

Introduction to Information Retrieval

<http://informationretrieval.org>

IIR 12: Language Models for IR

Hinrich Schütze

Institute for Natural Language Processing, Universität Stuttgart

2011-08-29

Models and Methods

- 1 Boolean model and its limitations (30)
- 2 Vector space model (30)
- 3 Probabilistic models (30)
- 4 Language model-based retrieval (30)
- 5 Latent semantic indexing (30)
- 6 Learning to rank (30)

Take-away

Take-away

- Statistical language models: Introduction

Take-away

- Statistical language models: Introduction
- Statistical language models in IR

Take-away

- Statistical language models: Introduction
- Statistical language models in IR
- Discussion: Properties of different probabilistic models in use in IR

Outline

- 1 Statistical language models
- 2 Statistical language models in IR
- 3 Discussion

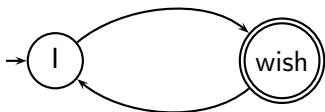
What is a language model?

What is a language model?

We can view a **finite state automaton** as a **deterministic** language model.

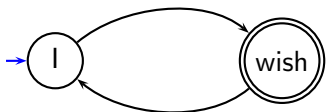
What is a language model?

We can view a **finite state automaton** as a **deterministic** language model.



What is a language model?

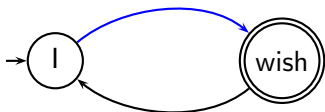
We can view a **finite state automaton** as a **deterministic** language model.



I

What is a language model?

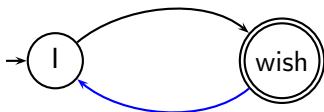
We can view a **finite state automaton** as a **deterministic** language model.



I wish

What is a language model?

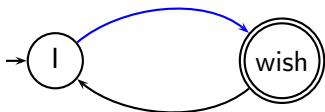
We can view a **finite state automaton** as a **deterministic** language model.



I wish I

What is a language model?

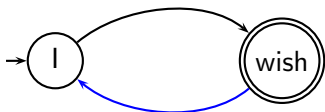
We can view a **finite state automaton** as a **deterministic** language model.



I wish I wish

What is a language model?

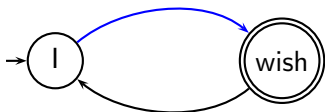
We can view a **finite state automaton** as a **deterministic** language model.



I wish I wish I

What is a language model?

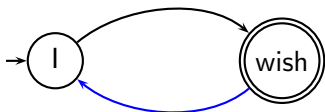
We can view a **finite state automaton** as a **deterministic** language model.



I wish I wish I wish

What is a language model?

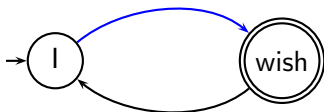
We can view a **finite state automaton** as a **deterministic** language model.



I wish I wish I wish I

What is a language model?

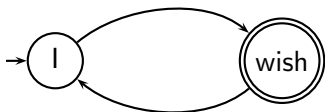
We can view a **finite state automaton** as a **deterministic** language model.



I wish I wish I wish I wish

What is a language model?

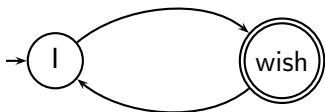
We can view a **finite state automaton** as a **deterministic** language model.



I wish I wish I wish I wish ...

What is a language model?

We can view a **finite state automaton** as a **deterministic** language model.

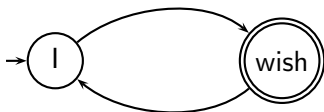


I wish I wish I wish I wish ...

Cannot generate: "wish I wish" or "I wish I"

What is a language model?

We can view a **finite state automaton** as a **deterministic** language model.



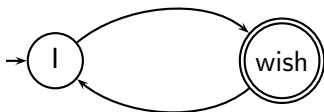
I wish I wish I wish I wish ...

Cannot generate: “wish I wish” or “I wish I”

Our basic model: each document was generated by a different automaton like this

What is a language model?

We can view a **finite state automaton** as a **deterministic** language model.



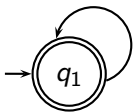
I wish I wish I wish I wish ...

Cannot generate: “wish I wish” or “I wish I”

Our basic model: each document was generated by a different automaton like this except that these automata are **probabilistic**.



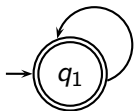
A probabilistic language model



w	$P(w q_1)$	w	$P(w q_1)$
STOP	0.2	toad	0.01
the	0.2	said	0.03
a	0.1	likes	0.02
frog	0.01	that	0.04
	

This is a one-state probabilistic finite-state automaton – a **unigram language model** – and the state emission distribution for its one state q_1 .

A probabilistic language model

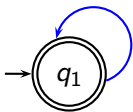


w	$P(w q_1)$	w	$P(w q_1)$
STOP	0.2	toad	0.01
the	0.2	said	0.03
a	0.1	likes	0.02
frog	0.01	that	0.04
	

This is a one-state probabilistic finite-state automaton – a **unigram language model** – and the state emission distribution for its one state q_1 .

