Recap

- Probabilistic models:
  - Naive Bayes Text Classification
  - Introduction to Text Classification
  - Probabilistic Language Models
  - Naive Bayes text categorization

Today

- The Language Model Approach to IR
  - Basic query generation model
  - Alternative models

Standard Probabilistic IR

Information need → query → \( P(R|Q,d) \) → matching → document collection

IR based on Language Model (LM)

Information need → query → \( P(Q|M) \) → document collection

- A common search heuristic is to use words that you expect to find in matching documents as your query - why, I saw Sergey Brin advocating that strategy on late night TV one night in my hotel room, so it must be good!
- The LM approach directly exploits that idea!

Formal Language (Model)

- Traditional generative model: generates strings
- Finite state machines or regular grammars, etc.
- Example:

  I wish
  I wish I wish
  I wish I wish I wish
  I wish I wish I wish I wish
  ...

  *wish I wish
Stochastic Language Models

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

Model M

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

\[ P(s | M) = 0.00000008 \]

Stochastic Language Models

- Model probability of generating any string

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>the</td>
<td>class</td>
<td>pleaseth</td>
</tr>
<tr>
<td>0.2</td>
<td>0.0001</td>
<td>0.0001</td>
</tr>
<tr>
<td>0.01</td>
<td>0.0001</td>
<td>0.02</td>
</tr>
<tr>
<td>0.01</td>
<td>0.0001</td>
<td>0.01</td>
</tr>
<tr>
<td>0.0001</td>
<td>maiden</td>
<td>woman</td>
</tr>
</tbody>
</table>

\[ P(s(M2)) > P(s(M1)) \]

Stochastic Language Models

- A statistical model for generating text
  - Probability distribution over strings in a given language

Unigram and higher-order models

\[ P(\cdot | \cdot | \cdot | \cdot | \cdot | \cdot | \cdot | \cdot | \cdot | \cdot | \cdot | \cdot | \cdot) \]

- Unigram Language Models
  \[ P(\cdot) P(\cdot | \cdot) P(\cdot | \cdot | \cdot) P(\cdot | \cdot | \cdot | \cdot) \]

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

- Other Language Models
  - Grammar-based models (PCFGs), etc.
    - Probably not the first thing to try in IR

Using Language Models in IR

- Treat each document as the basis for a model (e.g., unigram sufficient statistics)
- Rank document $d$ based on $P(d | q)$
- $P(d | q) = P(q | d) \times P(d) / P(q)$
  - $P(q)$ 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
  - $P(q | d)$ is the probability of $q$ given $d$’s model
- Very general formal approach

The fundamental problem of LMs

- Usually we don’t know the model $M$
  - But have a sample of text representative of that model

\[ P(\cdot | \cdot | \cdot | M(\cdot | \cdot | \cdot | \cdot | \cdot | \cdot | \cdot | \cdot | \cdot | \cdot | \cdot | \cdot)) \]

- Estimate a language model from a sample
- Then compute the observation probability
Language Models for IR

- Language Modeling Approaches
  - Attempt to model query generation process
  - Documents are ranked by the probability that a query would be observed as a random sample from the respective document model
    - Multinomial approach
      \[ P(Q|M_D) = \prod_{w} P(w|M_D)^{p(w)} \]

Retrieval based on probabilistic LM

- Treat the generation of queries as a random process.
- Approach
  - Infer a language model for each document.
  - Estimate the probability of generating the query according to each of these models.
  - Rank the documents according to these probabilities.
  - Usually a unigram estimate of words is used
    - Some work on bigrams, paralleling van Rijsbergen

Query generation probability (1)

- Ranking formula
  \[ p(Q, d) = p(d) p(Q \mid d) \]
  \[ p(d) = p(Q \mid M_d) \]
  \[ p(Q \mid M_d) = \prod_{t} \hat{p}_c(t \mid M_d) \]
  \[ \hat{p}_c(t \mid M_d) = \frac{tf(t, d)}{dl_d} \]

- Unigram assumption: Given a particular language model, the query terms occur independently.

