Language Models

CE-324: Modern Information Retrieval

Sharif University of Technology

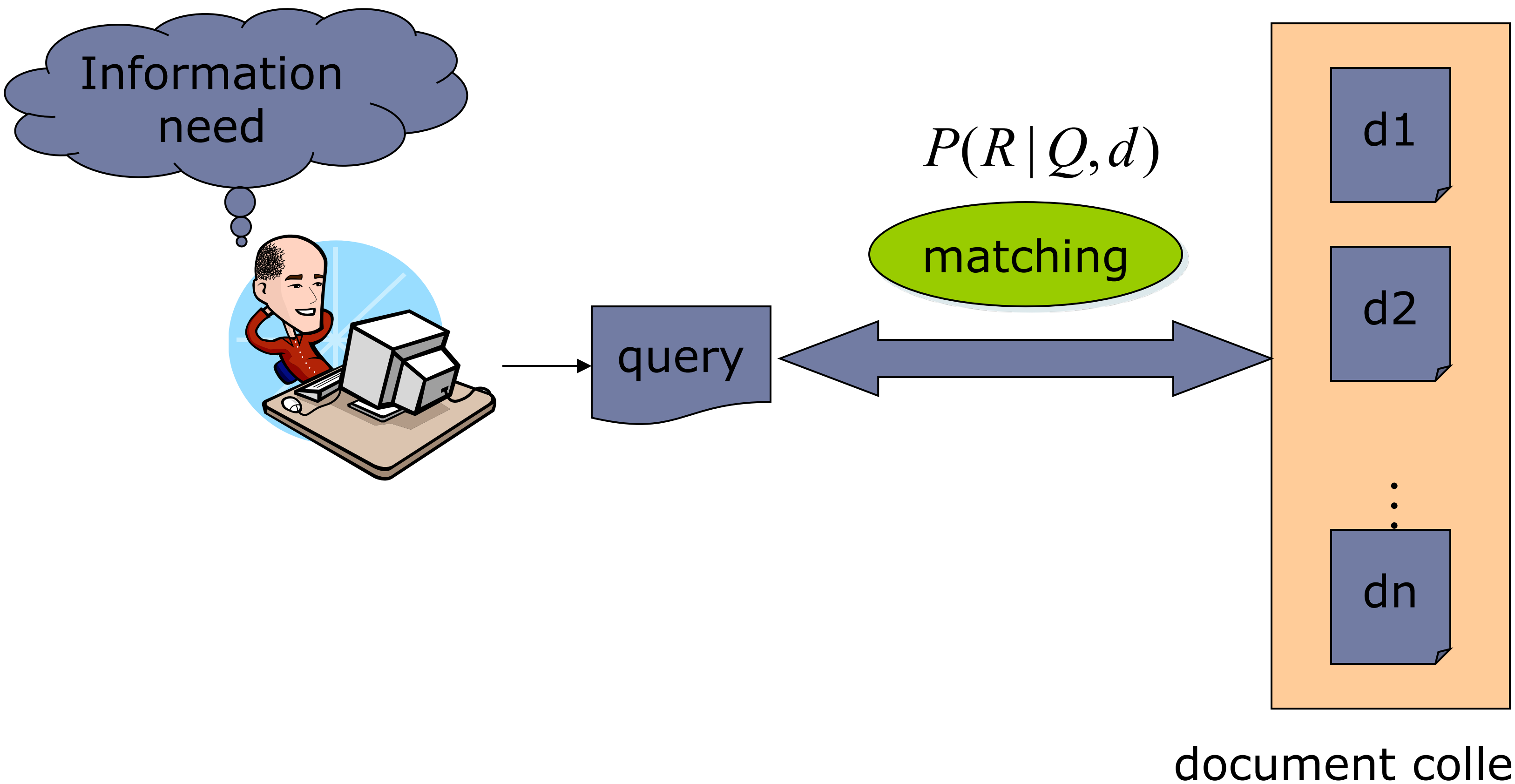
M. Soleymani

Fall 2018

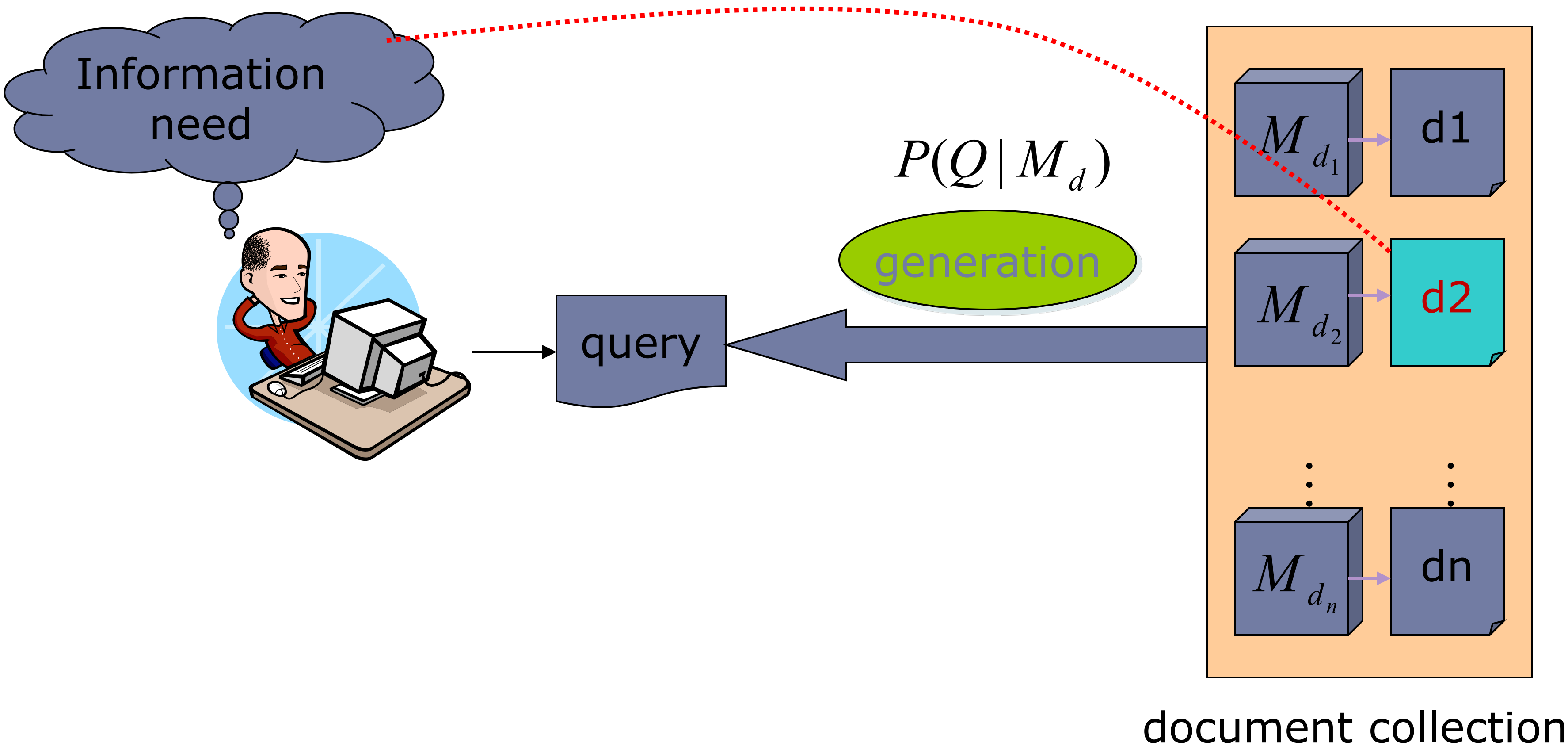
Most slides have been adapted from: Profs. Manning, Nayak & Raghavan (CS-276, Stanford)

Standard probabilistic IR: PRP

Ranking based on PRP



IR based on Language Model (LM)

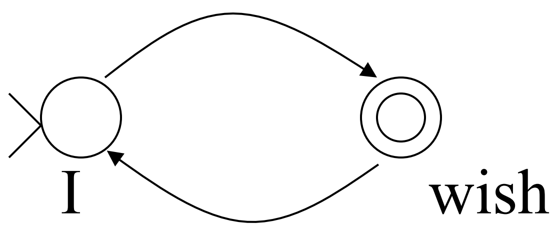


Language models in IR

- Often, users have a reasonable idea of terms that are likely to occur in docs of interest
- They choose query terms that distinguish these docs from others in the collection
- LM approach assumes that docs and query are objects of the same type
  - Thus, assesses their match by importing the methods of language modeling