STOP is not a word, but a special symbol indicating that the automaton stops.

A probabilistic language model



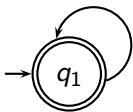
w	$P(w q_1)$	w	$P(w q_1)$
STOP	0.2	toad	0.01
the	0.2	said	0.03
a	0.1	likes	0.02
frog	0.01	that	0.04
	

This is a one-state probabilistic finite-state automaton – a **unigram language model** – and the state emission distribution for its one state q_1 .

STOP is not a word, but a special symbol indicating that the automaton stops.

frog

A probabilistic language model



w	$P(w q_1)$	w	$P(w q_1)$
STOP	0.2	toad	0.01
the	0.2	said	0.03
a	0.1	likes	0.02
frog	0.01	that	0.04
	

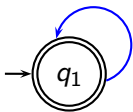
This is a one-state probabilistic finite-state automaton – a **unigram language model** – and the state emission distribution for its one state q_1 .

STOP is not a word, but a special symbol indicating that the automaton stops.

frog

$$P(\text{string}) = 0.01$$

A probabilistic language model



w	$P(w q_1)$	w	$P(w q_1)$
STOP	0.2	toad	0.01
the	0.2	said	0.03
a	0.1	likes	0.02
frog	0.01	that	0.04
	

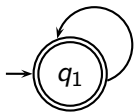
This is a one-state probabilistic finite-state automaton – a **unigram language model** – and the state emission distribution for its one state q_1 .

STOP is not a word, but a special symbol indicating that the automaton stops.

frog said

$$P(\text{string}) = 0.01$$

A probabilistic language model



w	$P(w q_1)$	w	$P(w q_1)$
STOP	0.2	toad	0.01
the	0.2	said	0.03
a	0.1	likes	0.02
frog	0.01	that	0.04
	

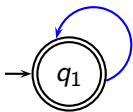
This is a one-state probabilistic finite-state automaton – a **unigram language model** – and the state emission distribution for its one state q_1 .

STOP is not a word, but a special symbol indicating that the automaton stops.

frog said

$$P(\text{string}) = 0.01 \cdot 0.03$$

A probabilistic language model



w	$P(w q_1)$	w	$P(w q_1)$
STOP	0.2	toad	0.01
the	0.2	said	0.03
a	0.1	likes	0.02
frog	0.01	that	0.04
	

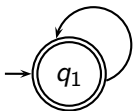
This is a one-state probabilistic finite-state automaton – a **unigram language model** – and the state emission distribution for its one state q_1 .

STOP is not a word, but a special symbol indicating that the automaton stops.

frog said that

$$P(\text{string}) = 0.01 \cdot 0.03$$

A probabilistic language model



w	$P(w q_1)$	w	$P(w q_1)$
STOP	0.2	toad	0.01
the	0.2	said	0.03
a	0.1	likes	0.02
frog	0.01	that	0.04
	

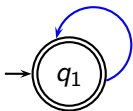
This is a one-state probabilistic finite-state automaton – a [unigram language model](#) – and the state emission distribution for its one state q_1 .

STOP is not a word, but a special symbol indicating that the automaton stops.

frog said that

$$P(\text{string}) = 0.01 \cdot 0.03 \cdot 0.04$$

A probabilistic language model



w	$P(w q_1)$	w	$P(w q_1)$
STOP	0.2	toad	0.01
the	0.2	said	0.03
a	0.1	likes	0.02
frog	0.01	that	0.04
	

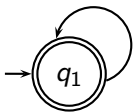
This is a one-state probabilistic finite-state automaton – a **unigram language model** – and the state emission distribution for its one state q_1 .

STOP is not a word, but a special symbol indicating that the automaton stops.

frog said that toad

$$P(\text{string}) = 0.01 \cdot 0.03 \cdot 0.04$$

A probabilistic language model



w	$P(w q_1)$	w	$P(w q_1)$
STOP	0.2	toad	0.01
the	0.2	said	0.03
a	0.1	likes	0.02
frog	0.01	that	0.04
	

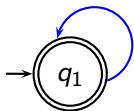
This is a one-state probabilistic finite-state automaton – a **unigram language model** – and the state emission distribution for its one state q_1 .

STOP is not a word, but a special symbol indicating that the automaton stops.

frog said that toad

$$P(\text{string}) = 0.01 \cdot 0.03 \cdot 0.04 \cdot 0.01$$

A probabilistic language model



w	$P(w q_1)$	w	$P(w q_1)$
STOP	0.2	toad	0.01
the	0.2	said	0.03
a	0.1	likes	0.02
frog	0.01	that	0.04
	

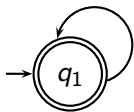
This is a one-state probabilistic finite-state automaton – a **unigram language model** – and the state emission distribution for its one state q_1 .

