Information Retrieval and Web Search

Cornelia Caragea

Computer Science
University of Illinois at Chicago

Credits for slides: Hofmann, Mobasher, Schutze

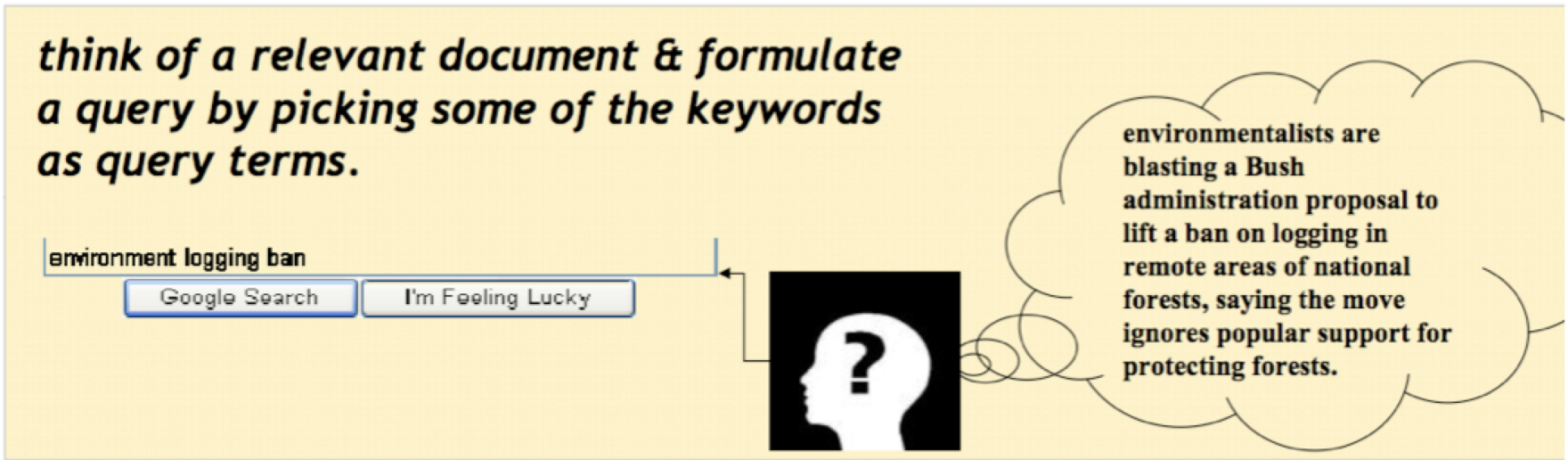
The Language Model Approach to IR

Required Reading

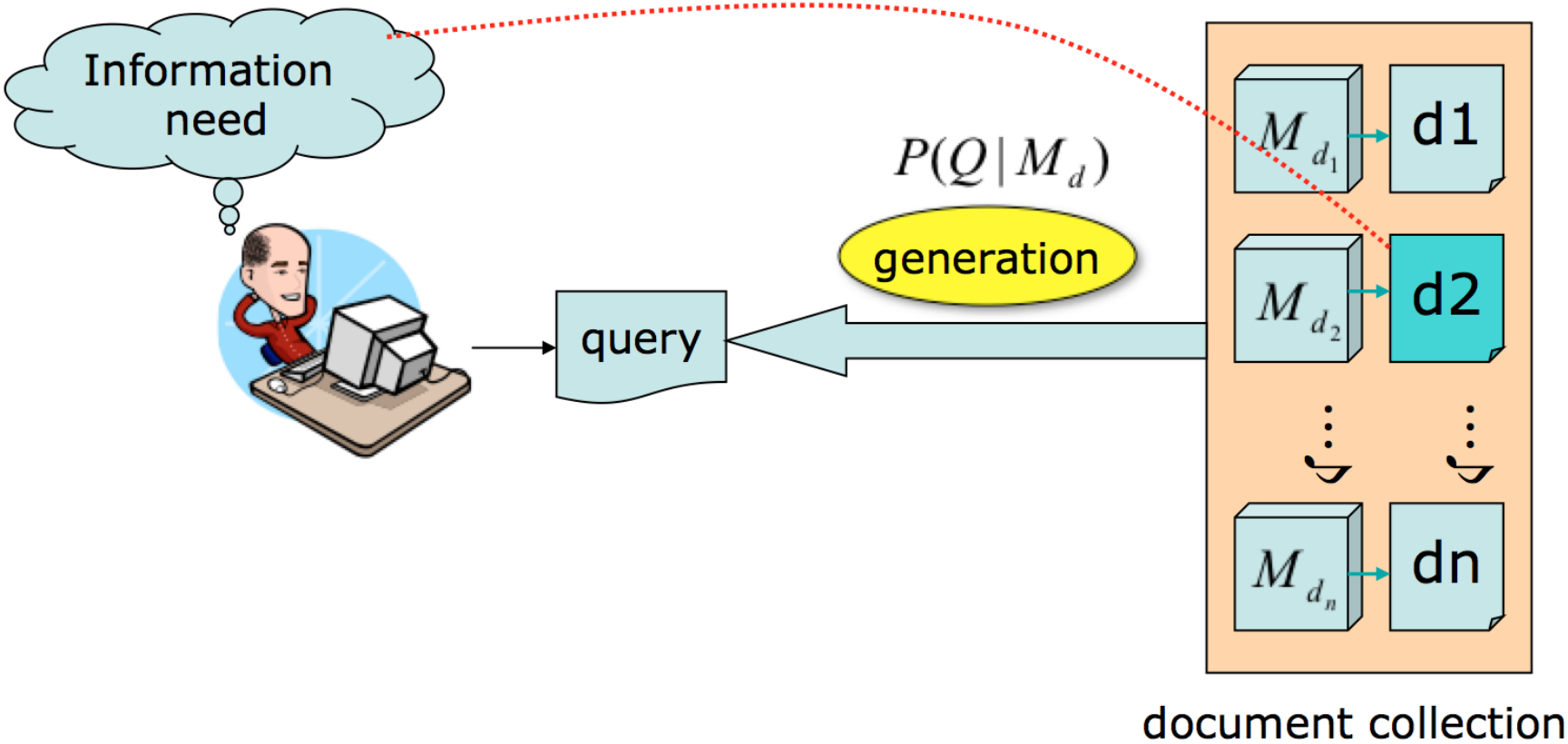
- "Information Retrieval" textbook
  - Chapter 12: Language models

