|\theta_q) + \alpha P(t|\theta_F) \]

with \( F \) as the set of documents with positive feedback from user

• MLE of \( \theta_F \) obtained by maximizing log-likelihood function

\[
\log P(F|\theta_F) = \sum_{t \in V} tf(t, F) \log ((1 - \lambda) P(t|\theta_F) + \lambda P(t|\theta_D))
\]

with \( tf(t, F) \) as the total term frequency of \( t \) in documents from \( F \) and \( \theta_D \) as the document-collection language model
5. Extensions

• Statistical language models have been one of the highly active areas in IR research during the past decade and continue to be

• Extensions:

  • Term-specific and document-specific smoothing
    (JM-style smoothing with term-specific $\lambda_t$ or document-specific $\lambda_d$)

  • (Semantic) Translation LMs
    (e.g., to consider synonyms or support cross-lingual IR)

  • Time-based LMs
    (e.g., with time-dependent document prior to favor recent documents)

  • LMs for (semi-)structured XML and RDF data
    (e.g., for entity search or question answering)

  • …
Translation LM for Cross-Lingual IR

- **Cross-Lingual IR:**
  - Users issue *queries in their native language* (e.g., German) (e.g., "spionage usa bundesregierung")
  - System returns *documents in another known language* (e.g., English)  (e.g., "reactions of the German government to U.S. eavesdropping on ...")

\[
P(q|d) = \prod_{t \in q} \sum_{w} P(t|w) P(w|d)
\]

- **Translation probabilities** $P(t|w)$ obtained from a *dictionary* or estimated based on a *parallel cross-lingual corpus*

- [Federico and Bertoldi ’01] as *more advanced approach* based on a Hidden-Markov Model that also considers *term contexts*
Time-Based LMs

- **Intuition**: For news-related queries (e.g., german election) documents published more recently are often preferable

- [Li and Croft ’03] rank documents according to

\[
P(q|d) P(d^t) = \left( \prod_{t \in q} P(t|d^t) \right) \left( \lambda e^{-\lambda (\text{now} - t)} \right)
\]

with document publication timestamp \( t \) and time-dependent exponentially decaying document prior \( P(d^t) \)

- [Peetz and de Rijke ’13] consider other document priors motivated by cognitive psychology research on human memory
LM for Entity Search

- **Objective**: Retrieve *entities* (e.g., people, locations, organizations) relevant to query $q$ as opposed to only documents [Ni et al. ‘07]

  **Candidate Entities:**
  1. Arjen Robben
  2. Rafael van der Vaart
  3. Louis van Gaal
  4. Daniel van Buyten
  5. Toni Kroos

- **Language model** $\theta_e$ for entity $e$ can be *estimated from contexts* in which the entity is mentioned in the document collection, possibly taking into account extraction accuracy.

**Query $q$:**
dutch soccer player munich

...munich’s flying dutchman...
...one of bayern’s most valuable players...
...winning soccer’s most prestigious champions league...
...with the dutch national team...
Summary of III.4

- **Statistical language models**
  widely used in natural language applications other than IR

- **Query-likelihood approaches**
  see the query as a sample from the document LM

- **Divergence approaches**
  are more expressive comparing query LM against document LM

- **Smoothing methods**
  are absolutely essential to make LMs work in practice

- **Various extensions**
  for advanced tasks such as cross-lingual IR or entity search
Additional Literature for III.4


- **Z. Nie, Y. Ma, S. Shi, J.-R. Wen and W.-Y. Ma**: *Web Object Retrieval*, WWW 2007

- **H. M. Peetz and M. de Rijke**: *Cognitive Temporal Document Priors*, ECIR 2013

- **J. M. Ponte and B. Croft**: *A Language Modeling Approach to Information Retrieval*, SIGIR 1998

- **C. Zhai and J. Lafferty**: *Model-based Feedback in the Language Modeling Approach for Information Retrieval*, CIKM 2001

III.5 Latent Topic Models

1. Latent Semantic Indexing
2. Probabilistic Latent Semantic Indexing
3. Latent Dirichlet Allocation

Based on MRS Chapter 18 and [Blei ‘12]
Latent Topic Models

• Retrieval models seen so far (e.g., TF*IDF, LMs) do not handle 
  synonymy (e.g., car and automobile), polysemy (e.g., java), etc.

