# Probabilistic Retrieval Models

Emine Yilmaz
emine.yilmaz@ucl.ac.uk

Some slides courtesy Javed A. Aslam@Northeastern

## About Me…

- Professor of Computer Science, Turing and ELLIS Fellow at UCL
- Amazon Scholar, working on Alexa
    - Previously a researcher at Microsoft
- Co-founder, Humanloop (A UCL Spin-out)
- Currently accepting PhD students

## What is a Retrieval Model?

- Model is an idealization or abstraction of an actual process
  - Mathematical models used to study the properties of the process, draw conclusions, make predictions
- Conclusions derived from a model depend on whether the model is a good approximation of the actual situation
- Retrieval models can attempt to describe the human process
  - e.g. the information need, interaction
  - Few do so meaningfully
- Retrieval models can describe the computational process
  - e.g. how documents are ranked

## Retrieval Models Overview

- Older models
  - Boolean retrieval
  - Vector Space model
- Probabilistic Models
  - Language models
  - BM25
- Combining evidence
  - Inference networks
  - Learning to Rank

## Retrieval Models Overview

- Older models
  - Boolean retrieval
  - Vector Space model
- Probabilistic Models
  - Language models
  - BM25
- Combining evidence
  - Inference networks
  - Learning to Rank

## Language Models

- A *language model* is a probability distribution over strings of text.
  - How likely is a given string in a given language?
- Consider probabilities for the following strings:
  - p1 = P("language model")
  - p2 = P("probability distribution")
  - p3 = P("a language model is a probability distribution")
- p2 > p1 > p3?
- Every possible string has some probability.
  - Depends on the language we are modelling

## Unigram Language Models

- A *unigram language model* is a probability distribution over words.
  - P("language"), P("model")
  - P("probability"), P("distribution")
  - P("a"), P("is")

- Each word has a probability, probabilities sum to 1.
- Assumption: words are independent, order doesn't matter.
  - P("language model") = P("language")P("model") = P("model language")
  - P("a language model is a probability distribution") = P("a probability distribution is a language model")
- "bag of words" model.

## Bigram Language Models

- A *bigram language model* is a probability distribution over pairs of words.
    - P("language model")
    - P("probability distribution")
- Or a collection of unigram language models, each conditioned on a word.
    - P("model" | "language")
    - P("distribution" | "probability")
- P("a language model is a probability distribution") = P("a")P("language" | "a")P("model" |"language")…P("distribution" | "probability")
- In general, n-gram models: models that consider n words
- Our main focus in this lecture: Unigram language models

## Language Models (LM) for IR

- Each document D is a sample of language.
    - Consider all possible strings the author could have written while producing the document.
    - Some strings are more likely than others
    - $P(s \mid M_D)$ is the probability that the author would write string s.

- Three main approaches to LM in IR: Given a query Q and document D,
    - Query likelihood model: $P(Q|M_D)$
        - What is the probability to generate the given query, given a document language model?

    - Document likelihood model: $P(D|M_Q)$
        - What is the probability to generate the given document, given a query language model?

    - Divergence model: $D(P(M_D) \,||\, P(M_Q))$
        - How ”close” are 2 statistical models?

# Query-Likelihood Model

- Standard language modelling approach

- Rank documents by the probability that the query could be generated by the document model
  - estimate a language model $M_D$ for every document D in the collection
  - rank documents by the probability of “generating” the query

$$P(Q \mid M_D) = P(q_1 \ldots q_k \mid M_D) = \prod_{i=1}^{k} P(q_i \mid M_D)$$

- Drawbacks:
  - no notion of relevance in the model: everything is random sampling
  - user feedback / query expansion not part of the model
    - examples of relevant documents cannot help us improve the language model

# Document-Likelihood Model

- Flip the direction of the query-likelihood approach
  - estimate a language model $M_Q$ for the query Q
  - rank documents D by the likelihood of being a random sample from $M_Q$

$$P(D \mid M_Q) = \prod_{w \in D} P(w \mid M_Q)$$

- $M_Q$ expected to "predict" a typical relevant document

- Drawbacks:
  - different doc lengths, probabilities not comparable
  - favours documents that contain frequent (low content) words

