11-442 / 11-642:
Search Engines

Best-Match Retrieval: Statistical Language Models

Jamie Callan
Carnegie Mellon University
callan@cs.cmu.edu

1

Outline

Statistical language models

- Introduction
- Query likelihood model
- Kullback-Leibler (KL) Divergence
- Indri

HW2 implementation (see notes posted online)

- Indri default beliefs
- Window operator

2

2

Statistical Language Models

A statistical language model uses a probability distribution to determine the probability of terms or term sequences

Common types of models:

Language model

- Unigrams: $p(t_i \mid \theta)$ p ("search" | θ)
- Bigrams: $p(t_i \mid t_{i-1}, \theta)$ p ("engines" | "search", θ)
- Trigrams: $p(t_i \mid t_{i-2}, t_{i-1}, \theta)$ p ("class" | "search", "engines", θ)

Bigrams, trigrams, etc., haven't helped much for IR (so far)

- But they are a key idea in speech recognition

Statistical Language Models

Word histogram

Term	tf_d
camera	17
image	13
picture	11
up	8
movie	8
like	7
mode	7
software	7
red	6
digital	5
eye	5
shutter	5
sony	5

Unigram language model

Term	$P(t\|\theta_d)$
camera	0.09551
image	0.07303
picture	0.06180
up	0.04494
movie	0.04494
like	0.03933
mode	0.03933
software	0.03933
red	0.03371
digital	0.02809
eye	0.02809
shutter	0.02809
sony	0.02809

$$P(t \mid d) = \frac{tf_{t,d}}{length_d}$$

A Language Model can be Created From Any Language Sample

Examples

- A document collection
- A document
- Also sentence, paragraph, chapter, …
- A query

This is similar to the vector space

- A vector can be created for any sample of text
- A language model can be created for any sample of text

Term	$P(t \mid \theta_d)$
camera	0.09551
image	0.07303
picture	0.06180
up	0.04494
movie	0.04494
like	0.03933
mode	0.03933
software	0.03933
red	0.03371
digital	0.02809
eye	0.02809
shutter	0.02809
sony	0.02809

5

5

Terminology

d: A document

θ_d: The language model for document d

q: A query

θ_q: The language model for query q

d and θ_d are not the same thing

- The document and the model of the document are different
- However, to simplify notation, we will often treat them as the same
  - i.e., $p(d \mid q)$ instead of $p(d \mid \theta_q)$

6

6

Retrieval Model Based Upon Statistical Language Models

A document d defines a probability distribution θ_d over index terms

- E.g., the probability of generating/observing an index term

A query q also defines a probability distribution θ_q

- A sparse distribution (more on this later)

t	p (t)
apple	0.0204
banana	0.0001
campaign	0.0034
dog	0.0102
:	:

Two common methods of using language modeling to rank document d

- Rank d by $p(d \mid \theta_q)$ "query likelihood"
- Rank d by the similarity of θ_d and θ_q "KL divergence"

Outline

Statistical language models

- Introduction
- Query likelihood model
- Kullback-Leibler (KL) Divergence
- Indri

HW2 implementation (see notes posted online)

- Indri default beliefs
- Window operator

Rank by P(d|q): The Query-Likelihood Approach

Task: Rank documents by p(d |q)

- Given a query q, what is the probability of document d?

Query: digital camera	
Term	$\mathbf{P(t\|\theta_q)}$
digital	0.5
camera	0.5

Problems

- q is a very sparse language model (few terms)
- q contains little frequency information (qtf=1 for most terms)

Solution

$$p(d\,|\,q) = \frac{p(q\,|\,d)\,p(d)}{p(q)}$$ **Bayes rule**

$$\propto p(q\,|\,d)\,p(d)$$ **Drop document-independent term**

Key issues

- How are p(q|d) and p(d) estimated?

9

9

Query-Likelihood Approach: Estimating *p(d)*

It is simple and convenient to assume that p(d) is uniform

- Later we will consider non-uniform p(d)

So…

$$p(d\,|\,q) \propto p(q\,|\,d)\,p(d)$$

$$\propto p(q\,|\,d)$$ **Drop the constant (uniform probability)**

10

10

What is p(q|d)?
Simple Unigram Approach

Assume that a query is composed of independent terms

- Unjustified, but convenient
- Doesn't hurt effectiveness of other models (e.g., vector space)

Then …

$$p(q \mid d) = \prod_{q_i \in q} p(q_i \mid d)$$

q_i: term in query q

This should look a little familiar (although the notation differs)

- The score of (q, d) is based on the scores of (q_i, d)
- Similar to what we saw with BM25 and the vector space
- How is $p(q_i|d)$ calculated?

Estimating $p(q_i|d)$

Maximum likelihood estimation (MLE) is a simple approach

$$p_{MLE}(q_i \mid d) = \frac{tf_{q_i,d}}{length(d)}$$

Is this a good estimate?

- Estimates are based on small samples (a single document)
  - So, perhaps not very accurate
- $p_{MLE}(q_i|d) = 0$ if q_i isn't in document d

$$p(q \mid d) = \prod_{q_i \in q} p_{MLE}(q_i \mid d) = 0$$

Boolean AND

  - This is exact-match AND
  - We would prefer best-match AND

Estimating $p(q_i|d)$: Smoothing

Smoothing is used to solve two problems in the language modeling framework

- Imprecise probability estimates from MLE
  - E.g., for infrequent terms (few observations)
  - E.g., for short documents (small sample)
- Probability estimates for unobserved terms
  - E.g., query terms that don't occur in the document

But … there are many smoothing methods … which to use?