Insufficient data

- Zero probability
  - \( p(t \mid M_d) = 0 \)
  - May not wish to assign a probability of zero to a document that is missing one or more of the query terms [gives conjunction semantics]
- General approach
  - A non-occurring term is possible, but no more likely than would be expected by chance in the collection.
  - If \( tf(t, d) = 0 \)
    \[ p(t \mid M_d) = \frac{tf(t, d)}{cs} \]

Insufficient data

- Zero probabilities spell disaster
  - We need to smooth probabilities
    - Discount nonzero probabilities
    - Give some probability mass to unseen things
- There’s a wide space of approaches to smoothing probability distributions to deal with this problem, such as adding 1, \( \frac{1}{2} \) or \( \epsilon \) to counts, Dirichlet priors, discounting, and interpolation
  - [See FSNLP ch. 6 or CS224N if you want more]
- A simple idea that works well in practice is to use a mixture between the document multinomial and the collection multinomial distribution
Mixture model

- $P(w|d) = \lambda P_{\text{mle}}(w|M_d) + (1-\lambda)P_{\text{mle}}(w|M_c)$
- Mixes the probability from the document with the general collection frequency of the word.
- Correctly setting $\lambda$ is very important
- A high value of lambda makes the search "conjunctive-like" – suitable for short queries
- A low value is more suitable for long queries
- Can tune $\lambda$ to optimize performance
- Perhaps make it dependent on document size (cf. Dirichlet prior or Witten-Bell smoothing)

Basic mixture model summary

- General formulation of the LM for IR
  $$p(Q,d) = p(d) \prod_{i \in Q} (1-\lambda) p(t) + \lambda p(t|\{M_d\})$$

- The user has a document in mind, and generates the query from this document.
- The equation represents the probability that the document that the user had in mind was in fact this one.

Example

- Document collection (2 documents)
  - $d_1$: Xerox reports a profit but revenue is down
  - $d_2$: Lucent narrows quarter loss but revenue decreases further
- Model: MLE unigram from documents; $\lambda = 1/2$
- Query: revenue down
  - $P(Q|d_1) = [(1/8 + 2/16)/2] \times [(1/8 + 1/16)/2]$
    $$= 1/8 \times 3/32 = 3/256$$
  - $P(Q|d_2) = [(1/8 + 2/16)/2] \times [0 + 1/16]/2$
    $$= 1/8 \times 1/32 = 1/256$$
- Ranking: $d_1 > d_2$

Ponte and Croft Experiments

- Data
  - TREC topics 202-250 on TREC disks 2 and 3
  - Natural language queries consisting of one sentence each
  - TREC topics 51-100 on TREC disk 3 using the concept fields
    - Lists of good terms

Precision/recall results 202-250

<table>
<thead>
<tr>
<th>Topic</th>
<th>User</th>
<th>Precision</th>
<th>Recall</th>
<th>IR</th>
<th>IRW</th>
<th>IRB</th>
<th>IRW</th>
<th>IRB</th>
</tr>
</thead>
<tbody>
<tr>
<td>088</td>
<td>0.50</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>089</td>
<td>0.50</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>090</td>
<td>0.50</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>091</td>
<td>0.50</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Precision/recall results 51-100

<table>
<thead>
<tr>
<th>Topic</th>
<th>User</th>
<th>Precision</th>
<th>Recall</th>
<th>IR</th>
<th>IRW</th>
<th>IRB</th>
<th>IRW</th>
<th>IRB</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>0.50</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>101</td>
<td>0.50</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>102</td>
<td>0.50</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>103</td>
<td>0.50</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
LM vs. Prob. Model for IR

- The main difference is whether “Relevance” figures explicitly in the model or not
  - LM approach attempts to do away with modeling relevance
  - LM approach assumes that documents and expressions of information problems are of the same type
  - Computationally tractable, intuitively appealing

- Problems of basic LM approach
  - Assumption of equivalence between document and information problem representation is unrealistic
  - Very simple models of language
  - Relevance feedback is difficult to integrate, as are user preferences, and other general issues of relevance
  - Can’t easily accommodate phrases, passages, Boolean operators
- Current extensions focus on putting relevance back into the model, etc.