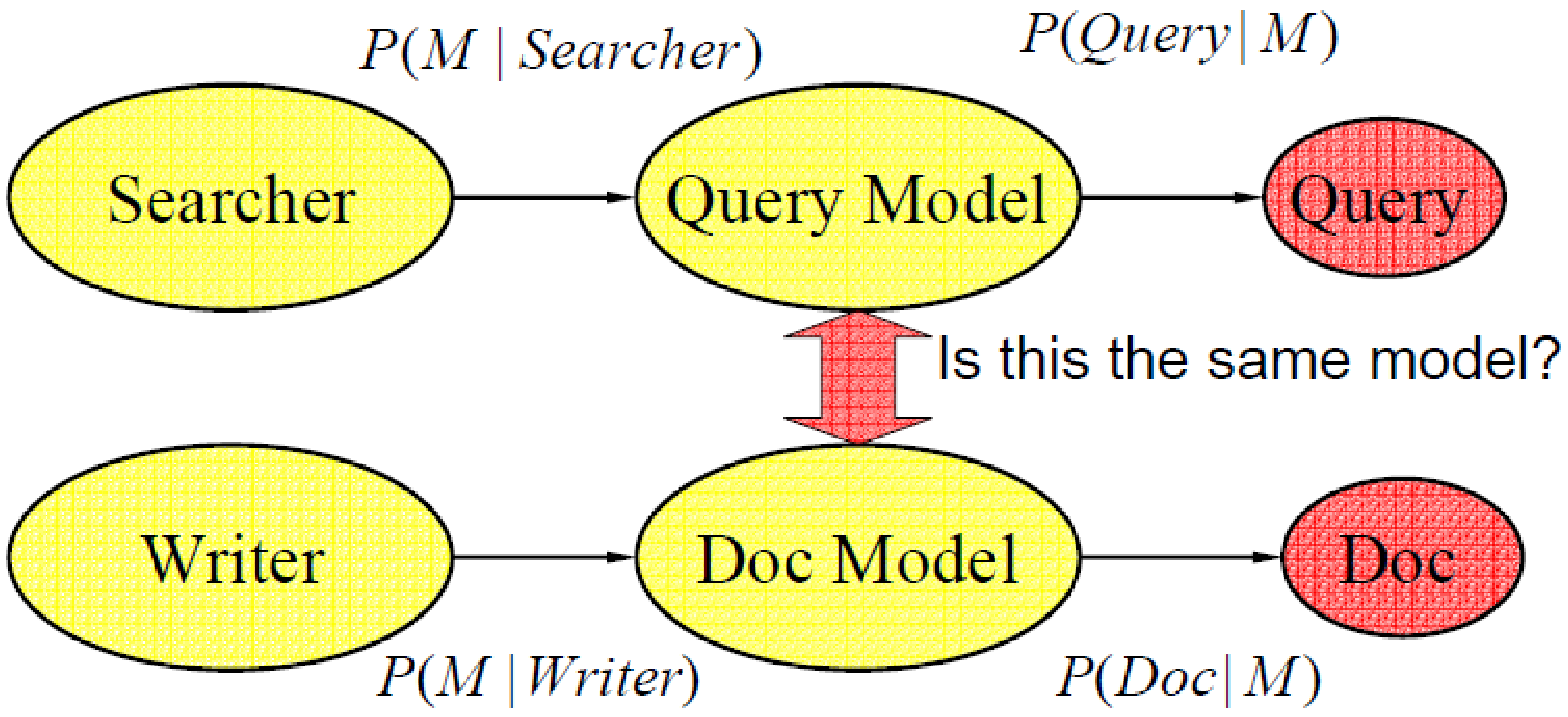
## Divergence Model

- Combine advantages of query-likelihood and document-likelihood models
  - estimate a model of both the query $M_Q$ and the document $M_D$
  - directly compare similarity of the two models
  - natural measure of similarity is cross-entropy (Kullback-Leiblar divergence)

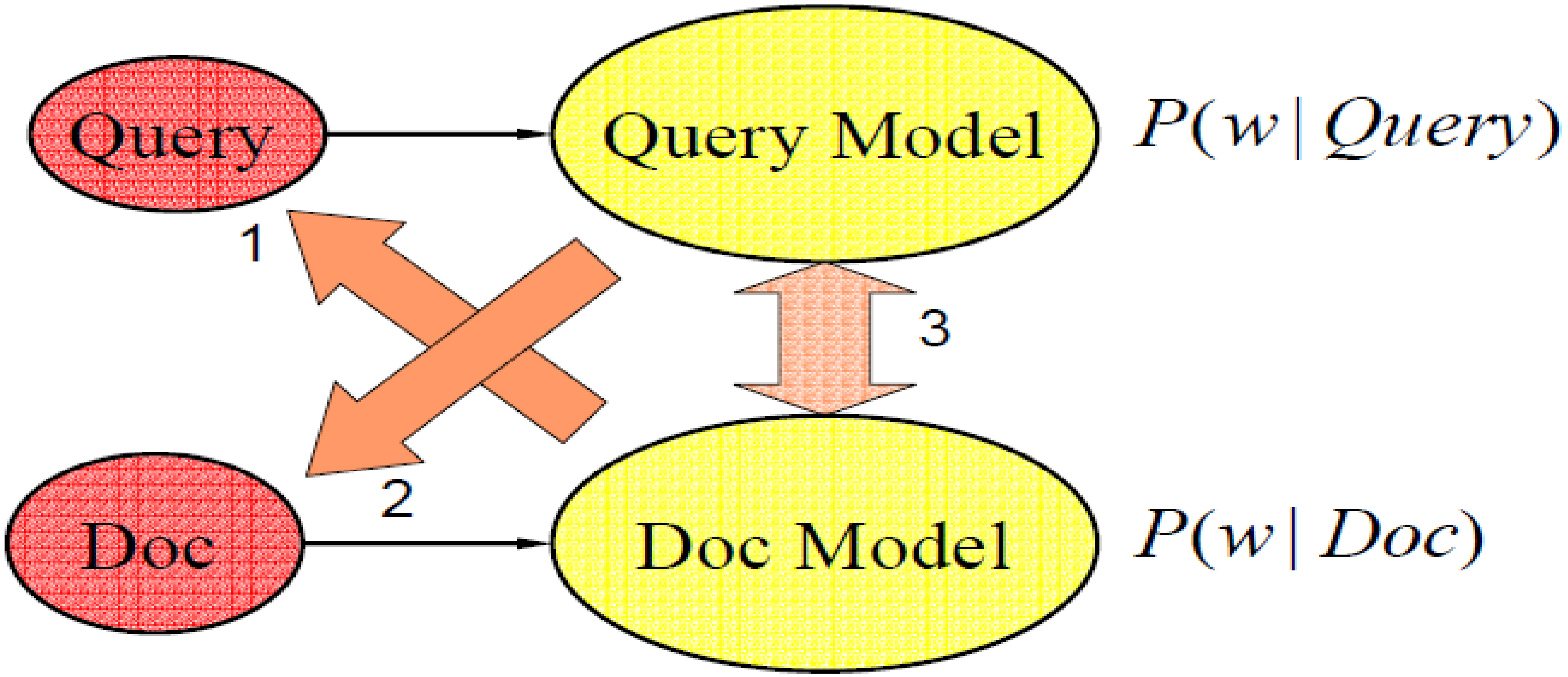
  - $H(M_Q || M_D) = -\sum_w P(w|M_Q) \log(P\,(w|M_D) / P(w|M_Q))$
    $$\overset{\text{RANK}}{=} -\sum_w P(w|M_Q) \log P\,(w|M_D)$$
  - number of bits needed to encode $M_Q$ using $M_D$