13

13

Estimating $p(q_i|d)$: Jelinek-Mercer ("Mixture Model") Smoothing

Linear interpolation with a reference language model

$$p(q_i \mid d) = (1-\lambda) p_{MLE}(q_i \mid d) + \lambda p_{MLE}(q_i \mid C)$$

Average an estimate from a small sample (document) with an estimate from a large sample (collection)

Smoothing decreases as $\lambda \rightarrow 0$

What value of λ is best?

- Small λ (little smoothing) is best for short queries
- Larger λ (more smoothing) is better for long queries
- We will see later in the lecture why this is true

14

14

Estimating p(q_i|d):
Bayesian Smoothing With Dirichlet Priors

Method #2: Bayesian smoothing using Dirichlet priors

$$p(q_i|d) = \frac{tf_{q_i,d} + \mu\, p_{MLE}(q_i|C)}{length(d) + \mu}$$

$$p_{MLE}(q_i|C) = \frac{ctf_{q_i}}{length_{tokens}(C)}$$

Increase tf of terms that are expected to be frequent in d (small sample) because they are frequent in C (large sample)

What value of μ is best?

- μ in [1,000-10,000] seems best
- μ might be approximately related to average document length

15

15

Estimating p(q_i|d):
Smoothing

We know that tf.idf weights are effective
…how do they relate to statistical language models

tf is easy to see

$$p_{MLE}(q_i \mid d) = \frac{tf_{q_i,d}}{length(d)}$$

idf is harder to see

- Smoothing provides an idf-like effect
- Next slides…

16

16

No Smoothing Example:

$$p(q \mid d) = \prod_{q_i \in q} p_{MLE}(q_i \mid d)$$

Two query terms: p (apple | C) = 0.01, p (ipad | C) = 0.001

"frequent" **"rare"**

Document d_1

doclen =50, tf_{apple} = 2, tf_{ipad} = 3

(2/50) ×
(3/50) =

0.04 × 0.06 =

$p(q|d_1)$ = 0.0024

Document d_2

doclen =50, tf_{apple} = 3, tf_{ipad} = 2

(3/50) ×
(2/50) =

0.06 × 0.04 =

$p(q|d_2)$ = 0.0024

The model <u>does not</u> distinguish between frequent and rare terms

17

Jelinek-Mercer Smoothing Example:

$$p(q \mid d) = \prod_{q_i \in q} (1-\lambda) p_{MLE}(q_i \mid d) + \lambda\, p_{MLE}(q_i \mid C)$$

Two query terms: p (apple | C) = 0.01, p (ipad | C) = 0.001

λ = 0.4 **"frequent"** **"rare"**

Document d_1

doclen =50, tf_{apple} = 2, tf_{ipad} = 3

(0.6 × (2/50) + 0.4 × 0.01) ×
(0.6 × (3/50) + 0.4 × 0.001) =

(0.6 × 0.04 + 0.004) ×
(0.6 × 0.06 + 0.0004) =

(0.024+0.004)×(0.036+0.0004)=

$p(q|d_1)$ = 0.001019

Document d_2

doclen =50, tf_{apple} = 3, tf_{ipad} = 2

(0.6 × (3/50) + 0.4 × 0.01) ×
(0.6 × (2/50) + 0.4 × 0.001) =

(0.6 × 0.06 + 0.004) ×
(0.6 × 0.04 + 0.0004) =

(0.036+0.004)×(0.024+0.0004)=

$p(q|d_2)$ = 0.000976

The model <u>does</u> distinguish between frequent and rare terms

18

Dirichlet Smoothing Example:

$$p(q_i|d) \quad = \quad \frac{tf_{q_i,d} + \mu\, p_{MLE}(q_i|C)}{length(d) + \mu}$$

Two query terms: p (apple | C) = 0.01, p (ipad | C) = 0.001

μ **= 2000** "frequent" "rare"

Document d_1

doclen =50, $tf_{apple} = 2$, $tf_{ipad} = 3$

$(2 + 2000 \times 0.01)\ / (50 + 2000) \times$
$(3 + 2000 \times 0.001) / (50 + 2000) =$

$(2 + 20) / 2050 \times$
$(3 + 2)\ / 2050 =$

$0.0107 \times 0.0024 =$

$p(q|d_1) = 0.000026$

Document d_2

doclen =50, $tf_{apple} = 3$, $tf_{ipad} = 2$

$(3 + 2000 \times 0.01)\ / (50 + 2000) \times$
$(2 + 2000 \times 0.001) / (50 + 2000) =$

$(3 + 20) / 2050 \times$
$(2 + 2)\ / 2050 =$

$0.0112 \times 0.0020 =$

$p(q|d_2) = 0.000022$

The model <u>does</u> distinguish between frequent and rare terms

Why Does Smoothing Work?

$$p(q\,|\,d) = \prod_{q_i \in q} (1-\lambda) p_{MLE}(q_i\,|\,d) + \lambda\, p_{MLE}(q_i\,|\,C)$$

doclen =50, $tf_{apple} = 2$, $tf_{ipad} = 2$

p (apple | d) = 0.6 × (2/50) + 0.4 × 0.010 = 0.0280

p (ipad | d) = 0.6 × (2/50) + 0.4 × 0.001 = 0.0244

What is the effect of matching one <u>additional</u> instance of a term?

p_δ (apple | d) = 0.6 × (1/50) = 0.012

p_δ (ipad | d) = 0.6 × (1/50) = 0.012

– The <u>unsmoothed</u> effect of each match is the same

The incremental value of matching a term is multiplied by the p (q | d) of other query terms

– $tf_{apple} = 3$, $tf_{ipad} = 2$: p (q | d) = (0.028 + <u>0.012</u>) × 0.0244 = 0.000976

– $tf_{apple} = 2$, $tf_{ipad} = 3$: p (q | d) = 0.028 × (0.0244 + <u>0.012</u>) = 0.001019