• Word co-occurrence can help us, e.g.:
  • car and automobile both occur together with garage, exhaust, fuel, ...
  • java occurs together with class and method but also with grind and coffee

• Latent topic models assume that documents are composed from 
  a small number \( k \) of latent (i.e., hidden, unknown) topics
  • Latent Semantic Indexing (LSI) [Deerwester et al. ‘90]
  • Probabilistic Latent Semantic Indexing (pLSI) [Hofmann ‘99]
  • Latent Dirichlet Allocation (LDA) [Blei et al. ‘03]
1. Latent Semantic Indexing (LSI)

- **Idea**: Apply SVD to $m$-by-$n$ term-document matrix $A$

- $U_k, V_k^T, \Sigma_k$ contain the first $k$ singular vectors and values
- $U_k$ maps terms to topics
- $V_k$ maps documents to topics
Operations in Latent Topic Space

- We can **map a query** \( q \) from \( m \)-dimensional term space into the \( k \)-dimensional topic space by \( q \rightarrow U_k^T q = q' \)

- **Ranking of documents** can then be determined by comparing \( q' \) against the columns of \( V_k^T \) using dot product or cosine similarity

- We can **fold in a new document** from \( m \)-dimensional term space by mapping it to \( k \)-dimensional topic space as \( d \rightarrow U_k^T d = d' \) and appending it as a new column to \( V_k^T \) (with quality deteriorating over time)
LSI (Example)

\[ A = \begin{pmatrix} 
0.5774 & 0.0000 & 0.0000 & 0.4082 & 0.0000 \\
0.5774 & 0.0000 & 1.0000 & 0.4082 & 0.7071 \\
0.5774 & 0.0000 & 0.0000 & 0.4082 & 0.0000 \\
0.0000 & 0.0000 & 0.0000 & 0.4082 & 0.0000 \\
0.0000 & 1.0000 & 0.0000 & 0.4082 & 0.7071 \\
0.0000 & 0.0000 & 0.0000 & 0.4082 & 0.0000 
\end{pmatrix} \]
LSI (Example)

$$A = \begin{pmatrix} 0.2670 & -0.2567 & 0.5308 & -0.2847 \\ 0.7479 & -0.3981 & -0.5249 & 0.0816 \\ 0.2670 & -0.2567 & 0.5308 & -0.2847 \\ 0.1182 & -0.0127 & 0.2774 & 0.6394 \\ 0.5198 & 0.8423 & 0.0838 & -0.1158 \\ 0.1182 & -0.0127 & 0.2774 & 0.6394 \end{pmatrix}$$

$$U \begin{pmatrix} 1.6950 & 0.0000 & 0.0000 & 0.0000 \\ 0.0000 & 1.1158 & 0.0000 & 0.0000 \\ 0.0000 & 0.0000 & 0.8403 & 0.0000 \\ 0.0000 & 0.0000 & 0.0000 & 0.4195 \end{pmatrix} \Sigma \begin{pmatrix} 0.4366 & 0.3067 & 0.4412 & 0.4909 & 0.5288 \\ -0.4717 & 0.7549 & -0.3568 & -0.0346 & 0.2815 \\ 0.3688 & 0.0998 & -0.6247 & 0.5711 & -0.3712 \\ -0.6715 & -0.2760 & 0.1945 & 0.6571 & -0.0577 \end{pmatrix} V^T$$
LSI (Example)

\[ A_3 = \begin{pmatrix}
0.4971 & -0.0330 & 0.0232 & 0.4867 & -0.0069 \\
0.6003 & 0.0094 & 0.9933 & 0.3858 & 0.7091 \\
0.4971 & -0.0330 & 0.0232 & 0.4867 & -0.0069 \\
0.1801 & 0.0740 & -0.0522 & 0.2320 & 0.0155 \\
-0.0326 & 0.9866 & 0.0094 & 0.4402 & 0.7043 \\
0.1801 & 0.0740 & -0.0522 & 0.2320 & 0.0155
\end{pmatrix} = U_3 \Sigma_3 V_3^T \]
LSI (Example)

• Query: \textit{baking bread}

  • \( q = (1 \ 0 \ 1 \ 0 \ 0 \ 0)^T \)
  
  • \( q' = U_3^T q = (0.5340 \ -0.5134 \ 1.0616)^T \)

• Dot-product similarity in topic space

  • \( \text{sim}(q, d_1) \approx 0.86 / \text{sim}(q, d_2) \approx -0.12 / \text{sim}(q, d_3) \approx -0.24 \)

• Adding \( d_6 = \text{“algorithmic recipes for the computation of pie”} \)

  • \( d = (0 \ 0.07071 \ 0 \ 0 \ 0 \ 0.07071)^T \)
  
  • \( d' = U_3^T d = (0.5 \ -0.28 \ -0.15)^T \)
  
  • \( d' \) becomes a new column of \( V_k^T \)
Issues with LSI

• **Parameter tuning**
  - How to select proper number of latent topics $k$?