- Cross-entropy is not symmetric: use H ($M_Q$ || $M_D$)
  - reverse works consistently worse, favours different documents

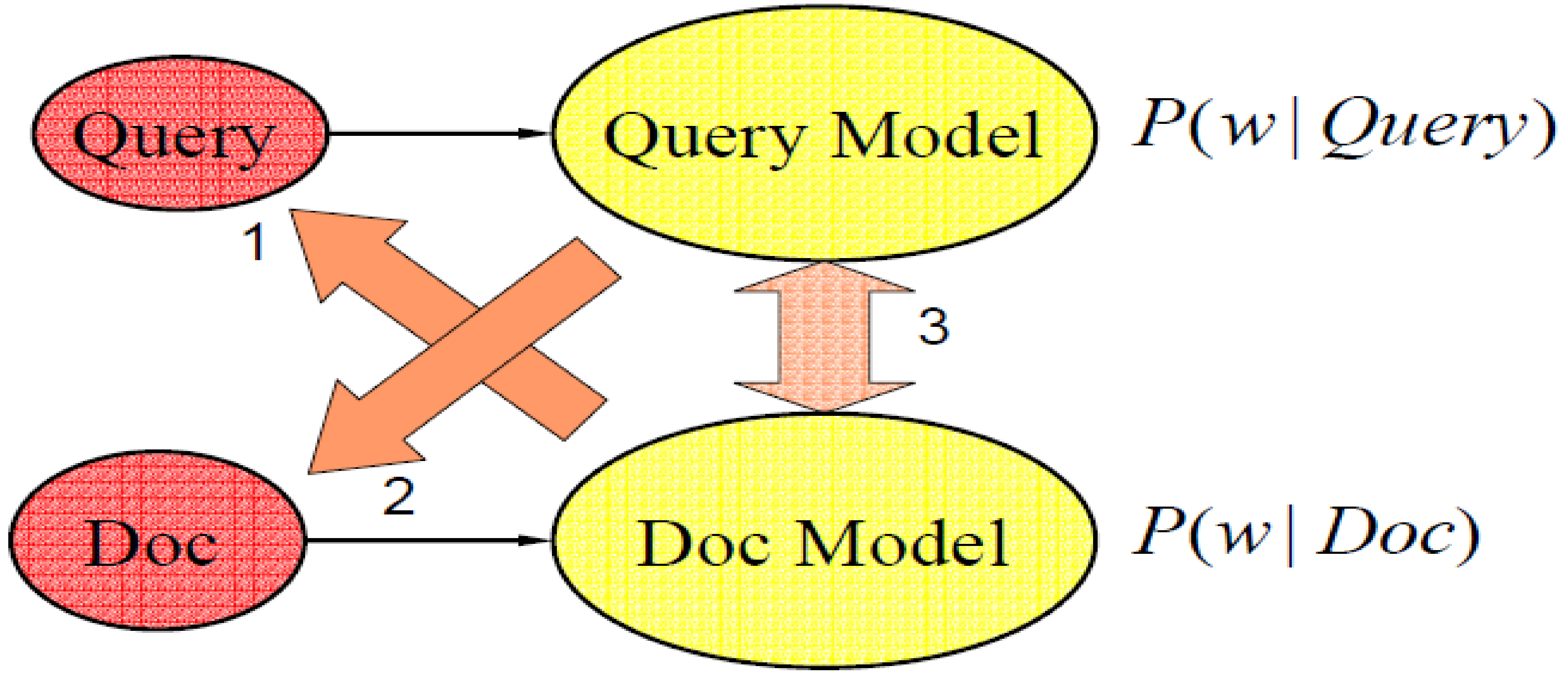
# Models of Text Generation



# Retrieval with Language Models



# Retrieval with Language Models



1: Query likelihood approach

2: Document likelihood approach

3: Divergence approach

# Constructing the Language Models

- We need to estimate $M_D$ and/or $M_Q$ from Q and/or D
  - Language models are distribution over words

- Estimating the distribution:

$$M_D = P(w \mid D) = \frac{f_{w,D}}{|D|}$$

  - Maximum likelihood estimate
  - Makes the observed value of $f_{w,D}$ most likely

- If query words are missing from document, score will be zero
  - Missing 1 out of 4 query words same as missing 3 out of 4

## Example: Unigram Query-Likelihood Model

- Document:

The majority of Americans consider tobacco advertising a major influence in promoting the killer habit. Approximately 57% of the public thinks that cigarette advertising causes people to smoke. Also, 47% thinks that cigarette advertising makes it harder for smokers to give up the habit. If the tobacco industry didn't agree with these stats it wouldn't concentrate so heavily on using young models in its ads.