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
…

Stochastic language models

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

$$\sum_{s \in \Sigma^*} p(s) = 1$$

- Unigram model:
  - probabilistic finite automaton consisting of just a single node
  - with a single probability distribution over producing different terms $\sum_{t \in V} p(t) = 1$
  - also requires a probability of stopping in the finishing state

Example

Model M

the	0.2
a	0.1
information	0.01
retrieval	0.01
data	0.02
compute	0.03
	…

the	information	retrieval
0.2	0.01	0.01

multiply

$P(s \mid M) \propto 0.00002$

Stochastic language models

- Model *probability* of generating any string

Model M1

the	0.2
a	0.1
data	0.02
information	0.01
retrieval	0.01
computing	0.005
system	0.004
…	…

Model M2

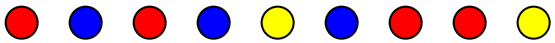
the	0.15
a	0.08
management	0.05
information	0.02
database	0.02
system	0.015
mining	0.002
…	…

information	system
0.01	0.004
0.02	0.015

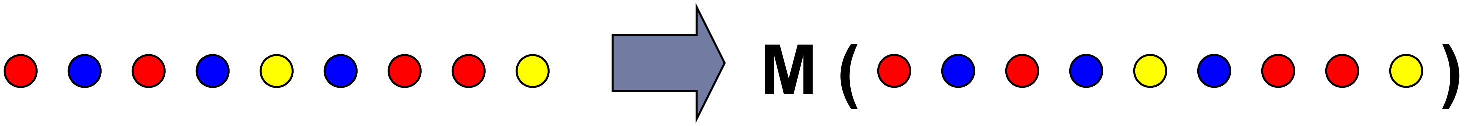
$$P(s|M_2) > P(s|M_1)$$

The fundamental problem of LMs

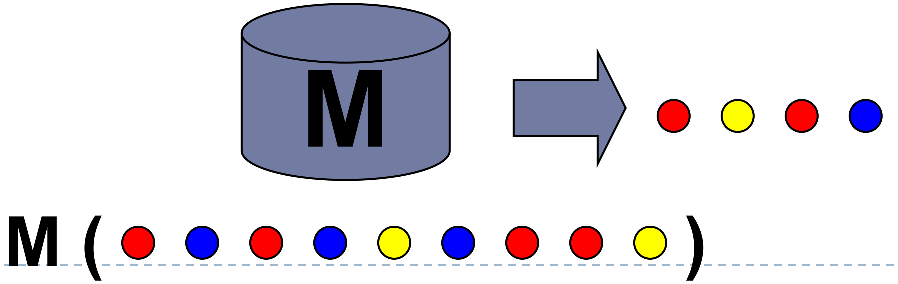
- Usually we don't know the model ***M***
  - But have a sample of text representative of that model