• **Memory consumption**
  - Term-by-document matrix $A$ is usually sparse
  - SVD factors $U$ and $V$ are almost never sparse

• **Computational cost**
  - SVD still expensive to compute when $m$ and $n$ at the order of millions

• **Retrieval effectiveness**
  - LSI achieved only mediocre performance on TREC datasets with good gains for some queries but losses for others
2. Probabilistic Latent Semantic Indexing (pLSI)

- **Idea**: Model documents as (probabilistic) mixtures of topics
- Each topic generates terms with topic-specific probabilities
- Assume **conditional independence** of word $w$ and document $d$ given topic $t$:

\[
P[w, d, t] = P[w, d|t]P[t] = P[w|t]P[d|t]P[t]
\]

\[
P[w, d] = \sum_t P[w|t]P[d|t]P[t]
\]

- **Generative model**

\[
P[w|d] = \sum_t P[w|t] P[t|d]
\]
pLSI Generative Model

\[ P[w|d] = \sum_t P[w|t] P[t|d] \]
Computing pLSI

- **Parameters** $P[t|d]$ and $P[w|t]$ can be determined using the iterative method **Expectation Maximization** (EM)

- **Query** $q$ is folded in by estimating the topic distribution $P[t|q]$ that provides the best explanation of the query terms

- **Ranking of documents** can then be determined by comparing the topic distributions $P[t|q]$ and $P[t|d]$, e.g., using KL divergence
pLSI vs. LSI

\[ P[w, d] = \sum_t P[w|t] P[d|t] P[t] \]

- Differences to SVD:
  - probabilities \( P[w|t], P[d|t], \) and \( P[t] \) are non-negative and normalized
  - loss function is Kullback-Leibler divergence instead of squared loss
pLSI (Example)

- Topics (10 of 128) extracted from 12K Science Magazine articles

Source: Thomas Hofmann, Tutorial at ADFOCS 2004
3. Latent Dirichlet Allocation (LDA)

- Multiple-cause mixture model (MCMM)
- Documents contain **multiple topics**
- Topics are expressed by specific **word distributions**
- LDA provides a **generative model** for this
LDA Generative Model

- For each of the $D$ documents $d$
  - Choose document length $N$ (# word occurrences) $\sim$ Poisson($\lambda$)
  - Choose topic-probability distribution parameters $\beta \sim$ Dirichlet($\alpha$)
  - For each of the $N$ word occurrences in $d$ (at position $n$)
    - Choose one of $k$ topics $t_n \sim$ Multinomial($\beta$, $k$)
    - Choose one of $M$ words $w_n$ from per-topic distribution $\sim$ Multinomial($\theta$, $M$)
Comparison to Other Generative Models

- LDA
- pLSI
- Single-Cause Mixture of Unigrams
- Simple Unigram Model
Computing LDA

- Dirichlet(\(\alpha\)) probability density function

\[
f(\beta | \alpha) = \frac{\Gamma\left(\sum_{i=1}^{k} \alpha_i\right)}{\prod_{i=1}^{k} \Gamma(\alpha_i)} \prod_{i=1}^{k} \beta_i^{\alpha_i - 1}
\]

with \(\alpha_i \geq 0\), \(\beta_i \geq 0\) and \(\Sigma \beta_i = 1\)

- Probability of document \(d\) given \(\alpha\) and \(\theta\)

\[
P[d | \alpha, \theta] = \int f(\beta | \alpha) \left( \prod_{n=1}^{N} \sum_{t_n=1}^{k} \beta_{t_n} \theta_{t_n,w_n} \right) d\beta
\]

- Log-likelihood function (for corpus of \(D\) documents) is analytically intractable
Computing LDA (cont’d)