**Query:** ***"tobacco advertising"***

$P(Q|M_D) =$ *2/65* *3/65* $P(Q|D) = 6/4225$

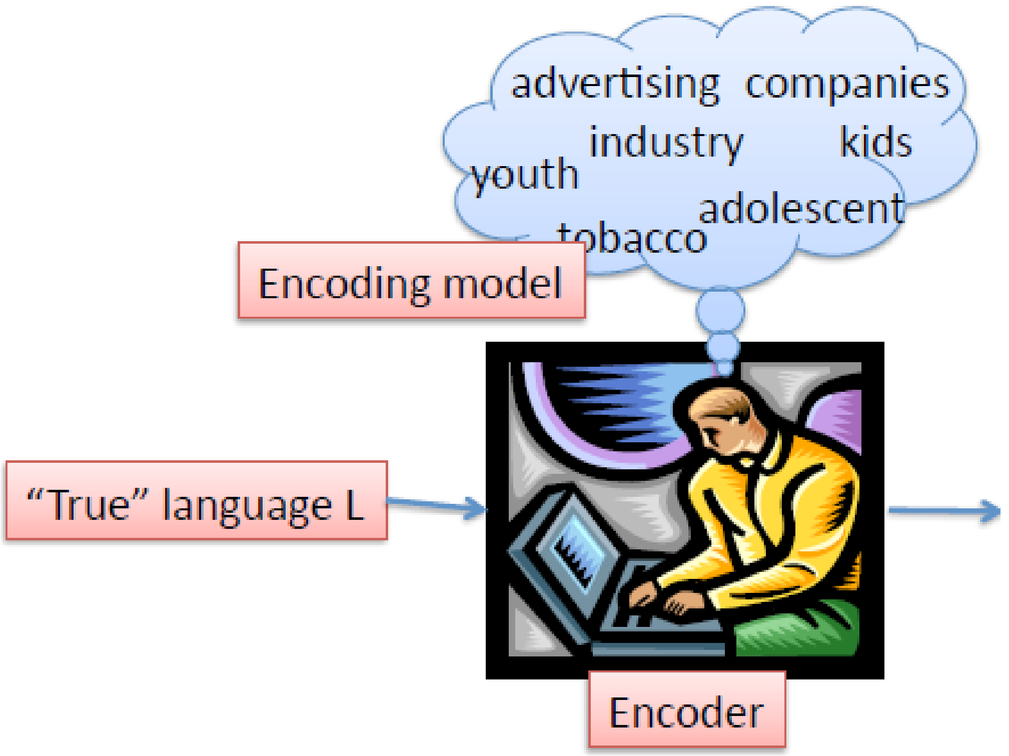
***"tobacco companies"***

$P(Q|M_D) =$ *2/65* *0/65* $P(Q|D) = 0$

***"tobacco companies and the young"***

$P(Q|D) = 0$

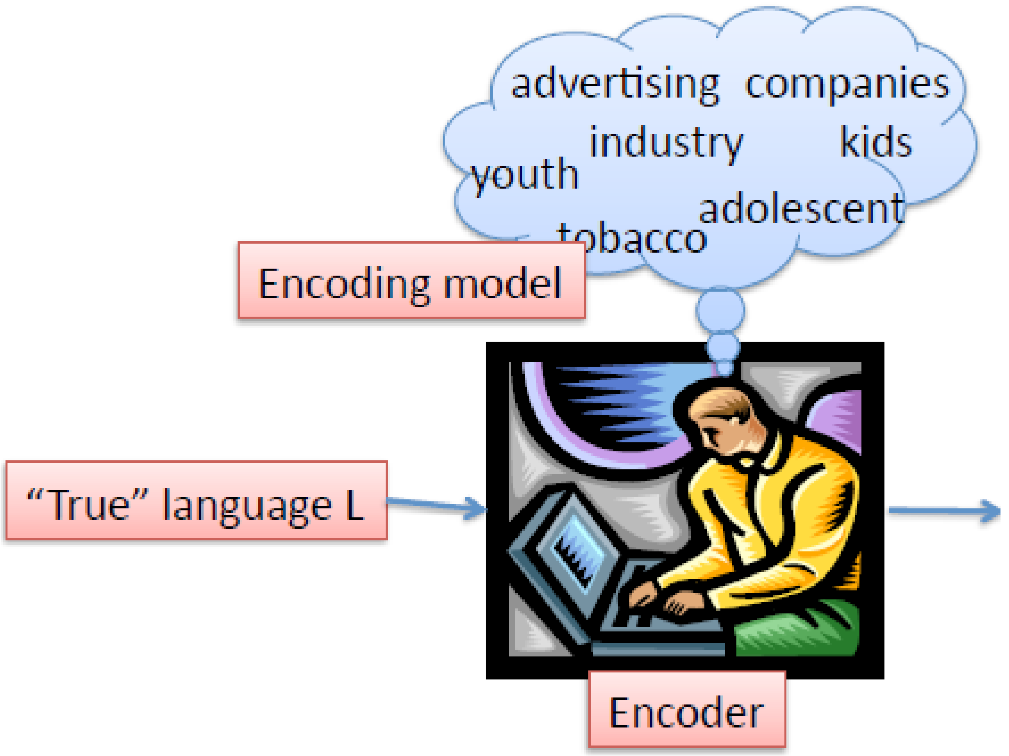
# Language Generation as Encoding



The majority of Americans consider tobacco advertising a major influence in promoting the killer habit. Approximately 57% of the public thinks that cigarette advertising causes people to smoke. Also, 47% thinks that cigarette advertising makes it harder for smokers to give up the habit. If the tobacco industry didn't agree with these stats it wouldn't concentrate so heavily on using young models in its ads.

Encoded language

# Language Generation as Encoding



The majority of Americans consider tobacco advertising a major influence in promoting the killer habit. Approximately 57% of the public thinks that cigarette advertising causes people to smoke. Also, 47% thinks that cigarette advertising makes it harder for smokers to give up the habit. If the tobacco industry didn't agree with these stats it wouldn't concentrate so heavily on using young models in its ads.

Encoded language

**Problem: We only have the encoded language.**
**We need to estimate our model from that.**

# Solution: Smoothing

- Document texts are a *sample* from the language model
  - Missing words should not have zero probability of occurring

- *Smoothing* is a technique for estimating probabilities for missing (or unseen) words
  - lower (or *discount*) the probability estimates for words that are seen in the document text