Estimating $p(q_i|d)$: The idf-like Effect of Jelenick-Mercer Smoothing

How Jelenick-Mercer smoothing has an idf-like effect

- Several steps of math manipulation are omitted for clarity

$$
\begin{aligned}
p(q|d) &= \prod_{q_i \in q} p(q_i|d) \\
&= \prod_{q_i \in q} \big((1-\lambda) p_{MLE}(q_i|d) + \lambda p_{MLE}(q_i|C)\big) \\
&\propto \prod_{q_i \in q} \left(\frac{(1-\lambda) p_{MLE}(q_i|d)}{\lambda p_{MLE}(q_i|C)} + 1 \right)
\end{aligned}
$$

J-M smoothing

"tf effect"
"idf effect"

Estimating $p(q_i|d)$: Jelinek-Mercer ("Mixture Model") Smoothing

HW2 requires you to think about the effects of parameters

$$p(q_i \mid d) = (1-\lambda) p_{MLE}(q_i \mid d) + \lambda p_{MLE}(q_i \mid C)$$

What happens when λ approaches 0?

- There is no smoothing (no "idf like" effect)

$$p(q_i \mid d) = (1-\lambda) p_{MLE}(q_i \mid d)$$

Estimating $p(q_i|d)$: Jelinek-Mercer ("Mixture Model") Smoothing

HW2 requires you to think about the effects of parameters

$$p(q_i \mid d) = (1-\lambda) p_{MLE}(q_i \mid d) + \lambda p_{MLE}(q_i \mid C)$$

How important is smoothing to <u>queries</u> of different lengths?

- Short queries
  - Few query terms, so (usually) <u>every</u> query term must match
  - "idf effect" is less important, so small λ is best
- Long queries
  - Many query terms, so (usually) <u>most</u> query terms must match
  - "idf effect" is more important
  - Give priority to the less common terms, so larger λ is best

23

23

Estimating $p(q_i|d)$: Bayesian Smoothing With Dirichlet Priors

HW2 requires you to think about the effects of parameters

$$p(q_i \mid d) = \frac{tf_{q_i,d} + \mu\, p_{MLE}(q_i \mid C)}{length(d) + \mu}$$

What happens when μ approaches 0?

- There is no smoothing (maximum likelihood only)

$$p(q_i \mid d) = \frac{tf_{q_i,d}}{length(d)}$$

24

24

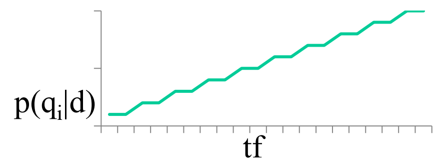
Estimating $p(q_i|d)$: Bayesian Smoothing With Dirichlet Priors

HW2 requires you to think about the effects of parameters

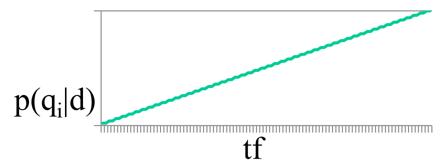
$$p(q_i \mid d) = \frac{tf_{q_i,d} + \mu \, p_{MLE}(q_i \mid C)}{length(d) + \mu}$$

How important is smoothing to <u>documents</u> of different lengths?

- Short documents
  - Probabilities are more granular
  - Larger μ is more important



- Long documents
  - Probabilities are more smooth
  - Larger μ is less important



25

Estimating $p(q_i|d)$: Two-Stage Smoothing

Mixture modeling and Bayesian smoothing with Dirichlet priors are both effective and common … which should you use?

- **They do different things**
  - **Mixture model:** Compensates for different word importance
    - » "idf effect"
  - **Dirichlet prior:** Improves the estimate of the document model
    - » "small sample", "unseen words"
- **Two-stage smoothing gives the best effects of both methods**

$$p(q_i \mid d) = (1-\lambda)\frac{tf_{q_i,d} + \mu \, p_{MLE}(q_i \mid C)}{length(d) + \mu} + \lambda \, p_{MLE}(q_i \mid C)$$

26

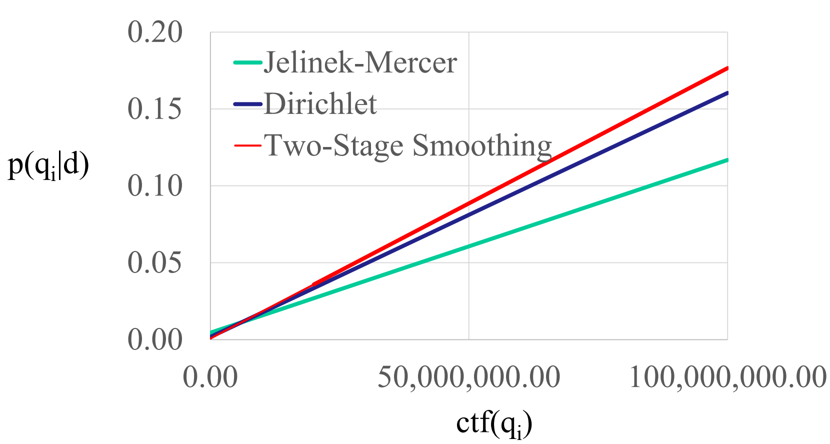
Query Likelihood With Two-Stage Smoothing: Putting it All Together

$$
\begin{aligned}
p(q \mid d) &= \prod_{q_i \in q} p(q_i \mid d) \\
&= \prod_{q_i \in q} \left((1-\lambda) \frac{tf_{q_i,d} + \mu\, p_{MLE}(q_i \mid C)}{length(d) + \mu} + \lambda\, p_{MLE}(q_i \mid C) \right) \\
&= \prod_{q_i \in q} \left((1-\lambda) \frac{tf_{q_i,d} + \mu \frac{ctf(q_i)}{length(c)}}{length(d) + \mu} + \lambda \frac{ctf(q_i)}{length(c)} \right)
\end{aligned}
$$

$tf_{q,d}$: Term frequency of term q in document d
ctf (q): Term frequency of term q in the entire collection
length (d): Total number of word occurrences in document d
length (c): Total number of word occurrences in collection c

Estimating $p(q_i|d)$: Smoothing Comparison

How do different smoothing methods treat different types of terms?