The Query Generation Model

- How might a query look like that would ask for a specific document?
- A common search heuristic is to use words that you expect to find in matching document(s) as your query.
- The language model (LM) approach directly exploits that idea!
  - A document is a good match to a query if the document model is likely to generate the query, which will, in turn, happen if the document contains the query words often.



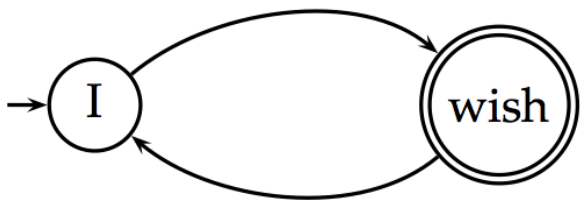
IR based on Language Model (LM)



What do we mean by a document model generating a query?

Formal Language (Model)

- A traditional generative model of a language: generates strings
  - Example: finite state machines.
- An example of a finite automaton:

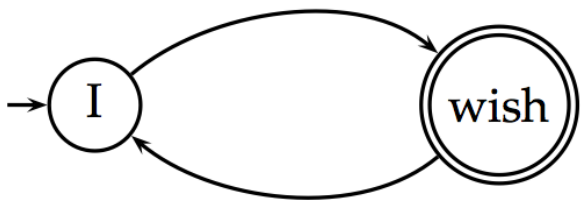


I wish
I wish I wish
I wish I wish I wish
I wish I wish I wish I wish
...

wish I wish ???

Formal Language (Model)

- A traditional generative model of a language: generates strings
  - Example: finite state machines.
- An example of a finite automaton:



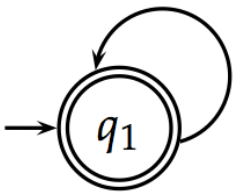
I wish
I wish I wish
I wish I wish I wish
I wish I wish I wish I wish
...

wish I wish ??? No!

If each node has a probability distribution over generating different terms, we have a language model.

Stochastic Language Model

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



A probabilistic finite automaton consisting of a single node with a single probability distribution over producing different terms, so that $\sum_{t \in V} P(t) = 1$.

- After generating a word, decide to stop or loop around.

Model M:

the	0.2
a	0.1
frog	0.01
toad	0.01
said	0.03
likes	0.02
that	0.04
...	...

P(s={frog said that toad likes frog}|M) =
$0.01 \times 0.01 \times 0.03 \times 0.04 \times 0.01 \times 0.02 \times 0.01$
$P(s|M) \approx 0.00000000000024$

Stochastic Language Models

Model M_1		Model M_2	
the	0.2	the	0.15
a	0.1	a	0.12
frog	0.01	frog	0.0002
toad	0.01	toad	0.0001
said	0.03	said	0.03
likes	0.02	likes	0.04
that	0.04	that	0.04
dog	0.005	dog	0.01
cat	0.003	cat	0.015
monkey	0.001	monkey	0.002
...	...	...	...

s	frog	said	that	toad	likes	that	dog
M_1	0.01	0.03	0.04	0.01	0.02	0.04	0.005
M_2	0.0002	0.03	0.04	0.0001	0.04	0.04	0.01

