CSCI 5417
Information Retrieval Systems
Jim Martin

Lecture 10
9/22/2011

## Today 9/22

- Finish LM-based IR
  - Language models in general
  - Smoothing
  - LM for ad hoc retrieval performance
- Project brainstorming

## An Alternative to the VS Model

- Basic vector space model uses a geometric metaphor/framework for the ad hoc retrieval problem
  - One dimension for each word in the vocab
  - Weights are usually tf-idf based
- An alternative is to use a probabilistic approach
  - So we'll take a short detour into probabilistic language modeling



## In General

- When you propose a probabilistic approach to problems like this you need to specify three things
  1. Exactly what you want to the model to be
  2. How you will acquire the parameters of that model
  3. How you will use the model operationally

## Where we are

- In the LM approach to IR, we attempt to model the query generation process.
  - Think of a query as being generated from a model derived from a document (or documents)
- Then we rank documents by the probability that a query would be observed as a random sample from the respective document model.
- That is, we rank according to $P(q|d)$.
- Next: how do we compute $P(q|d)$?

5

## Stochastic Language Models

- Models *probability* of generating strings (each word in turn) in the language (commonly all strings over Σ). E.g., unigram model

Model M

| | |
|---|---|
| 0.2 | the |
| 0.1 | a |
| 0.01 | man |
| 0.01 | woman |
| 0.03 | said |
| 0.02 | likes |
| … | |

| the | man | likes | the | woman |
|---|---|---|---|---|
| 0.2 | 0.01 | 0.02 | 0.2 | 0.01 |

multiply

P(s | M) = 0.00000008

## Stochastic Language Models

- Model *probability* of generating any string (for example, a query)

| Model M1 | | Model M2 | |
|---|---|---|---|
| 0.2 | the | 0.2 | the |
| 0.01 | class | 0.0001 | class |
| 0.0001 | sayst | 0.03 | sayst |
| 0.0001 | pleaseth | 0.02 | pleaseth |
| 0.0001 | yon | 0.1 | yon |
| 0.0005 | maiden | 0.01 | maiden |
| 0.01 | woman | 0.0001 | woman |

| the | class | pleaseth | yon | maiden |
|---|---|---|---|---|
| 0.2 | 0.01 | 0.0001 | 0.0001 | 0.0005 |
| 0.2 | 0.0001 | 0.02 | 0.1 | 0.01 |

P(s|M2) > P(s|M1)



## How to compute $P(q|d)$

- This kind of conditional independence assumption is often called a Markov model

$$P(q|M_d) = P(\langle t_1, \ldots, t_{|q|}\rangle|M_d) = \prod_{1\le k\le|q|} P(t_k|M_d)$$

($|q|$: length ofr $q$; $t_k$ : the token occurring at position k in q)

- This is equivalent to:

$$P(q|M_d) = \prod_{\text{distinct term } t \text{ in } q} P(t|M_d)^{\text{tf}_{t,q}}$$

- $\text{tf}_{t,q}$: term frequency (# occurrences) of $t$ in $q$

8

## Unigram and higher-order models

P ( ● ○ ● ● )

- = P ( ●) P ( ○ | ●) P ( ● | ● ○) P ( ●| ● ○ ●)
- Unigram Language Models
  P ( ●) P ( ○) P ( ●) P ( ●)
- Bigram (generally, *n*-gram) Language Models
  P ( ●) P ( ○| ●) P ( ●| ○) P ( ●| ●)
- Other Language Models
  - Grammar-based models (PCFGs), etc.
    - Probably not the first thing to try in IR

Easy. Effective!



## Using Language Models for ad hoc Retrieval

- Each document is treated as (the basis for) a language model.
- Given a query *q*
- Rank documents based on *P*(*d*|*q*) via

$$P(d|q) = \frac{P(q|d)P(d)}{P(q)}$$

- *P*(*q*) is the same for all documents, so ignore
- *P*(*d*) is the prior – often treated as the same for all *d*
  - But we can give a higher prior to "high-quality" documents
    - PageRank, click through, social tags, etc.
- *P*(*q*|*d*) is the probability of *q* given *d*.
  - So to rank documents according to relevance to *q*, rank according to *P*(*q*|*d*)

10

## How to compute $P(q|d)$

- We will make the same conditional independence assumption as for Naive Bayes.

$$P(q|M_d) = P(\langle t_1, \ldots, t_{|q|}\rangle|M_d) = \prod_{1\leq k\leq |q|} P(t_k|M_d)$$

($|q|$: length ofr $q$; $t_k$ : the token occurring at position k in q)

- This is equivalent to:

$$P(q|M_d) = \prod_{\text{distinct term } t \text{ in } q} P(t|M_d)^{\text{tf}_{t,q}}$$