STOP is not a word, but a special symbol indicating that the automaton stops.

frog said that toad likes

$$P(\text{string}) = 0.01 \cdot 0.03 \cdot 0.04 \cdot 0.01$$

A probabilistic language model



w	$P(w q_1)$	w	$P(w q_1)$
STOP	0.2	toad	0.01
the	0.2	said	0.03
a	0.1	likes	0.02
frog	0.01	that	0.04
	

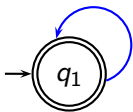
This is a one-state probabilistic finite-state automaton – a **unigram language model** – and the state emission distribution for its one state q_1 .

STOP is not a word, but a special symbol indicating that the automaton stops.

frog said that toad likes

$$P(\text{string}) = 0.01 \cdot 0.03 \cdot 0.04 \cdot 0.01 \cdot 0.02$$

A probabilistic language model



w	$P(w q_1)$	w	$P(w q_1)$
STOP	0.2	toad	0.01
the	0.2	said	0.03
a	0.1	likes	0.02
frog	0.01	that	0.04
	

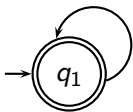
This is a one-state probabilistic finite-state automaton – a **unigram language model** – and the state emission distribution for its one state q_1 .

STOP is not a word, but a special symbol indicating that the automaton stops.

frog said that toad likes frog

$$P(\text{string}) = 0.01 \cdot 0.03 \cdot 0.04 \cdot 0.01 \cdot 0.02$$

A probabilistic language model



w	$P(w q_1)$	w	$P(w q_1)$
STOP	0.2	toad	0.01
the	0.2	said	0.03
a	0.1	likes	0.02
frog	0.01	that	0.04
	

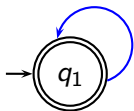
This is a one-state probabilistic finite-state automaton – a **unigram language model** – and the state emission distribution for its one state q_1 .

STOP is not a word, but a special symbol indicating that the automaton stops.

frog said that toad likes frog

$$P(\text{string}) = 0.01 \cdot 0.03 \cdot 0.04 \cdot 0.01 \cdot 0.02 \cdot 0.01$$

A probabilistic language model



w	$P(w q_1)$	w	$P(w q_1)$
STOP	0.2	toad	0.01
the	0.2	said	0.03
a	0.1	likes	0.02
frog	0.01	that	0.04
	

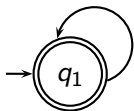
This is a one-state probabilistic finite-state automaton – a **unigram language model** – and the state emission distribution for its one state q_1 .

STOP is not a word, but a special symbol indicating that the automaton stops.

frog said that toad likes frog STOP

$$P(\text{string}) = 0.01 \cdot 0.03 \cdot 0.04 \cdot 0.01 \cdot 0.02 \cdot 0.01$$

A probabilistic language model



w	$P(w q_1)$	w	$P(w q_1)$
STOP	0.2	toad	0.01
the	0.2	said	0.03
a	0.1	likes	0.02
frog	0.01	that	0.04
	

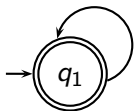
This is a one-state probabilistic finite-state automaton – a **unigram language model** – and the state emission distribution for its one state q_1 .

STOP is not a word, but a special symbol indicating that the automaton stops.

frog said that toad likes frog STOP

$$P(\text{string}) = 0.01 \cdot 0.03 \cdot 0.04 \cdot 0.01 \cdot 0.02 \cdot 0.01 \cdot 0.2$$

A probabilistic language model



w	$P(w q_1)$	w	$P(w q_1)$
STOP	0.2	toad	0.01
the	0.2	said	0.03
a	0.1	likes	0.02
frog	0.01	that	0.04
	

This is a one-state probabilistic finite-state automaton – a **unigram language model** – and the state emission distribution for its one state q_1 .

STOP is not a word, but a special symbol indicating that the automaton stops.

frog said that toad likes frog STOP

$$\begin{aligned}
 P(\text{string}) &= 0.01 \cdot 0.03 \cdot 0.04 \cdot 0.01 \cdot 0.02 \cdot 0.01 \cdot 0.2 \\
 &= 0.00000000000048
 \end{aligned}$$



A different language model for each document

language model of d_1

w	$P(w .)$	w	$P(w .)$
STOP	.2	toad	.01
the	.2	said	.03
a	.1	likes	.02
frog	.01	that	.04
	

language model of d_2

w	$P(w .)$	w	$P(w .)$
STOP	.2	toad	.02
the	.15	said	.03
a	.08	likes	.02
frog	.01	that	.05
	

query: frog said that toad likes frog STOP

A different language model for each document

language model of d_1

w	$P(w .)$	w	$P(w .)$
STOP	.2	toad	.01
the	.2	said	.03
a	.1	likes	.02
frog	.01	that	.04
	

language model of d_2

w	$P(w .)$	w	$P(w .)$
STOP	.2	toad	.02
the	.15	said	.03
a	.08	likes	.02
frog	.01	that	.05
	

query: frog said that toad likes frog STOP

$$\begin{aligned}
 P(\text{query}|M_{d_1}) &= 0.01 \cdot 0.03 \cdot 0.04 \cdot 0.01 \cdot 0.02 \cdot 0.01 \cdot 0.2 \\
 &= 0.00000000000048 = 4.8 \cdot 10^{-12}
 \end{aligned}$$