Extension: 3-level model

- 3-level model
  - Whole collection model ($M_d$)
  - Specific-topic model; relevant-documents model ($M_f$)
  - Individual-document model ($M_i$)
- Relevance hypothesis
  - A request(query; topic) is generated from a specific-topic model ($M_f, M_r$).
  - If a document is relevant to the topic, the same model will apply to the document.
  - It will replace part of the individual-document model in explaining the document.
  - The probability of relevance of a document
  - The probability that this model explains part of the document
  - The probability that the $\{M_d, M_f, M_i\}$ combination is better than the $\{M_d, M_i\}$ combination

Alternative Models of Text Generation

- Searcher
  - $P(M_{\text{Searcher}})$
- Query Model
  - $P(\text{Query}|M)$
- Doc Model
  - $P(\text{Doc}|M)$
- Writer
  - $P(M_{\text{Writer}})$
- Doc
  - $P(Doc|M)$
- Is this the same model?

Retrieval Using Language Models

- Query
  - $P(w|\text{Query})$
- Doc
  - $P(w|\text{Doc})$

Retrieval: Query likelihood (1), Document likelihood (2), Model comparison (3)
Query Likelihood

- $P(Q|D_m)$
- Major issue is estimating document model
- i.e. smoothing techniques instead of tf.idf weights
- Good retrieval results
  - e.g. UMass, BBN, Twente, CMU
- Problems dealing with relevance feedback, query expansion, structured queries

Document Likelihood

- Rank by likelihood ratio $P(D|R)/P(D|NR)$
- $P(w|R)$ is estimated by $P(w|Q_m)$
- $P(w|NR)$ is estimated by collection probabilities $P(w)$
- Problems dealing with relevance feedback, query expansion
- Better results than query-likelihood or document-likelihood approaches
  - e.g. UMass at SIGIR 01
  - inconsistent with heterogeneous document collections

Model Comparison

- Estimate query and document models and compare
- Suitable measure is KL divergence $D(Q_m || D_m)$
- $D(Q_m || D_m) = \sum_{x \in \mathcal{X}} Q_m(x) \log \frac{Q_m(x)}{D_m(x)}$
- Equivalent to query-likelihood approach if simple empirical distribution used for query model
- More general risk minimization framework has been proposed
  - Zhai and Lafferty 2001
- Better results than query-likelihood or document-likelihood approaches

Two-stage smoothing: Another Reason for Smoothing

<table>
<thead>
<tr>
<th>Query = “the    algorithms     for      data       mining”</th>
<th>d1:</th>
<th>d2:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0.04</td>
<td>0.001</td>
</tr>
<tr>
<td></td>
<td>0.02</td>
<td>0.001</td>
</tr>
</tbody>
</table>

- $p( \text{algorithm}|d1) = p( \text{algorithm}|d2)$
- $p( \text{data}|d1) = p( \text{data}|d2)$
- $p( \text{mining}|d1) < p( \text{mining}|d2)$
- But $p(q|d1) > p(q|d2)$!

We should make $p(\text{“the”})$ and $p(\text{“for”})$ less different for all docs.

Two-stage Smoothing

Stage-1
- Explain unseen words
- Dirichlet prior (Bayesian)

Stage-2
- Explain noise in query
- 2-component mixture

$P(w|d) = \frac{(1-\lambda) p(w|d) + \lambda p(w|U)}{d} + \lambda$
 Expansion-based vs. Model-based

Feedback as Model Interpolation

Translation model (Berger and Lafferty)

Language models: pro & con

Comparison With Vector Space

Comparison With Vector Space
**Resources**


[Several relevant newer papers at SIGIR 23–25, 2000–2002.]