  - assign that “left-over” probability to the estimates for the words that are not seen in the text

- Gives every term (word) some probability of occurring in every document.

# Simple Smoothing Methods: Laplace Smoothing

- Count events in observed data
- Add 1 to every count
- Renormalize to obtain probabilities
- Corresponds to having uniform priors over words

- If number of occurrences of words in D are ($m_1$, $m_2$, …, $m_{|V|}$) with $\sum_i m_i = N$, where |V|: number of unique words in the entire collection (vocabulary size)
  - Max likelihood estimates: $\left(\frac{m_1}{|D|}, \frac{m_2}{|D|}, \ldots, \frac{m_{|V|}}{|D|}\right)$
  - Laplace estimates: $\left(\frac{m_1+1}{|D|+|V|}, \frac{m_2+1}{|D|+|V|}, \ldots, \frac{m_{|V|}+1}{|D|+|V|}\right)$

# Discounting Methods

- Laplace smoothing gives too much weight to unseen terms

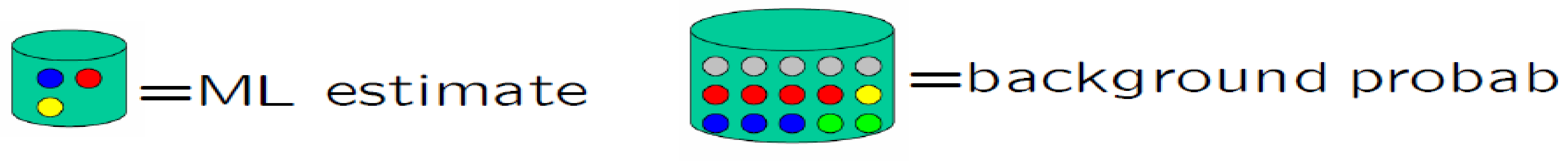
- Lidstone correction:
  - Add small number ε to every term count, renormalize:

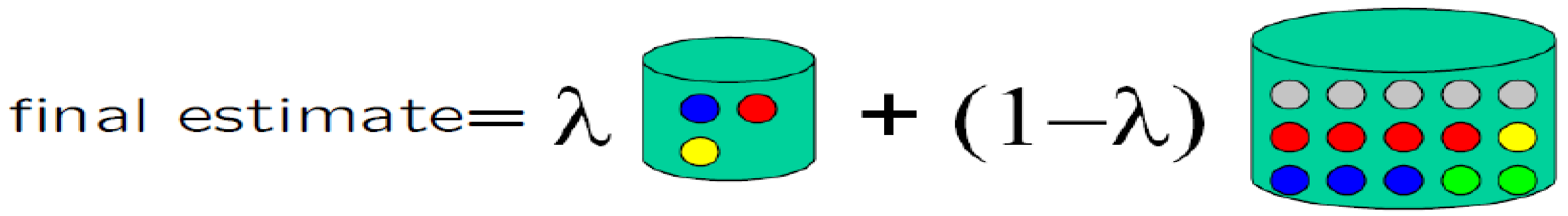
$$P(w|D) = \frac{tf_{w,D} + \varepsilon}{|D| + \varepsilon|V|}$$

- Absolute discounting:
  - Subtract ε, re-distribute the probability mass

## Interpolation Methods

- Problem with all discounting methods:
  - discounting treats unseen words equally (add or subtract **ε**)
  - some words are more frequent than others
- Idea: use background probabilities
  - "interpolate" ML estimates with the estimates from the collection
    - E.g. General English expectations (computed as relative frequency of a word in a large collection)
  - reflects expected frequency of events