A different language model for each document

language model of d_1

w	$P(w .)$	w	$P(w .)$
STOP	.2	toad	.01
the	.2	said	.03
a	.1	likes	.02
frog	.01	that	.04
	

language model of d_2

w	$P(w .)$	w	$P(w .)$
STOP	.2	toad	.02
the	.15	said	.03
a	.08	likes	.02
frog	.01	that	.05
	

query: frog said that toad likes frog STOP

$$P(\text{query}|M_{d_1}) = 0.01 \cdot 0.03 \cdot 0.04 \cdot 0.01 \cdot 0.02 \cdot 0.01 \cdot 0.2$$

$$= 0.00000000000048 = 4.8 \cdot 10^{-12}$$

$$P(\text{query}|M_{d_2}) = 0.01 \cdot 0.03 \cdot 0.05 \cdot 0.02 \cdot 0.02 \cdot 0.01 \cdot 0.2$$

$$= 0.00000000000120 = 12 \cdot 10^{-12}$$

A different language model for each document

language model of d_1

w	$P(w .)$	w	$P(w .)$
STOP	.2	toad	.01
the	.2	said	.03
a	.1	likes	.02
frog	.01	that	.04
	

language model of d_2

w	$P(w .)$	w	$P(w .)$
STOP	.2	toad	.02
the	.15	said	.03
a	.08	likes	.02
frog	.01	that	.05
	

query: frog said that toad likes frog STOP

$$P(\text{query}|M_{d_1}) = 0.01 \cdot 0.03 \cdot 0.04 \cdot 0.01 \cdot 0.02 \cdot 0.01 \cdot 0.2$$

$$= 0.0000000000048 = 4.8 \cdot 10^{-12}$$

$$P(\text{query}|M_{d_2}) = 0.01 \cdot 0.03 \cdot 0.05 \cdot 0.02 \cdot 0.02 \cdot 0.01 \cdot 0.2$$

$$= 0.0000000000120 = 12 \cdot 10^{-12}$$

$P(\text{query}|M_{d_1}) < P(\text{query}|M_{d_2})$ Thus, document d_2 is “more relevant” to the query “frog said that toad likes frog STOP” than d_1 is. □

Outline

- 1 Statistical language models
- 2 Statistical language models in IR
- 3 Discussion

Using language models in IR

- Each document is treated as (the basis for) a language model.

Using language models in IR

- Each document is treated as (the basis for) a language model.
- Given a query q

Using language models in IR

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

Using language models in IR

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

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

Using language models in IR

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



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

- $P(q)$ is the same for all documents, so ignore

Using language models in IR

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



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

- $P(q)$ is the same for all documents, so ignore
- $P(d)$ is the prior – often treated as the same for all d

Using language models in IR

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



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

- $P(q)$ 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, e.g., those with high PageRank.

Using language models in IR

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



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

- $P(q)$ 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, e.g., those with high PageRank.
- $P(q|d)$ is the probability of q given d .

Using language models in IR

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



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

- $P(q)$ 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, e.g., those with high PageRank.
- $P(q|d)$ is the probability of q given d .
- Under the assumptions we made, ranking documents according to $P(q|d)P(d)$ and $P(d|q)$ is equivalent.



How to compute $P(q|d)$

How to compute $P(q|d)$

- We will make the same conditional independence assumption as in BIM.

How to compute $P(q|d)$

- We will make the same conditional independence assumption as in BIM.
-

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

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

How to compute $P(q|d)$

- We will make the same conditional independence assumption as in BIM.



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

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

- This is equivalent to:

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

How to compute $P(q|d)$

- We will make the same conditional independence assumption as in BIM.



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

($|q|$: length of 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

How to compute $P(q|d)$

- We will make the same conditional independence assumption as in BIM.



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

($|q|$: length of 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
- **Multinomial model** (omitting constant factor)



Parameter estimation

- Missing piece: Where do the parameters $P(t|M_d)$ come from?

Parameter estimation

- Missing piece: Where do the parameters $P(t|M_d)$ come from?
- Start with maximum likelihood estimates

Parameter estimation

- Missing piece: Where do the parameters $P(t|M_d)$ come from?
- Start with maximum likelihood estimates
-

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

($|d|$: length of d ; $\text{tf}_{t,d}$: # occurrences of t in d)

Parameter estimation

- Missing piece: Where do the parameters $P(t|M_d)$ come from?
- Start with maximum likelihood estimates

-

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

($|d|$: length of d ; $\text{tf}_{t,d}$: # occurrences of t in d)

- We have a problem with zeros.

Parameter estimation

- Missing piece: Where do the parameters $P(t|M_d)$ come from?
- Start with maximum likelihood estimates



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

($|d|$: length of d ; $\text{tf}_{t,d}$: # occurrences of t in d)

- We have a problem with zeros.
- A single t in the query with $P(t|M_d) = 0$ will make $P(q|M_d) = \prod P(t|M_d)$ zero.