- $\text{tf}_{t,q}$: term frequency (# occurrences) of $t$ in $q$

11

## Parameter estimation

- Where do the parameters $P(t|M_d)$. come from?
- Start with simple counts (maximum likelihood estimates)

$$\hat{P}(t|M_d) = \frac{\text{tf}_{t,d}}{|d|}$$

$$P(q|M_d) = \prod P(t|M_d)$$

$|d|$: length of document $d$;

$\text{tf}_{t,d}$ : # occurrences of term $t$ in document $d$

12

## Problem: Zero counts

- A single term $t$ with $P(t|M_d) = 0$ will make this

$$P(q|M_d) = \prod P(t|M_d)$$

  zero.
- This would give a *single term* the power to eliminate an otherwise relevant document.
- For example, for query
  - "Michael Jackson top hits"

  a document about "Jackson top songs" (but not using the word "hits") would have $P(t|M_d) = 0$. – That's bad.



## Smoothing

- Key intuition: A non-occurring term is possible (even though it didn't occur). That is it's probability shouldn't be zero
- If it isn't zero what should it be? Remember that we're developing LMs for each document in a collection.

$$T = \sum_t \mathrm{cf}_t$$

- but no more likely than would be expected by chance in the collection.

$$\hat{P}(t|M_d) = \frac{\mathrm{tf}_{t,d}}{|d|}$$

- Notation: $M_c$: the collection mod of occurrence $\hat{P}(t|M_c)$ e collection; : the total number of tokens in the collection.

## Smoothing

- Fall back on using the probability of that term in the collection as whole.
- Notation: $M_c$: the collection model; $\text{cf}_t$: the number of occurrences of $t$ in the collection; $T = \sum_t \text{cf}_t$ : the total number of tokens in the collection.

$$\hat{P}(t|M_d) = \frac{\text{tf}_{t,d}}{|d|}$$

- We will use $\hat{P}(t|M_c)$ to "smooth" $P(t|d)$ away from zero.



## Mixture model

- $P(t|d) = \lambda P(t|M_d) + (1 - \lambda)P(t|M_c)$
  - Mixes the probability from the document with the general collection frequency of the word
- If a term in query occurs in a document we combine the two scores with differing weights
- If a term doesn't occur then its just the second factor
  - The P of the term in the collection discounted by $(1 - \lambda)$

## Smoothing

- High value of $\lambda$: "conjunctive-like" search – tends to retrieve documents containing all query words.
- Low value of $\lambda$: more disjunctive, best for long queries
- Correctly setting $\lambda$ is very important for good performance.



## Mixture model: Summary

$$P(q|d) \propto \prod_{1 \leq k \leq |q|} (\lambda P(t_k|M_d) + (1-\lambda)P(t_k|M_c))$$

- What we model: 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

- Collection: $d_1$ and $d_2$
- $d_1$ : Jackson was one of the most talented entertainers of all time
- $d_2$: Michael Jackson anointed himself King of Pop
- Query $q$: Michael Jackson
- Use mixture model with $\lambda = 1/2$
- $P(q|d_1) = [(0/11 + 1/18)/2] \cdot [(1/11 + 2/18)/2] \approx 0.003$
- $P(q|d_2) = [(1/7 + 1/18)/2] \cdot [(1/7 + 2/18)/2] \approx 0.013$
- Ranking: $d_2 > d_1$

19

## Vector space (tf-idf) vs. LM

| | precision | | | significant? |
|---|---|---|---|---|
| Rec. | tf-idf | LM | %chg | |
| 0.0 | 0.7439 | 0.7590 | +2.0 | |
| 0.1 | 0.4521 | 0.4910 | +8.6 | |
| 0.2 | 0.3514 | 0.4045 | +15.1 | * |
| 0.4 | 0.2093 | 0.2572 | +22.9 | * |
| 0.6 | 0.1024 | 0.1405 | +37.1 | * |
| 0.8 | 0.0160 | 0.0432 | +169.6 | * |
| 1.0 | 0.0028 | 0.0050 | +76.9 | |
| 11-point average | 0.1868 | 0.2233 | +19.6 | * |

The language modeling approach always does better in these experiments . . . . . But the approach shows significant gains is at higher levels of recall.

20

## LMs vs. vector space model (1)

- LMs have some things in common with vector space models.
- Term frequency is clearly part of the model
  - But it not log-scaled as in VS
- Mixing document and collection frequencies has an effect similar to idf.
  - Terms rare in the general collection, but common in some documents will have a greater influence on the ranking.

21

## Indri

- The INDRI search engine is partially based on this kind of language model notion. Along with some bayesian inference.
- INDRI was one of the search systems used in IBM's Watson (Jeopardy) system
  - Along with Lucene

## Next time

- Quiz