# Smoothing with Background

- Estimate for unseen words is $P(w|C)$, where $P(w|C)$ is the probability for word w in the *collection* language model for collection (background probability)

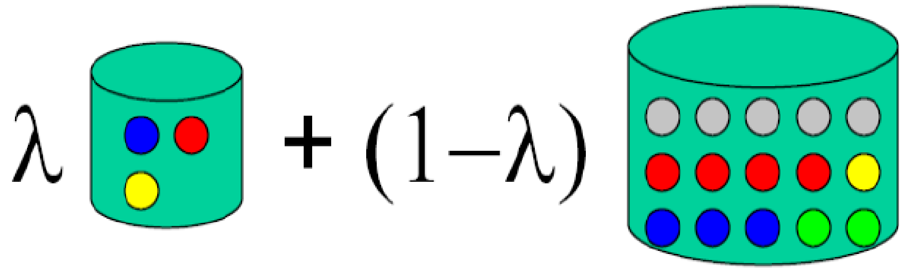
- Estimate for words that occur is
  - $\lambda P(w|D) + (1 - \lambda) P(w|C)$

- $\lambda$ is a parameter
  - Correctly setting $\lambda$ is very important
  - Different forms of estimation come from how $\lambda$ *is set*

# Jelinek-Mercer Smoothing

- Start simple
  - set **λ** to be a constant, independent of document, query

- Tune to optimize retrieval performance
  - optimal value of **λ** varies with different databases, query sets, etc.

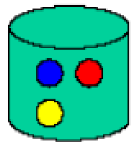


# Dirichlet Smoothing

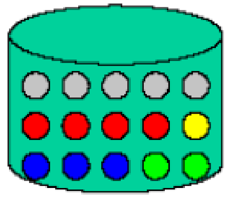
- Problem with Jelinek-Mercer:
  - longer documents provide better estimates
    - could get by with less smoothing

- Make smoothing depend on sample size
  - N is length of sample = document length
  - μ is a constant

$$\underbrace{N / (N + \mu)}_{\lambda}$$



$$+ \underbrace{\mu / (N + \mu)}_{(1-\lambda)}$$



## Example: Query-likelihood Example with Dirichlet Smoothing

- Query: “president lincoln”

- For the term “president”
  - $f_{qi,D} = 15$, $c_{qi} = 160{,}000$
- For the term “lincoln”
  - $f_{qi,D} = 25$, $c_{qi} = 2{,}400$

- Number of word occurrences in the document $|D|$ is assumed to be 1,800

- Number of word occurrences in the collection is $10^9$
  - 500,000 documents times an average of 2,000 words
- $\mu = 2{,}000$

# Example: Query-likelihood Example with Dirichlet Smoothing

- $P(Q|D_M) \propto \log P(Q|D_M) = \log \prod_{\forall q_i \in Q} P(q_i | D) = \sum_{\forall q_i \in Q} \log P(q_i | D)$

$$= \log P(\text{"president"} | D) + \log P(\text{"lincoln"} | D)$$

$$= \log[\frac{1800}{1800+2000} \cdot \frac{15}{1800} + \frac{2000}{1800+2000} \cdot \frac{160000}{10^9}] +$$

$$\log[\frac{1800}{1800+2000} \cdot \frac{25}{1800} + \frac{2000}{1800+2000} \cdot \frac{2400}{10^9}]$$

$$= -10.55$$

# Example: Query-likelihood Example with Dirichlet Smoothing

| Frequency of "president" | Frequency of "Lincoln" | Query Likelihood Score |
|---|---|---|
| 15 | 25 | -10.55 |
| 15 | 1 | -13.75 |
| 15 | 0 | -19.05 |
| 1 | 25 | -12.99 |
| 0 | 25 | -14.40 |

## What is the best smoothing?

- Don’t make too many assumptions.
- Don’t smooth too much.

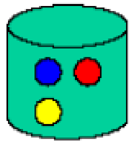
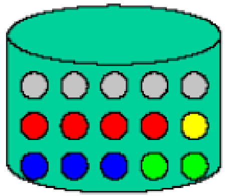
- Dirichlet works well for short queries (need to tune the parameter)
- Jelinek-Mercer works well for longer queries (also needs tuning)
- In general, Dirichlet smoothing seems to provide the best “happy-medium”.

## Language Models: Summary

- Goal: estimate a model M from a sample text S
    - Use maximum-likelihood estimator
    - Count the number of times each word occurs in S, divide by length
- Smoothing to avoid zero frequencies
    - Discounting methods: add or subtract a constant, redistribute mass
    - Better: interpolate with background probability of a word
    - Smoothing has a role similar to IDF in classical models
    - Smoothing parameters very important

## Example: Query-likelihood Example with Dirichlet Smoothing

- Make smoothing depend on sample size
  - N is length of sample = document length
  - μ is a constant

$\underbrace{N / (N + \mu)}_{\lambda}$  + $\underbrace{\mu / (N + \mu)}_{(1-\lambda)}$ 