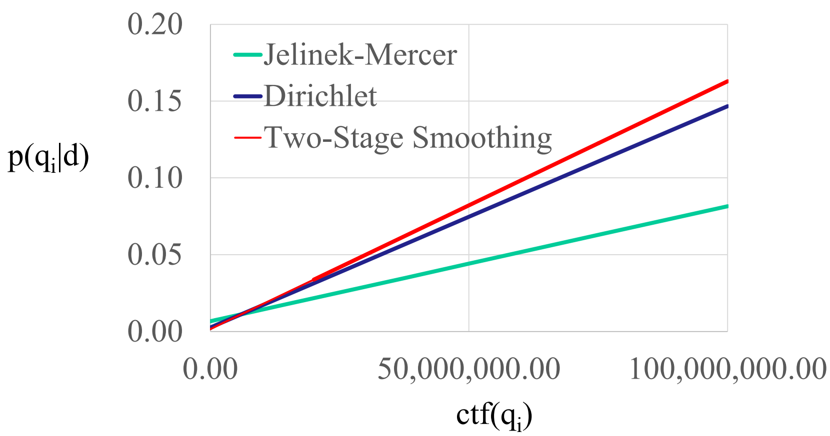


Assumptions
- tf: 5
- Doc len: 450
- Coll len: 530M
- λ: 0.6
- μ: 2,500

λ and μ control the slope (more smoothing)

Estimating $p(q_i|d)$: Smoothing Comparison

How do different smoothing methods treat different types of terms?



Assumptions

- tf: 5
- Doc len: 450
- Coll len: 530M
- λ: 0.4
- μ: 1,500

λ and μ control the slope (less smoothing)

Outline

Statistical language models

- Introduction
- Query likelihood model
- Kullback-Leibler (KL) Divergence
- Indri

HW2 implementation (see notes posted online)

- Indri default beliefs
- Window operator

Retrieval Model Based Upon Statistical Language Models

How is a language model used to rank document d?

- Rank d by $p(d \mid \theta_q)$ "query likelihood"
- Rank d by the similarity of θ_d and θ_q "KL divergence"

In some cases, these approaches are equivalent…

KL Divergence: Measuring the Similarity of Language Models

Kullback-Leibler divergence measures the relative entropy between two probability mass functions

$$KL(a||b) = \sum_{x \in X} a(x) log \frac{a(x)}{b(x)}$$

Language models are probability mass functions

- The probability of observing a term in a sample of text
- Why not use KL divergence to measure the similarity of q and d?

Measuring the Similarity of Statistical Language Models

Ranking by (negative) KL-divergence

$$-KL(q \| d) = -\sum_{w \in V} p(w|q) \log \frac{p(w|q)}{p(w|d)}$$

33 © 2020 Jamie Callan

33

Measuring the Similarity of Statistical Language Models

Ranking by (negative) KL-divergence

$$-KL(q \| d) = -\sum_{w \in V} p(w|q) \log \frac{p(w|q)}{p(w|d)}$$

$$= -\sum_{q_i \in q} p(q_i|q) \log \frac{p(q_i|q)}{p(q_i|d)}$$

P(w|q) = 0 when w ∉ q

34 © 2020 Jamie Callan

34

Measuring the Similarity of Statistical Language Models

Ranking by (negative) KL-divergence

$$
\begin{aligned}
-KL(q \| d) &= -\sum_{w \in V} p(w|q) \log \frac{p(w|q)}{p(w|d)} \\
&= -\sum_{q_i \in q} p(q_i|q) \log \frac{p(q_i|q)}{p(q_i|d)} \\
&= -\left(\sum_{q_i \in q} p(q_i|q) \log p(q_i|q) - \sum_{q_i \in q} p(q_i|q) \log p(q_i|d) \right)
\end{aligned}
$$

P(w|q) = 0 when w ∉ q

35

35

Measuring the Similarity of Statistical Language Models

Ranking by (negative) KL-divergence

$$
\begin{aligned}
-KL(q \| d) &= -\sum_{w \in V} p(w|q) \log \frac{p(w|q)}{p(w|d)} \\
&= -\sum_{q_i \in q} p(q_i|q) \log \frac{p(q_i|q)}{p(q_i|d)} \\
&= -\left(\sum_{q_i \in q} p(q_i|q) \log p(q_i|q) - \sum_{q_i \in q} p(q_i|q) \log p(q_i|d) \right) \\
&\propto \sum_{q_i \in q} p(q_i|q) \log p(q_i|d)
\end{aligned}
$$

P(w|q) = 0 when w ∉ q

Drop constant term

36

36

Measuring the Similarity of Statistical Language Models

Ranking by (negative) KL-divergence

$$
\begin{aligned}
-KL(q \,\|\, d) &= -\sum_{w \in V} p(w \mid q) \, \log \frac{p(w \mid q)}{p(w \mid d)} \\
&= -\sum_{q_i \in q} p(q_i \mid q) \, \log \frac{p(q_i \mid q)}{p(q_i \mid d)} && \text{P(w|q) = 0 when } w \notin q \\
&= -\left(\sum_{q_i \in q} p(q_i \mid q) \, \log p(q_i \mid q) - \sum_{q_i \in q} p(q_i \mid q) \log p(q_i \mid d) \right) \\
&\propto \sum_{q_i \in q} p(q_i \mid q) \log p(q_i \mid d) && \text{Drop constant term} \\
&\propto \sum_{q_i \in q} \frac{1}{|q|} \log p(q_i \mid d) && \text{Assume uniform probabilities for query terms } \left(p(q_i|q) = \frac{1}{|q|} \right)
\end{aligned}
$$