$P(s|M_1) = 0.00000000000048$
$P(s|M_2) = 0.000000000000000384$

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

Stochastic Language Models

- A statistical model for generating text has a probability distribution over strings (words) in a given language.
- How do we calculate probabilities over sequences of strings?
  - Use the chain rule to decompose the probability of a sequence of events into the probability of each successive event conditioned on earlier events:

$$P(t_1 t_2 t_3 t_4) = P(t_1)P(t_2|t_1)P(t_3|t_1 t_2)P(t_4|t_1 t_2 t_3)$$

  - Assume term independence: **the unigram language model**

$$P(t_1 t_2 t_3 t_4) = P(t_1)P(t_2)P(t_3)P(t_4)$$

  - Condition on the previous term: **the bigram language model**

$$P(t_1 t_2 t_3 t_4) = P(t_1)P(t_2|t_1)P(t_3|t_2)P(t_4|t_3)$$

Language Models for IR

- Users have a reasonable idea of terms that are likely to occur in the documents of interest.
- They will choose query terms that distinguish these documents from others in the collection.

Query Likelihood Model

- Treat each document d as the basis for a model M_d (e.g., unigram statistics)
- Rank documents d based on $P(M_d|Q)$
- Applying Bayes rule, we have:

$$P(M_d|Q) = \frac{P(Q|M_d) \cdot P(M_d)}{P(Q)}$$

  - $P(Q)$ is the same for all documents, so ignore
  - $P(M_d)$ [the prior] is often treated as the same for all documents
  - $P(Q|M_d)$ is the probability of Q given d's model M_d.

More Precisely ...

IR Approach

- Infer a language model for each document.
- Estimate the probability of generating the query according to each of these models
  - $P(Q|M_d)$ is the probability of Q given d's model, M_d
- Rank the documents according to these probabilities.
- Usually a unigram estimate of words is used
- Collection statistics: integral parts of the language model.

How do we estimate $P(Q|M_d)$?

Query Generation Probability

- The probability of producing the query given the language model of document d under the unigram assumption is:

$$P(Q|M_d) = \prod_{w \in Q} P(w|M_d) = \prod_{w \in Q} \frac{tf_{w,d}}{L_d}$$

- M_d : language model of document d
- $tf_{w,d}$: raw tf of word w in document d
- L_d : total number of tokens in document d

Insufficient Data

- Zero probability $P(w|M_d) = 0$
  - May not wish to assign a probability of zero to a document that is missing one or more of the query terms [gives strict conjunctive semantics]
  - Need to smooth probabilities
  - Many approaches to smoothing probability distributions to deal with this problem, e.g., adding 1, 1/2, or ϵ to counts.
- General approach here
  - A non-occurring term is possible, but no more likely than would be expected by chance in the collection.
    If $tf_{(w,d)} = 0$, then:

$$P(w|M_c) = \frac{cf_w}{T}$$

  - cf_w: raw count of word w in the collection
  - T raw collection size (the total number of tokens in the collection)

Mixture Model

- A simple idea: use a mixture between the document multinomial and the collection multinomial distribution:

$$P(w|d) = \lambda P(w|M_d) + (1 - \lambda)P(w|M_c)$$

  - linear interpolation language model
- Mixes the probability from the document with the general collection frequency of the word.
- Correctly setting λ is very important
- Can tune λ to optimize performance

Basic Mixture Model Summary

- General formulation of the LM for IR

$$P(Q|d) = \prod_{w \in Q} ((1-\lambda)P(w|M_c) + \lambda P(w|M_d))$$

  - $P(w|M_c)$ - general language model
  - $P(w|M_d)$ - individual-document 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

- 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:

$$P(Q|d) = \prod_{w \in Q} ((1-\lambda)P(w|M_c) + \lambda P(w|M_d)); \lambda = 1/2$$

- Query: revenue down
  - $P(Q|d_1) = ?$
  - $P(Q|d_2) = ?$
- Ranking: ?

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:

$$P(Q|d) = \prod_{w \in Q} ((1-\lambda)P(w|M_c) + \lambda P(w|M_d)); \lambda = 1/2$$

- Query: revenue down
  - $P(Q|d_1) = [(1/8 + 2/16)/2] * [(1/8 + 1/16)/2]$
    $= 1/8 * 3/32 = 3/256$
  - $P(Q|d_2) = [(1/8 + 2/16)/2] * [(0 + 1/16)/2]$
    $= 1/8 * 1/32 = 1/256$
- Ranking: $d_1 > d_2$

Example - Add-One Smoothing

- 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:

$$P(Q|d) = \prod_{w \in Q} P(w|M_d)$$

- Query: revenue down
  - $P(Q|d_1) = \frac{1+1}{8+14} * \frac{1+1}{8+14} = \frac{2}{22} * \frac{2}{22} = 0.00826446$
  - $P(Q|d_2) = \frac{1+1}{8+14} * \frac{0+1}{8+14} = \frac{2}{22} * \frac{1}{22} = 0.00413223$
- Ranking: $d_1 > d_2$