# Where is tf-idf weight?

$$P(Q|D) \alpha \log P(Q|D) = \sum_{i=1}^{n} \log(\lambda \frac{f_{q_i,D}}{|D|} + (1-\lambda)\frac{c_{q_i}}{|C|})$$

$$= \sum_{i:f_{q_i,D}>0} \log(\lambda \frac{f_{q_i,D}}{|D|} + (1-\lambda)\frac{c_{q_i}}{|C|}) + \sum_{i:f_{q_i,D}=0} \log((1-\lambda)\frac{c_{q_i}}{|C|})$$

$$= \sum_{i:f_{q_i,D}>0} \log \frac{\lambda \frac{f_{q_i,D}}{|D|} + (1-\lambda)\frac{c_{q_i}}{|C|}}{(1-\lambda)\frac{c_{q_i}}{|C|}} + \sum_{i=1}^{n} \log((1-\lambda)\frac{c_{q_i}}{|C|})$$

Add and subtract

$$\sum_{i:f_{q_i,D}>0} \log(1-\lambda)\frac{c_{q_i}}{|C|}$$

$$\overset{rank}{=} \sum_{i:f_{q_i,D}>0} \log\left( \frac{\lambda \frac{f_{q_i,D}}{|D|}}{(1-\lambda)\frac{c_{q_i}}{|C|}} + 1 \right)$$

# Where is tf-idf weight?

$$P(Q \mid D) \alpha \log P(Q \mid D) = \sum_{i=1}^{n} \log\left(\lambda \frac{f_{q_i,D}}{|D|} + (1-\lambda)\frac{c_{q_i}}{|C|}\right)$$

$$= \sum_{i:f_{q_i,D}>0} \log\left(\lambda \frac{f_{q_i,D}}{|D|} + (1-\lambda)\frac{c_{q_i}}{|C|}\right) + \sum_{i:f_{q_i,D}=0} \log\left((1-\lambda)\frac{c_{q_i}}{|C|}\right)$$

$$= \sum_{i:f_{q_i,D}>0} \log \frac{\lambda \frac{f_{q_i,D}}{|D|} + (1-\lambda)\frac{c_{q_i}}{|C|}}{(1-\lambda)\frac{c_{q_i}}{|C|}} + \sum_{i=1}^{n} \log\left((1-\lambda)\frac{c_{q_i}}{|C|}\right)$$

$$\overset{rank}{=} \sum_{i:f_{q_i,D}>0} \log\left(\frac{\lambda \frac{f_{q_i,D}}{|D|}}{(1-\lambda)\frac{c_{q_i}}{|C|}} + 1\right)$$

- Proportional to the term frequency
- Inversely proportional to the collection frequency

## Comparison with Vector Space

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

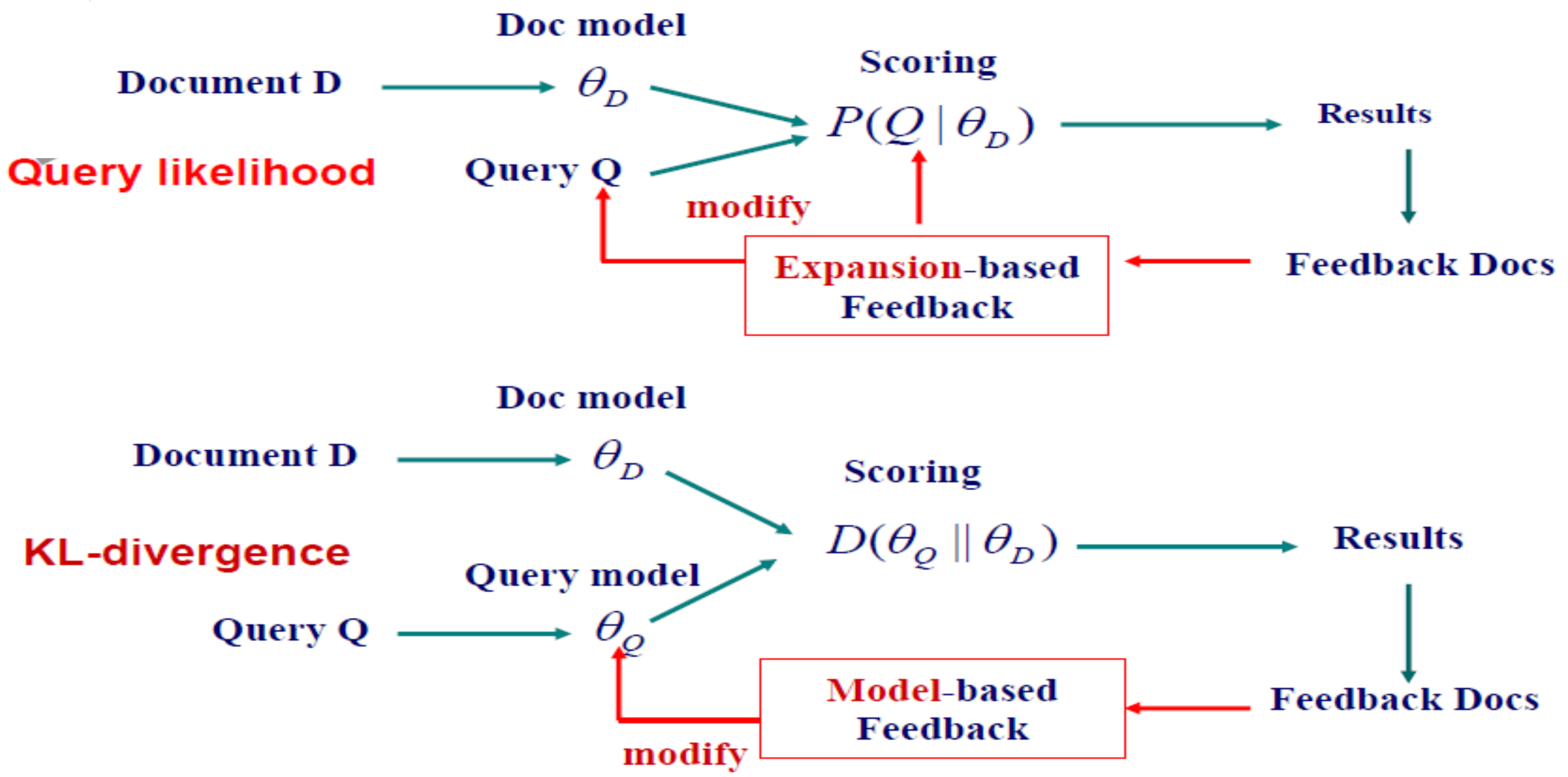
- Different in others
  - Based on probability rather than similarity
    - Intuitions are probabilistic rather than geometric
  - Details of use of document length and term, document, and collection frequency differ