37

37

Measuring the Similarity of Statistical Language Models

Ranking by (negative) KL-divergence

$$
\begin{aligned}
-KL(q \,\|\, d) &= -\sum_{w \in V} p(w \mid q) \, \log \frac{p(w \mid q)}{p(w \mid d)} \\
&= -\sum_{q_i \in q} p(q_i \mid q) \, \log \frac{p(q_i \mid q)}{p(q_i \mid d)} && \text{P(w|q) = 0 when } w \notin q \\
&= -\left(\sum_{q_i \in q} p(q_i \mid q) \, \log p(q_i \mid q) - \sum_{q_i \in q} p(q_i \mid q) \log p(q_i \mid d) \right) \\
&\propto \sum_{q_i \in q} p(q_i \mid q) \log p(q_i \mid d) && \text{Drop constant term} \\
&\propto \sum_{q_i \in q} \frac{1}{|q|} \log p(q_i \mid d) && \text{Assume uniform probabilities for query terms } \left(p(q_i|q) = \frac{1}{|q|} \right) \\
&\propto \sum_{q_i \in q} \log p(q_i \mid d) && \text{Drop constant term}
\end{aligned}
$$

38

38

Two Different Paths to a Common Destination

Query likelihood ranks by $p(q \mid d) = \prod_{q_i \in q} p(q_i \mid d)$

KL diverge ranks by $\sum_{q_i \in q} \log p(q_i \mid d)$

These are rank equivalent

$$p(q \mid d) = \prod_{q_i \in q} p(q_i \mid d) \propto \sum_{q_i \in q} \log p(q_i \mid d)$$

So…the query likelihood and KL divergence approaches are equivalent (when document priors are uniform)

Outline

Statistical language models

- Introduction
- Query likelihood model
- Kullback-Leibler (KL) Divergence
- Indri

HW2 implementation (see notes posted online)

- Indri default beliefs
- Window operator

Inference Nets + Language Models: Indri

The Indri retrieval model combines statistical language models with Bayesian inference networks

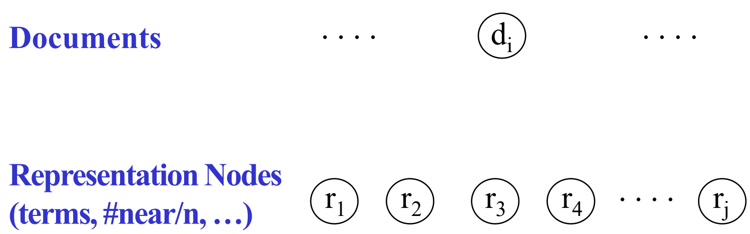
- A probabilistic retrieval model
- Structured queries
- Documents with multiple representations
- Documents with structure (a later lecture)

The theory is sophisticated, but the underlying ideas are familiar

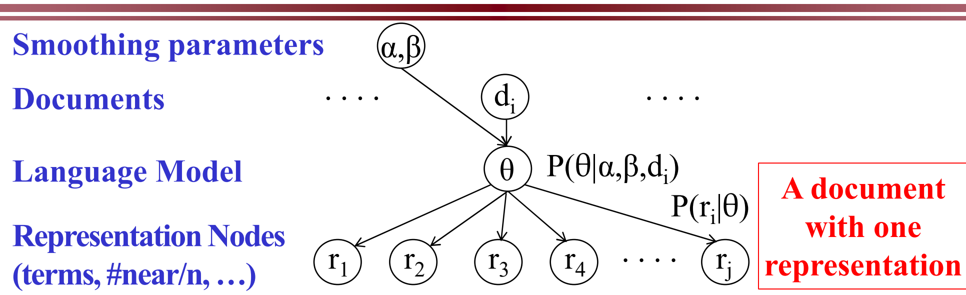
Inference Nets + Language Models: Indri

Documents . . . . d_i . . . .