- Estimate a language model from a sample doc

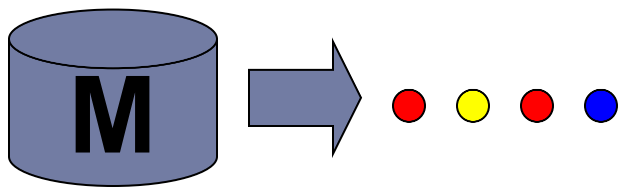


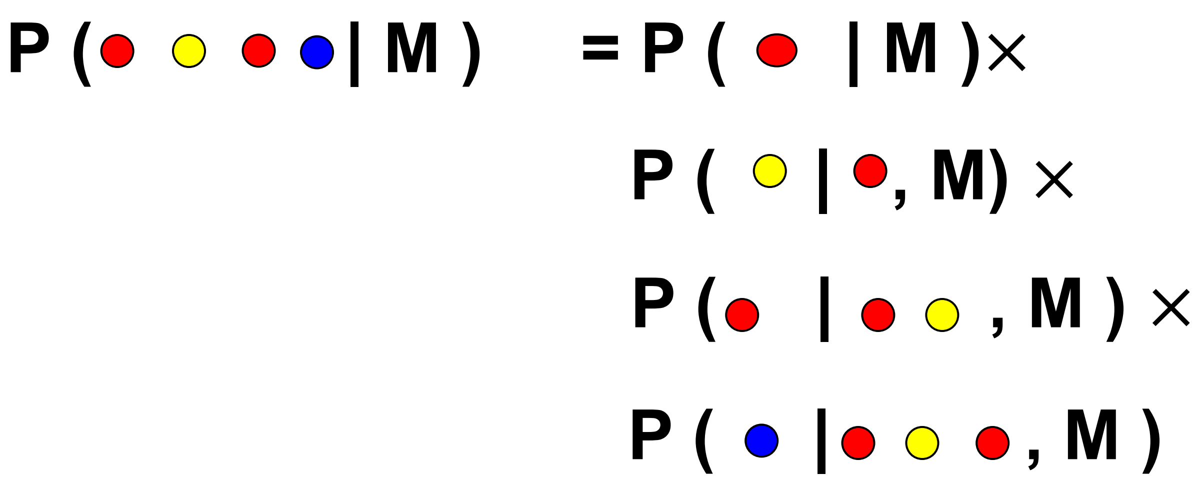
- Then compute the observation probability



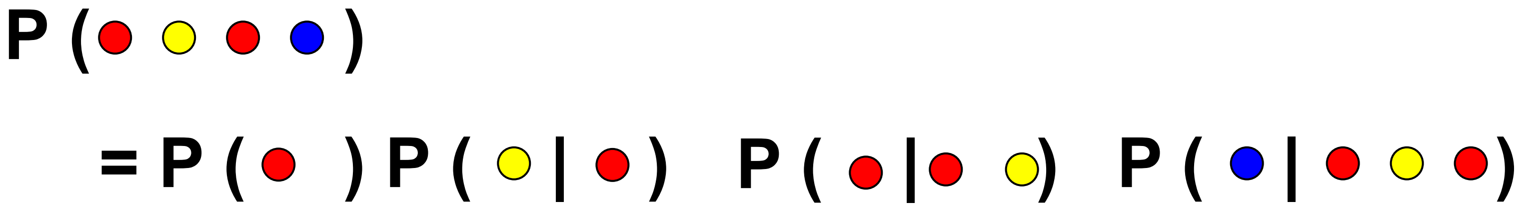
Stochastic language models

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





Unigram and higher-order models



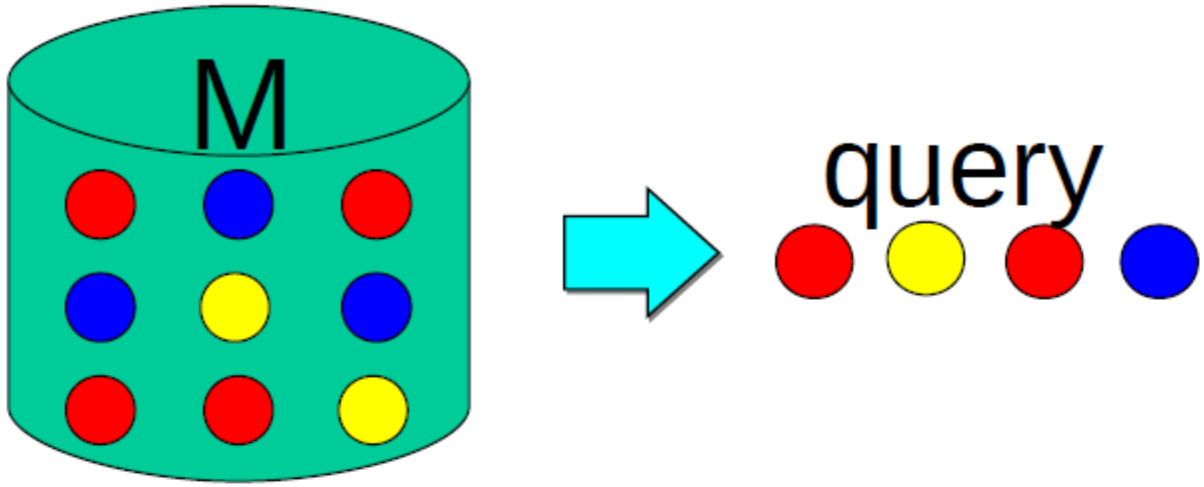
- **Unigram** Language Models

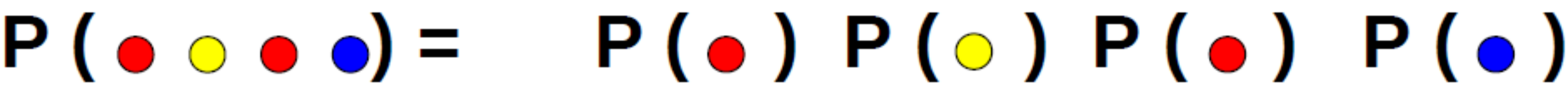
Easy.
Effective!