- Parameters $\alpha$ and $\theta$ can be estimated using Expectation Maximization (EM) with lower-bound distributions

  - **E-Step:** Determine optimal parameters $\gamma^*$ and $\phi^*$ of lower-bound distributions given $\alpha^{(i-1)}$ and $\theta^{(i-1)}$

  - **M-Step:** Given fixed lower-bound distributions determine parameters $\alpha^{(i)}$ and $\theta^{(i)}$ that maximize log-likelihood

- **Full details:** [Blei et al ’03]
LDA (Example)

- Topics from 5K scientific articles and 16K newswire articles

<table>
<thead>
<tr>
<th>“Arts”</th>
<th>“Budgets”</th>
<th>“Children”</th>
<th>“Education”</th>
</tr>
</thead>
<tbody>
<tr>
<td>NEW</td>
<td>MILLION</td>
<td>CHILDREN</td>
<td>SCHOOL</td>
</tr>
<tr>
<td>FILM</td>
<td>TAX</td>
<td>WOMEN</td>
<td>STUDENTS</td>
</tr>
<tr>
<td>SHOW</td>
<td>PROGRAM</td>
<td>PEOPLE</td>
<td>SCHOOLS</td>
</tr>
<tr>
<td>MUSIC</td>
<td>BUDGET</td>
<td>CHILD</td>
<td>EDUCATION</td>
</tr>
<tr>
<td>MOVIE</td>
<td>BILLION</td>
<td>YEARS</td>
<td>TEACHERS</td>
</tr>
<tr>
<td>PLAY</td>
<td>FEDERAL</td>
<td>FAMILIES</td>
<td>HIGH</td>
</tr>
<tr>
<td>MUSICAL</td>
<td>YEAR</td>
<td>WORK</td>
<td>PUBLIC</td>
</tr>
<tr>
<td>BEST</td>
<td>SPENDING</td>
<td>PARENTS</td>
<td>TEACHER</td>
</tr>
<tr>
<td>ACTOR</td>
<td>NEW</td>
<td>SAYS</td>
<td>BENNETT</td>
</tr>
<tr>
<td>FIRST</td>
<td>STATE</td>
<td>FAMILY</td>
<td>MANAGET</td>
</tr>
<tr>
<td>YORK</td>
<td>PLAN</td>
<td>WELFARE</td>
<td>NAMPHY</td>
</tr>
<tr>
<td>OPERA</td>
<td>MONEY</td>
<td>MEN</td>
<td>STATE</td>
</tr>
<tr>
<td>THEATER</td>
<td>PROGRAMS</td>
<td>PERCENT</td>
<td>PRESIDENT</td>
</tr>
<tr>
<td>ACTRESS</td>
<td>GOVERNMENT</td>
<td>CARE</td>
<td>ELEMENTARY</td>
</tr>
<tr>
<td>LOVE</td>
<td>CONGRESS</td>
<td>LIFE</td>
<td>HAITI</td>
</tr>
</tbody>
</table>

The William Randolph Hearst Foundation will give $1.25 million to Lincoln Center, Metropolitan Opera Co., New York Philharmonic and Juilliard School. “Our board felt that we had a real opportunity to make a mark on the future of the performing arts with these grants an act every bit as important as our traditional areas of support in health, medical research, education and the social services,” Hearst Foundation President Randolph A. Hearst said Monday in announcing the grants. Lincoln Center’s share will be $200,000 for its new building, which will house young artists and provide new public facilities. The Metropolitan Opera Co. and New York Philharmonic will receive $400,000 each. The Juilliard School, where music and the performing arts are taught, will get $250,000. The Hearst Foundation, a leading supporter of the Lincoln Center Consolidated Corporate Fund, will make its usual annual $100,000 donation, too.

Source: [Blei et al. ’03]
Summary of III.5

• **Latent topic models**
  consider word co-occurrence and implicitly handle synonymy etc.

• **Latent Semantic Indexing (LSI)**
  applies SVD to term-document matrix $A$

• **Probabilistic Latent Semantic Indexing (pLSI)**
  uses a non-negative probabilistic decomposition of $A$

• **Latent Dirichlet Allocation (LDA)**
  uses a probabilistic generative model
Additional Literature for III.5


- **T. Hofmann**: *Probabilistic Latent Semantic Indexing*, SIGIR 1999