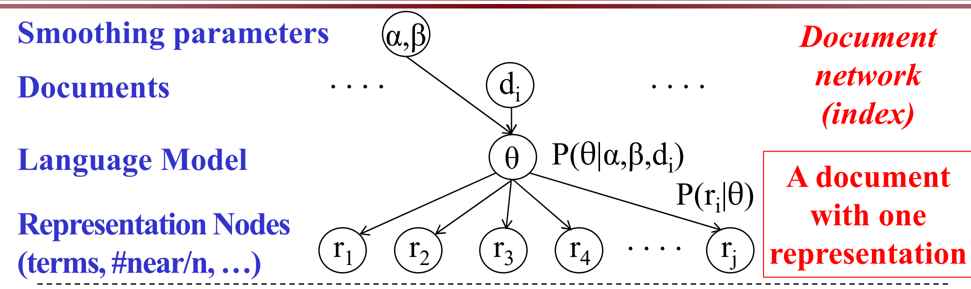
Inference Nets + Language Models: Indri



Inference Nets + Language Models: Indri



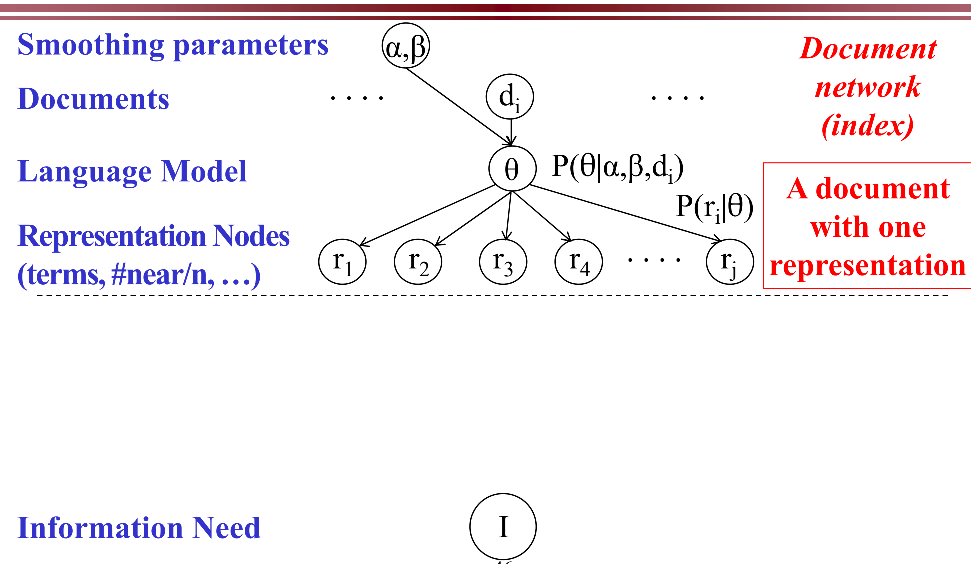
Inference Nets + Language Models: Indri



45 © 2020 Jamie Callan

45

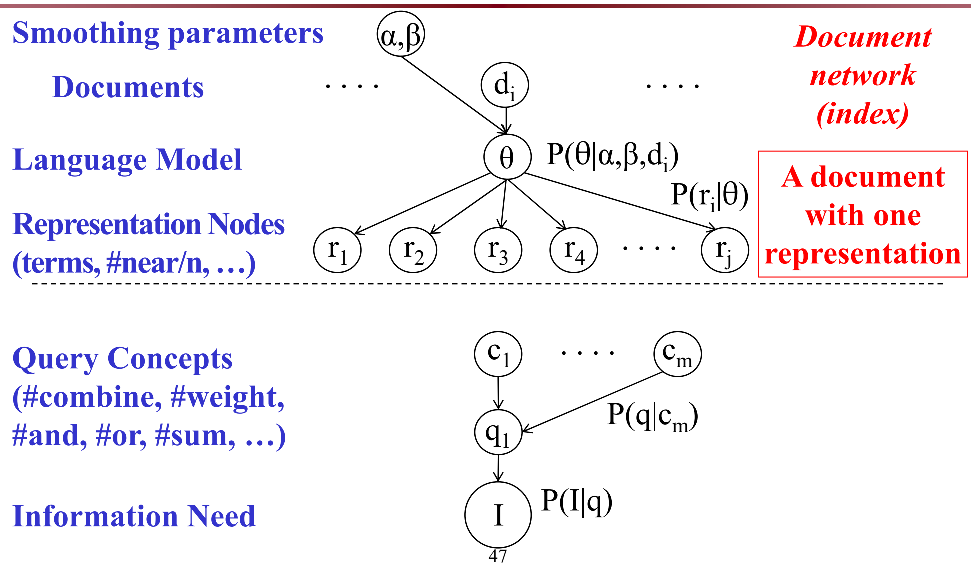
Inference Nets + Language Models: Indri



46 © 2020 Jamie Callan

46

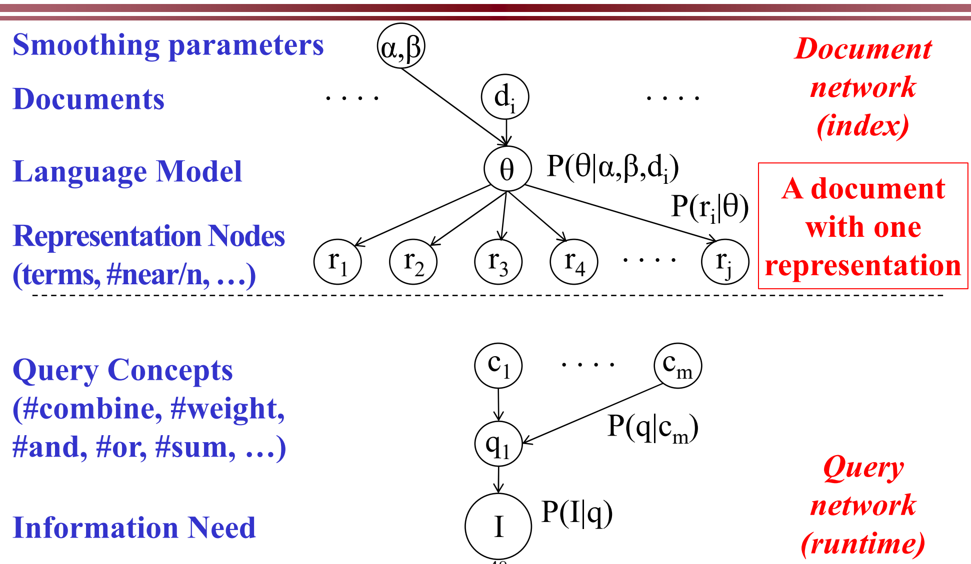
Inference Nets + Language Models: Indri



47

47

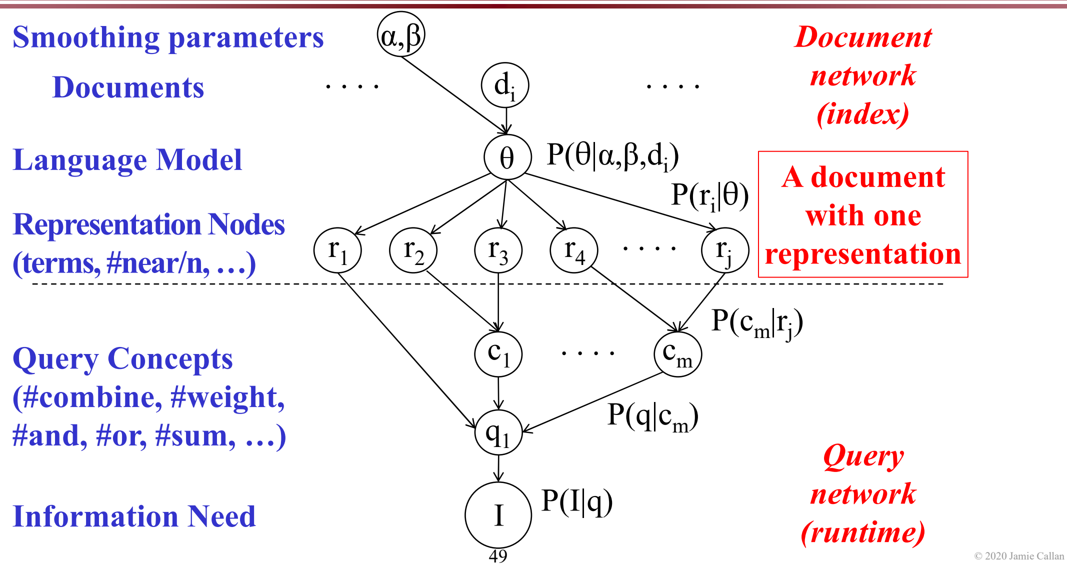
Inference Nets + Language Models: Indri



48

48

Inference Nets + Language Models: Indri

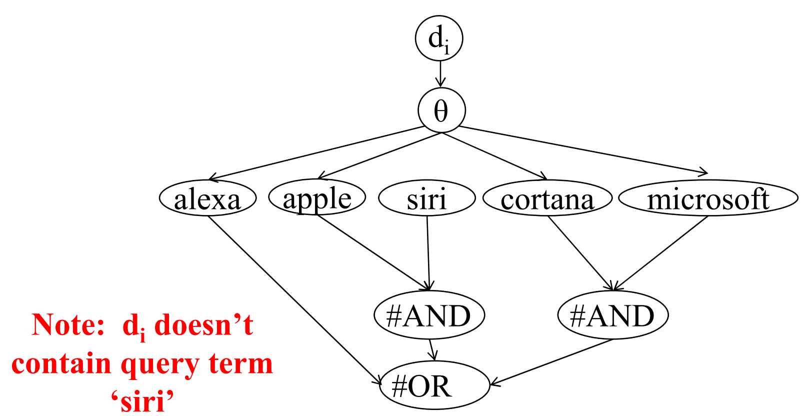


49

49

Inference Nets + Language Models: Indri

This isn't as complicated as it looks

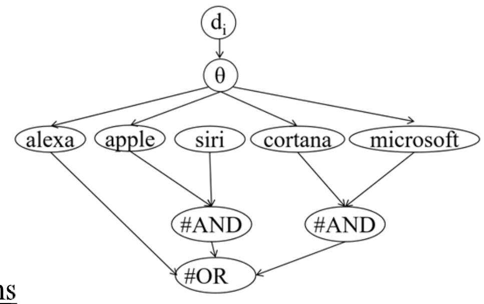


50

50

Inference Nets + Language Models: Indri

This isn't as complicated as it looks



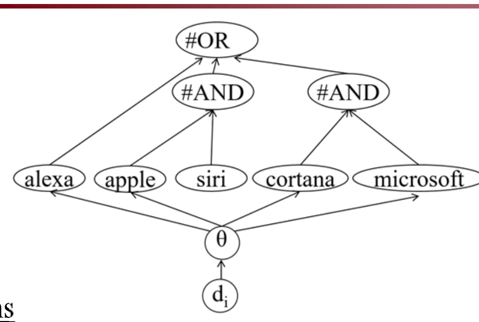