- **Bigram** (generally, *n*-gram) Language Models

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

Unigram model





Probabilistic language models in IR

- Treat each doc as the basis for a model
  - e.g., unigram sufficient statistics

- Rank doc d based on $P(d|q)$
  - $P(d|q) = P(q|d) \times P(d) / P(q)$
    - $P(q)$ is the same for all docs, 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

Query likelihood language model

$$p(d|q) = \frac{p(q|d) \times p(d)}{p(q)}$$

$$\approx \frac{p(q|M_d) \times p(d)}{p(q)}$$

- Ranking formula

$$p(d)p(q \mid M_d)$$

Language models for IR

- Language Modeling Approaches
  - Attempt to **model query generation process**
  - Docs are ranked by **the probability that a query would be observed as a random sample from the doc model**
    - Multinomial approach

$$P(q|M_d) = K_q \prod_{t \in V} P(t|M_d)^{\mathrm{tf}_{t,q}}$$

$$K_q = \frac{L_q!}{tf_{1,q}! \times \cdots \times tf_{M,q}!}$$

Retrieval based on probabilistic LM

- Generation of queries as a random process

- Approach
  - Infer a language model for each doc.
    - Usually a unigram estimate of words is used
      - Some work on bigrams

  - Estimate the probability of generating the query according to each of these models.

  - Rank the docs according to these probabilities.

Query generation probability

- The probability of producing the query given the language model of doc d using MLE is:

$$\hat{p}(t|M_d) = \frac{tf_{t,d}}{L_d}$$

$$\hat{p}(q|M_d) \propto \prod_{t \in q} \hat{p}(t|M_d)^{tf_{t,q}}$$

Unigram assumption:
Given a particular language moc
the query terms occur independer

M_d : language model of document d

$tf_{t,d}$: raw tf of term t in document d

L_d: total number of tokens in document d

$tf_{t,q}$: raw tf of term t in query q

Insufficient data

- Zero probability
  - May not wish to assign a probability of zero to a doc missing one or more of the query terms [gives conjunction semantics]

$$\hat{p}(t|M_d) = 0$$

- Poor estimation: occurring words may also be badly estimated
  - in particular, the probability of words occurring for example once in the doc is normally overestimated

Insufficient data: solution

- Zero probabilities spell disaster
  - We need to smooth probabilities
    - Discount nonzero probabilities
    - Give some probability mass to unseen things

- Smoothing: discounts non-zero probabilities and gives some probability mass to unseen words

- Many approaches to smoothing probability distributions to deal with this problem
  - i.e., adding 1, 1/2 or α to counts, interpolation, and etc.

Collection statistics

- A non-occurring term is possible, but no more likely than would be expected by chance in the collection.

$$\text{If } tf_{t,d} = 0 \text{ then } \hat{p}(t|M_d) < \frac{cf_t}{T}$$

cf_t : raw count of term t in the collection

$cs = T$: raw collection size (total number of tokens in the collection)

$$\hat{p}(t|M_c) = \frac{cf_t}{T}$$

- Collection statistics …
  - Are integral parts of the language model (as we will see).
  - Are not used heuristically as in many other approaches.
    - However there's some wiggle room for empirically set parameters

Bayesian smoothing

$$\hat{p}(t|d) = \frac{tf_{t,d} + \alpha\hat{p}(t|Mc)}{L_d + \alpha}$$

- For a word present in the doc:
  - combines a discounted MLE and a fraction of the estimate of its prevalence in the whole collection
- For words not present in a doc:
  - is just a fraction of the estimate of the prevalence of the word in the whole collection.