Parameter estimation

- Missing piece: Where do the parameters $P(t|M_d)$ come from?
- Start with maximum likelihood estimates

-

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

($|d|$: length of d ; $\text{tf}_{t,d}$: # occurrences of t in d)

- We have a problem with zeros.
- A single t in the query with $P(t|M_d) = 0$ will make $P(q|M_d) = \prod P(t|M_d)$ zero.
- We would give a single term in the query “veto power”.

Parameter estimation

- Missing piece: Where do the parameters $P(t|M_d)$ come from?
- Start with maximum likelihood estimates

-

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

($|d|$: length of d ; $\text{tf}_{t,d}$: # occurrences of t in d)

- We have a problem with zeros.
- A single t in the query with $P(t|M_d) = 0$ will make $P(q|M_d) = \prod P(t|M_d)$ zero.
- We would give a single term in the query “veto power”.
- For example, for query [Michael Jackson top hits] a document about “Michael Jackson top songs” (but not using the word “hits”) would have $P(q|M_d) = 0$. – That’s bad.

Parameter estimation

- Missing piece: Where do the parameters $P(t|M_d)$ come from?
- Start with maximum likelihood estimates

-

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

($|d|$: length of d ; $\text{tf}_{t,d}$: # occurrences of t in d)

- We have a problem with zeros.
- A single t in the query with $P(t|M_d) = 0$ will make $P(q|M_d) = \prod P(t|M_d)$ zero.
- We would give a single term in the query “veto power”.
- For example, for query [Michael Jackson top hits] a document about “Michael Jackson top songs” (but not using the word “hits”) would have $P(q|M_d) = 0$. – That’s bad.
- We need to smooth the estimates to avoid zeros. □

Smoothing

- Key intuition: A nonoccurring term is possible (even though it didn't occur), ...

Smoothing

- Key intuition: A nonoccurring term is possible (even though it didn't occur), ...
- ...but no more likely than would be expected by chance in the collection.

Smoothing

- Key intuition: A nonoccurring term is possible (even though it didn't occur), ...
- ...but no more likely than would be expected by chance in the collection.
- Notation: M_C : the collection model; cf_t : the number of occurrences of t in the collection; $T = \sum_t cf_t$: the total number of tokens in the collection.

Smoothing

- Key intuition: A nonoccurring term is possible (even though it didn't occur), ...
- ...but no more likely than would be expected by chance in the collection.
- Notation: M_C : the collection model; cf_t : the number of occurrences of t in the collection; $T = \sum_t cf_t$: the total number of tokens in the collection.



$$\hat{P}(t|M_C) = \frac{cf_t}{T}$$

Smoothing

- Key intuition: A nonoccurring term is possible (even though it didn't occur), ...
- ...but no more likely than would be expected by chance in the collection.
- Notation: M_c : the collection model; cf_t : the number of occurrences of t in the collection; $T = \sum_t cf_t$: the total number of tokens in the collection.



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

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

Jelinek-Mercer smoothing

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

Jelinek-Mercer smoothing

- $P(t|d) = \lambda P(t|M_d) + (1 - \lambda)P(t|M_c)$
- Mixes the probability from the document with the general collection frequency of the word.

Jelinek-Mercer smoothing

- $P(t|d) = \lambda P(t|M_d) + (1 - \lambda)P(t|M_c)$
- Mixes the probability from the document with the general collection frequency of the word.
- High value of λ : “conjunctive-like” search – tends to retrieve documents containing all query words.

Jelinek-Mercer smoothing

- $P(t|d) = \lambda P(t|M_d) + (1 - \lambda)P(t|M_c)$
- Mixes the probability from the document with the general collection frequency of the word.
- High value of λ : “conjunctive-like” search – tends to retrieve documents containing all query words.
- Low value of λ : more disjunctive, suitable for long queries

Jelinek-Mercer smoothing

- $P(t|d) = \lambda P(t|M_d) + (1 - \lambda)P(t|M_c)$
- Mixes the probability from the document with the general collection frequency of the word.
- High value of λ : “conjunctive-like” search – tends to retrieve documents containing all query words.
- Low value of λ : more disjunctive, suitable for long queries
- Tuning λ is important for good performance. □

Jelinek-Mercer smoothing: Summary



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

Jelinek-Mercer smoothing: 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.