- Document + smoothing parameters (α,β) → language model (θ)
- The language model is defined by representation nodes (r_i)
  - – Terms stored in the index
  - – Query operators that create index terms
    - » QryIop (#SYN, #NEAR/n, #WINDOW/n, …)
- Information needs are represented by queries
- Queries are composed of query operators that combine evidence
    - » QrySop (#AND, #OR, #SUM, #WSUM, …)

51

51

Inference Nets + Language Models: Indri

This isn't as complicated as it looks

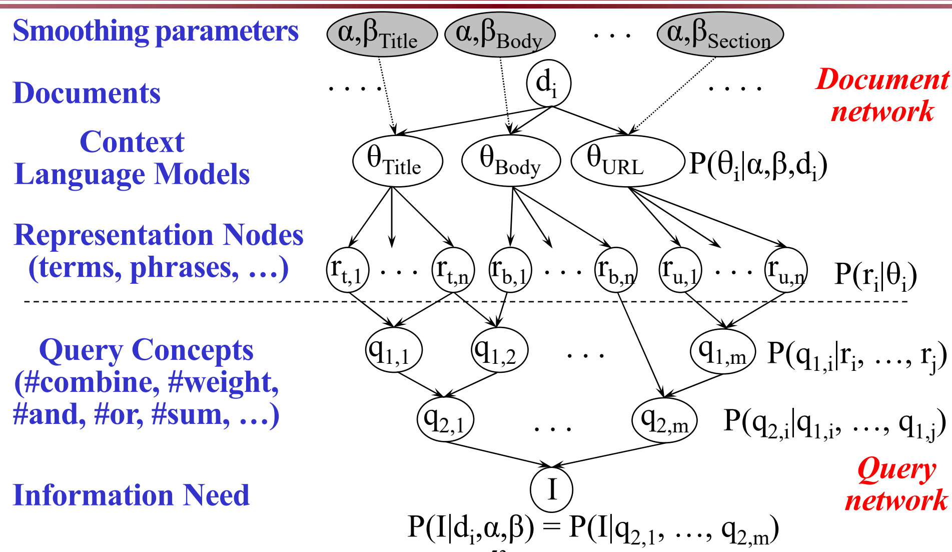


- Document + smoothing parameters (α,β) → language model (θ)
- The language model is defined by representation nodes (r_i)
  - – Terms stored in the index
  - – Query operators that create index terms
    - » QryIop (#SYN, #NEAR/n, #WINDOW/n, …)
- Information needs are represented by queries
- Queries are composed of query operators that combine evidence
    - » QrySop (#AND, #OR, #SUM, #WSUM, …)