Linear interpolation: Mixture model

- **Linear interpolation**: Mixes the probability from the doc with the general collection frequency of the word. $0 \leq \lambda \leq 1$
  - using a mixture between the doc multinomial and the collection multinomial distribution

$$\hat{p}(t|d) = \lambda\hat{p}(t|M_d) + (1 - \lambda)\hat{p}(t|Mc)$$

$$\hat{p}(t|d) = \lambda\frac{tf_{t,d}}{L_d} + (1 - \lambda)\frac{cf_t}{T}$$

- It works well in practice

Linear interpolation: Mixture model

- Correctly setting λ is very important
  - high value: "conjunctive-like" search– suitable for short queries
  - low value for long queries
  - Can tune λ to optimize performance
    - Perhaps make it dependent on doc size (cf. Dirichlet prior or Witten-Bell smoothing)

Basic mixture model: summary

- General formulation of the LM for IR

$$\hat{p}(q|d) = \prod_{t \in q} \lambda \hat{p}(t|M_d) + (1 - \lambda)\hat{p}(t|Mc)$$

general language model

individual-document model

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

Example

- Doc collection (2 docs)
  - d_1:"Xerox reports a profit but **revenue** is **down**"
  - d_2:"Lucent narrows quarter loss but **revenue** decreases further"

- Model: MLE unigram from docs; $\lambda = \frac{1}{2}$

- 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

<num>Number: 054

<dom>Domain: International Economics

<title>Topic: Satellite Launch Contracts

<desc>Description:

… </desc>

<con>Concept(s):

1. Contract, agreement
2. Launch vehicle, rocket, payload, satellite
3. Launch services, … </con>

Precision/recall results 202-250

	Precision			
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	*
0.3	0.2761	0.3342	+21.0	*
0.4	0.2093	0.2572	+22.9	*
0.5	0.1558	0.2061	+32.3	*
0.6	0.1024	0.1405	+37.1	*
0.7	0.0451	0.0760	+68.7	*
0.8	0.0160	0.0432	+169.6	*
0.9	0.0033	0.0063	+89.3	
1.0	0.0028	0.0050	+76.9	
Ave	0.1868	0.2233	+19.55	*

LM vs. probabilistic model for IR (PRP)

- Main difference: whether "Relevance" figures explicitly in the model or not
  - LM approach attempts to do away with modeling relevance

- LM approach assumes that docs and queries are of the same type

LM vs. probabilistic model for IR

- Problems of basic LM approach
  - Assumption of equivalence between doc and information problem representation is unrealistic
  - Very simple models of language
  - Relevance feedback is difficult to integrate
    - user preferences, and other general issues of relevance
  - Can't easily accommodate phrases, passages, Boolean operators

- Extensions focus on putting relevance back into the model, etc.

- It has shown the LM approach to be very effective in retrieval experiments, beating tf-idf and BM25 weights

Translation model (Berger and Lafferty)

- Basic LMs do not address issues of synonymy.
  - Or any deviation in expression of information need from language of docs

- A translation model: generate query words not in doc via "translation" to synonyms etc.
  - Or to do cross-language IR, or multimedia IR

$$P(q \mid M_d) = \prod_{t \in q} \sum_{v \in V} \underbrace{P(v \mid M_d)}_{\text{Basic LM}} \times \underbrace{T(t \mid v)}_{\text{Translation}}$$

  - Need to learn a translation model (using a dictionary or via statistical machine translation)

Language models: summary

- Novel way of looking at IR problem based on probabilistic language modeling
  - Conceptually simple and explanatory
  - Formal mathematical model
  - Natural use of collection statistics, not heuristics (almost…)
- Effective retrieval and can be improved to the extent that the following conditions can be met
  - accurate representations of the data
  - users have some sense of term distribution
    - we get more sophisticated with translation model

Comparison with vector space

- There's some relation to traditional tf.idf models:
  - (unscaled) term frequency is directly in model
  - probabilities do length normalization of term frequencies
  - 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

- Similar in some ways
  - Term weights based on their frequency
  - Terms often used as if they were independent
  - Inverse document/collection frequency used
  - Some form of length normalization useful

- Different in others
  - 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

Resources

IIR, Chapter 12.

The Lemur Toolkit for Language Modeling and Information Retrieval. [CMU/Umass LM and IR system in C(++)] http://www-2.cs.cmu.edu/~lemur/