Jelinek-Mercer smoothing: 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.
- $P(q|d)$ is the probability that the document that the user had in mind was in fact this one. □

Example

- Collection: d_1 and d_2

Example

- Collection: d_1 and d_2
- d_1 : Jackson was one of the most talented entertainers of all time

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

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

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 $\lambda = 1/2$

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 $\lambda = 1/2$
- $P(q|d_1) = [(0/11 + 1/18)/2] \cdot [(1/11 + 2/18)/2] \approx 0.003$

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 $\lambda = 1/2$
- $P(q|d_1) = [(0/11 + 1/18)/2] \cdot [(1/11 + 2/18)/2] \approx 0.003$
- $P(q|d_2) = [(1/7 + 1/18)/2] \cdot [(1/7 + 2/18)/2] \approx 0.013$

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: $d_2 > d_1$



Dirichlet smoothing

Dirichlet smoothing



$$P(t|d) = \frac{tf_{t,d} + \alpha P(t|M_c)}{L_d + \alpha}$$

Dirichlet smoothing



$$P(t|d) = \frac{tf_{t,d} + \alpha P(t|M_c)}{L_d + \alpha}$$

- The background distribution $P(t|M_c)$ is the prior for $P(t|d)$.

Dirichlet smoothing



$$P(t|d) = \frac{tf_{t,d} + \alpha P(t|M_c)}{L_d + \alpha}$$

- The background distribution $P(t|M_c)$ is the prior for $P(t|d)$.
- Intuition: Before having seen any part of the document we start with the background distribution as our estimate.

Dirichlet smoothing



$$P(t|d) = \frac{tf_{t,d} + \alpha P(t|M_c)}{L_d + \alpha}$$

- The background distribution $P(t|M_c)$ is the prior for $P(t|d)$.
- Intuition: Before having seen any part of the document we start with the background distribution as our estimate.
- As we read the document and count terms we update the background distribution.

Dirichlet smoothing



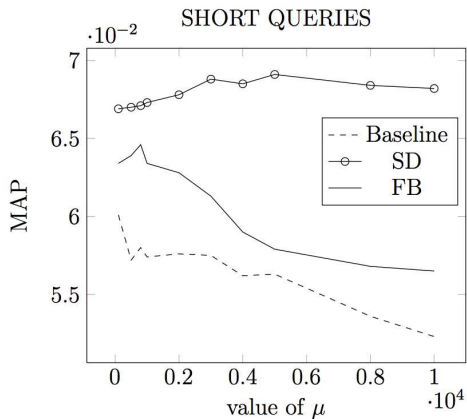
$$P(t|d) = \frac{tf_{t,d} + \alpha P(t|M_c)}{L_d + \alpha}$$

- The background distribution $P(t|M_c)$ is the prior for $P(t|d)$.
- Intuition: Before having seen any part of the document we start with the background distribution as our estimate.
- As we read the document and count terms we update the background distribution.
- The weighting factor α determines how strong an effect the prior has. □

Jelinek-Mercer or Dirichlet?

- Dirichlet performs better for keyword queries, Jelinek-Mercer performs better for verbose queries.
- Both models are sensitive to the smoothing parameters – you shouldn't use these models without parameter tuning.

Sensitivity of Dirichlet to smoothing parameter



μ is the Dirichlet smoothing parameter (called α on the previous slides)



Vector space (tf-idf) vs. LM

Rec.	precision		%chg	significant
	tf-idf	LM		
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	*

Vector space (tf-idf) vs. LM

Rec.	precision		%chg	significant
	tf-idf	LM		
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 ...

Vector space (tf-idf) vs. LM

Rec.	tf-idf	precision		significant
		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 note that where the approach shows significant gains is at higher levels of recall. □

Summary: IR language models

Summary: IR language models

- 1 View the document as a generative model that generates the query

Summary: IR language models

- 1 View the document as a generative model that generates the query
- 2 Define the precise generative model we want to use

Summary: IR language models

- 1 View the document as a generative model that generates the query
- 2 Define the precise generative model we want to use
- 3 Estimate parameters (different parameters for each document's model)

Summary: IR language models

- 1 View the document as a generative model that generates the query
- 2 Define the precise generative model we want to use
- 3 Estimate parameters (different parameters for each document's model)
- 4 Smooth to avoid zeros

Summary: IR language models

- 1 View the document as a generative model that generates the query
- 2 Define the precise generative model we want to use
- 3 Estimate parameters (different parameters for each document's model)
- 4 Smooth to avoid zeros
- 5 Apply to query and find document most likely to have generated the query

Summary: IR language models

- 1 View the document as a generative model that generates the query
- 2 Define the precise generative model we want to use
- 3 Estimate parameters (different parameters for each document's model)
- 4 Smooth to avoid zeros
- 5 Apply to query and find document most likely to have generated the query
- 6 Present most likely document(s) to user



Outline

- 1 Statistical language models
- 2 Statistical language models in IR
- 3 Discussion

Naive Bayes

generative model

- We want to classify document d .

Naive Bayes

generative model

- We want to classify document d .
 - Human-defined classes: e.g., politics, economics, sports.

Naive Bayes

generative model

- We want to classify document d .
 - Human-defined classes: e.g., politics, economics, sports.
- Assume that d was generated by the generative model.

Naive Bayes

generative model

- We want to classify document d .
 - Human-defined classes: e.g., politics, economics, sports.
- Assume that d was generated by the generative model.
- Key question: Which of the classes (= class models) is most likely to have generated the document?