52

52

Inference Nets + Language Models: Indri



53 © 2020 Jamie Callan

53

Indri Term Weights

Indri can use any language modeling method to estimate $p(r_i|\theta)$

- This weight is equivalent to $p(q_i|d)$ in most language models
- Dirichlet smoothing is the most common choice

$$p(q_i \mid d) = \frac{tf_{q_i,d} + \mu\, p_{MLE}(q_i \mid C)}{length(d) + \mu}$$

- Two-stage smoothing is also used

$$p(q_i \mid d) = (1-\lambda)\frac{tf_{q_i,d} + \mu\, p_{MLE}(q_i \mid C)}{length(d) + \mu} + \lambda\, p_{MLE}(q_i \mid C)$$

54 © 2020 Jamie Callan

54

Indri Query Operators

Operators that map inverted lists to an inverted list are easy

- E.g., InvertedList+ → InvertedList
- E.g., #NEAR/n, #WINDOW/n, #SYN, …
- To the language model these look just like ordinary terms

What about operators that combine scores?

- E.g., ScoreList+ → ScoreList
- E.g., #AND, #OR, …
- Indri views these as operators that <u>combine evidence</u>
  – "evidence" means "conditional probabilities"

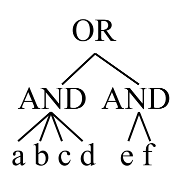
Indri Query Operators

AND is the default query operator for most language modeling systems

- Typically implemented as the product of the argument weights

$$p_{and}(q \mid d) = \prod_{q_i \in q} p(q_i \mid d)$$

- This is pleasing theoretically, but it has a problem
  – Typically #AND (a b c d) < #AND (e f)



Indri's AND operator uses the geometric mean

$$p_{and}(q \mid d) = \prod_{q_i \in q} p(q_i \mid d)^{\boxed{\frac{1}{|q|}}}$$

Which AND operator has more impact on the score of the OR operator?

Indri Query Operators

We may want to give different weight to different evidence

#wand (
0.7 #and (time traveler wife)
0.2 #and (#near/1 (time traveler) #near/1 (traveler wife))
0.1 #and (#window/8 (time traveler) #window/8 (traveler wife)))

Indri's WAND operator (also called WEIGHT) gives this control

- WAND: weighted AND

$$p_{wand}(q \mid d) = \prod_{q_i \in q} p(q_i \mid d)^{\frac{w_i}{\sum w_i}}$$

57

57

Indri Query Operators

Indri provides an OR operator

$$p_{or}(q \mid d) = 1 - \prod_{q_i \in q} (1 - p(q_i \mid d))$$

58

58

Indri Query Operators

The AND operator assumes that its arguments are estimates of <u>independent probabilities</u>

- E.g., #AND (buy ipad)

The WSUM operator assumes that its arguments are different ways of estimating <u>the same probability</u>

- E.g., the probability that this document is about "apple")
  #WSUM (0.3 apple.title 0.1 apple.url 0.6 apple.body)
- WSUM takes a <u>weighted average</u> of the estimates
  – It should be called WAVG – the name is a historical artifact

$$p_{wsum}(q \mid d) = \sum_{q_i \in q} \frac{w_i}{\sum w_i} p(q_i \mid d)$$

59 © 2020 Jamie Callan

59

Indri Query Operators

$$\text{AND, COMBINE}: \quad p_{and}(q \mid d) = \prod_{q_i \in q} p(q_i \mid d)^{\frac{1}{|q|}}$$

$$\text{WAND, WEIGHT}: \quad p_{wand}(q \mid d) = \prod_{q_i \in q} p(q_i \mid d)^{\frac{w_i}{w}}, \quad w = \sum w_i$$

$$\text{OR}: \quad p_{or}(q \mid d) = 1 - \prod_{q_i \in q} \left(1 - p(q_i \mid d)\right)$$

$$\text{WSUM}: \quad p_{wsum}(q \mid d) = \sum_{q_i \in q} \frac{w_i}{w} p(q_i \mid d), \quad w = \sum w_i$$

$$\text{NOT}: \quad p_{not}(q \mid d) = 1 - p(q \mid d)$$

60 © 2020 Jamie Callan

60

Outline

Statistical language models

- Introduction
- Query likelihood model
- Kullback-Leibler (KL) Divergence
- Indri

HW2 implementation (see notes posted online)

- Indri default beliefs
- Window operator

61

61

Retrieval Models Summary

We have discussed the following retrieval models

- Unranked Boolean
- Ranked Boolean, using tf scoring
- Vector Space, using lnc.ltc scoring
- Okapi BM25
- Language Models
  - Query likelihood, with two-stage smoothing
  - KL Divergence and Jensen-Shannon Divergence
  - Indri

That's a lot of material … what's important?

62

62

Retrieval Models Summary

Important differences among the models that you should know

- Unranked vs. ranked retrieval
- Exact-match vs. best-match (partial-match) retrieval
- The kinds of statistics used by most ranking functions
  - The theories are different, but the stats are mostly the same
- How well the different models support query operators
  - Inverted-list operators vs. score operators
  - Strict Boolean vs. probabilistic Boolean
- Which models you could use (or not use) for different types of tasks

Retrieval Models Summary

All of these retrieval models are used widely

- There must be a reason why … you should know what it is