## Relevance Feedback

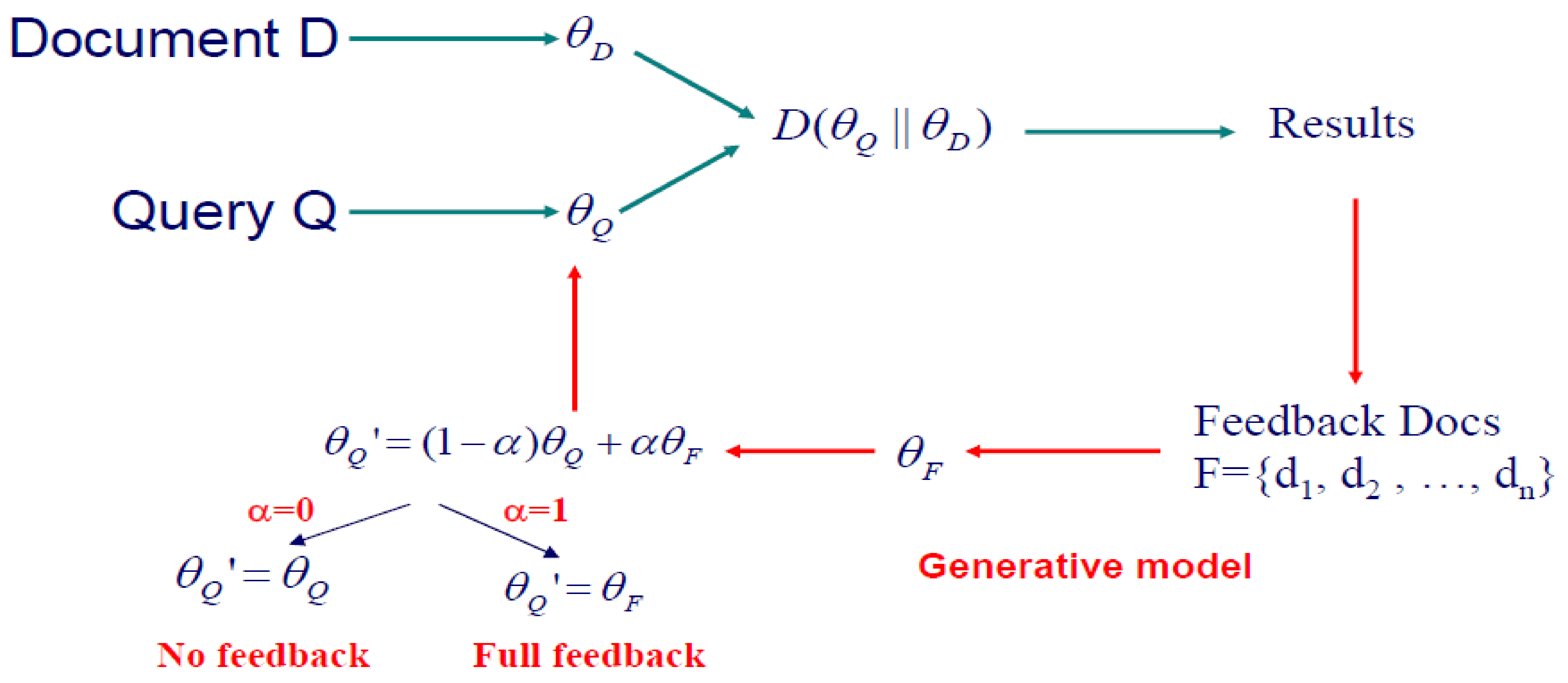
- Relevance feedback: user feedback on relevance of docs in initial set of results
  - User issues a (short, simple) query
  - The user marks some results as relevant or non-relevant.
  - The system computes a better representation of the information need based on feedback.
  - Relevance feedback can go through one or more iterations.

- Idea: it may be difficult to formulate a good query when you don’t know the collection well, so iterate

# Expansion-based vs. Model-based Feedback



# Feedback as Model Interpolation



## Summary

- Three different approaches to language modelling
  - Query likelihood model
  - Document likelihood model
  - Divergence model
- Different approaches to smoothing
  - Important role in performance
- Relevance feedback and language modelling