Naive Bayes

generative model

- We want to classify document d .
 - Human-defined classes: e.g., politics, economics, sports.
- Assume that d was generated by the generative model.
- Key question: Which of the classes (= class models) is most likely to have generated the document?
 - Or: for which class do we have the most evidence?

Naive Bayes and LM generative models

- We want to classify document d .
 - Human-defined classes: e.g., politics, economics, sports.
- Assume that d was generated by the generative model.
- Key question: Which of the classes (= class models) is most likely to have generated the document?
 - Or: for which class do we have the most evidence?

Naive Bayes and LM generative models

- We want to classify document d .
We want to classify a query q .
 - Human-defined classes: e.g., politics, economics, sports.
- Assume that d was generated by the generative model.
- Key question: Which of the classes (= class models) is most likely to have generated the document?
 - Or: for which class do we have the most evidence?

Naive Bayes and LM generative models

- We want to classify document d .
We want to classify a query q .
 - Human-defined classes: e.g., politics, economics, sports.
Each document in the collection is a different class.
- Assume that d was generated by the generative model.
- Key question: Which of the classes (= class models) is most likely to have generated the document?
 - Or: for which class do we have the most evidence?

Naive Bayes and LM generative models

- We want to classify document d .
We want to classify a query q .
 - Human-defined classes: e.g., politics, economics, sports.
Each document in the collection is a different class.
- Assume that d was generated by the generative model.
Assume that q was generated by a generative model
- Key question: Which of the classes (= class models) is most likely to have generated the document?
 - Or: for which class do we have the most evidence?

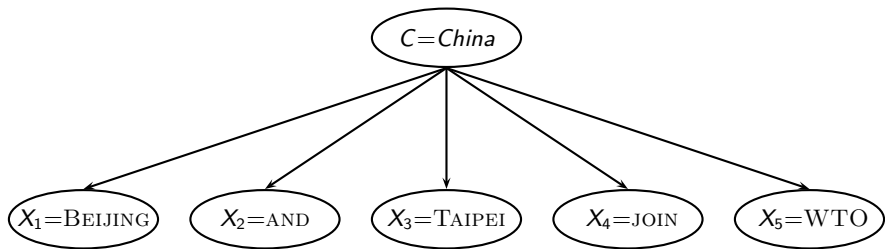
Naive Bayes and LM generative models

- We want to classify document d .
We want to classify a query q .
 - Human-defined classes: e.g., politics, economics, sports.
Each document in the collection is a different class.
- Assume that d was generated by the generative model.
Assume that q was generated by a generative model
- Key question: Which of the classes (= class models) is most likely to have generated the document? Which document (=class) is most likely to have generated the query q ?
 - Or: for which class do we have the most evidence?

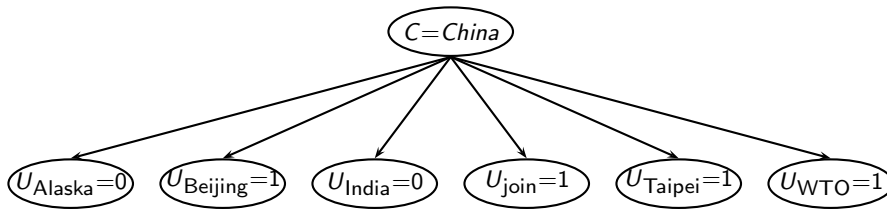
Naive Bayes and LM generative models

- We want to classify document d .
We want to classify a query q .
 - Human-defined classes: e.g., politics, economics, sports.
Each document in the collection is a different class.
- Assume that d was generated by the generative model.
Assume that q was generated by a generative model
- Key question: Which of the classes (= class models) is most likely to have generated the document? Which document (=class) is most likely to have generated the query q ?
 - Or: for which class do we have the most evidence? For which document (as the source of the query) do we have the most evidence? □

Naive Bayes Multinomial model / IR language models



Naive Bayes Bernoulli model / Binary independence model



Comparison of the two models

	multinomial model / IR LM	Bernoulli model / BIM
event model	generation of (multi)set of tokens	generation of subset of vocabulary
random variable(s)	$X = t$ iff t occurs at given pos	$U_t = 1$ iff t occurs in doc
doc. representation	$d = \langle t_1, \dots, t_k, \dots, t_{n_d} \rangle, t_k \in V$	$d = \langle e_1, \dots, e_i, \dots, e_M \rangle,$ $e_i \in \{0, 1\}$
parameter estimation	$\hat{P}(X = t c)$	$\hat{P}(U_i = e c)$
dec. rule: maximize	$\hat{P}(c) \prod_{1 \leq k \leq n_d} \hat{P}(X = t_k c)$	$\hat{P}(c) \prod_{t_i \in V} \hat{P}(U_i = e_i c)$
multiple occurrences	taken into account	ignored
length of docs	can handle longer docs	works best for short docs
# features	can handle more	works best with fewer
estimate for THE	$\hat{P}(X = \text{the} c) \approx 0.05$	$\hat{P}(U_{\text{the}} = 1 c) \approx 1.0$

Vector space vs BM25 vs LM

Vector space vs BM25 vs LM

- BM25/LM: based on probability theory

Vector space vs BM25 vs LM

- BM25/LM: based on probability theory
- Vector space: based on similarity, a geometric/linear algebra notion

Vector space vs BM25 vs LM

- BM25/LM: based on probability theory
- Vector space: based on similarity, a geometric/linear algebra notion
- Term frequency is directly used in all three models.

Vector space vs BM25 vs LM

- BM25/LM: based on probability theory
- Vector space: based on similarity, a geometric/linear algebra notion
- Term frequency is directly used in all three models.
 - LMs: raw term frequency, BM25/Vector space: more complex

Vector space vs BM25 vs LM

- BM25/LM: based on probability theory
- Vector space: based on similarity, a geometric/linear algebra notion
- Term frequency is directly used in all three models.
 - LMs: raw term frequency, BM25/Vector space: more complex
- Length normalization

Vector space vs BM25 vs LM

- BM25/LM: based on probability theory
- Vector space: based on similarity, a geometric/linear algebra notion
- Term frequency is directly used in all three models.
 - LMs: raw term frequency, BM25/Vector space: more complex
- Length normalization
 - Vector space: Cosine or pivot normalization

Vector space vs BM25 vs LM

- BM25/LM: based on probability theory
- Vector space: based on similarity, a geometric/linear algebra notion
- Term frequency is directly used in all three models.
 - LMs: raw term frequency, BM25/Vector space: more complex
- Length normalization
 - Vector space: Cosine or pivot normalization
 - LMs: probabilities are inherently length normalized

Vector space vs BM25 vs LM

- BM25/LM: based on probability theory
- Vector space: based on similarity, a geometric/linear algebra notion
- Term frequency is directly used in all three models.
 - LMs: raw term frequency, BM25/Vector space: more complex
- Length normalization
 - Vector space: Cosine or pivot normalization
 - LMs: probabilities are inherently length normalized
 - BM25: tuning parameters for optimizing length normalization

Vector space vs BM25 vs LM

- BM25/LM: based on probability theory
- Vector space: based on similarity, a geometric/linear algebra notion
- Term frequency is directly used in all three models.
 - LMs: raw term frequency, BM25/Vector space: more complex
- Length normalization
 - Vector space: Cosine or pivot normalization
 - LMs: probabilities are inherently length normalized
 - BM25: tuning parameters for optimizing length normalization
- idf: BM25/vector space use it directly.

Vector space vs BM25 vs LM

- BM25/LM: based on probability theory
- Vector space: based on similarity, a geometric/linear algebra notion
- Term frequency is directly used in all three models.
 - LMs: raw term frequency, BM25/Vector space: more complex
- Length normalization
 - Vector space: Cosine or pivot normalization
 - LMs: probabilities are inherently length normalized
 - BM25: tuning parameters for optimizing length normalization
- idf: BM25/vector space use it directly.
- LMs: Mixing term and collection frequencies has an effect similar to idf.

Vector space vs BM25 vs LM

- BM25/LM: based on probability theory
- Vector space: based on similarity, a geometric/linear algebra notion
- Term frequency is directly used in all three models.
 - LMs: raw term frequency, BM25/Vector space: more complex
- Length normalization
 - Vector space: Cosine or pivot normalization
 - LMs: probabilities are inherently length normalized
 - BM25: tuning parameters for optimizing length normalization
- idf: BM25/vector space use it directly.
- LMs: Mixing term 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.

Vector space vs BM25 vs LM

- BM25/LM: based on probability theory
- Vector space: based on similarity, a geometric/linear algebra notion
- Term frequency is directly used in all three models.
 - LMs: raw term frequency, BM25/Vector space: more complex
- Length normalization
 - Vector space: Cosine or pivot normalization
 - LMs: probabilities are inherently length normalized
 - BM25: tuning parameters for optimizing length normalization
- idf: BM25/vector space use it directly.
- LMs: Mixing term 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.
- Collection frequency (LMs) vs. document frequency (BM25, vector space)



Take-away

- **Statistical language models:** Introduction
- **Statistical language models in IR**
- **Discussion:** Properties of different probabilistic models in use in IR

Resources

- Chapter 12 of Introduction to Information Retrieval
- Resources at <http://informationretrieval.org/essir2011>
 - Ponte and Croft's 1998 SIGIR paper (one of the first on LMs in IR)
 - Zhai and Lafferty: A study of smoothing methods for language models applied to information retrieval. ACM Trans. Inf. Syst. (2004).
 - Lemur toolkit (good support for LMs in IR)
 - Bernoulli vs multinomial models

Exercise: Compute ranking

- Collection: d_1 and d_2
- d_1 : Xerox reports a profit but revenue is down
- d_2 : Lucene narrows quarter loss but revenue decreases further
- Query q : revenue down
- Use mixture model with $\lambda = 1/2$

Exercise: Compute ranking

- Collection: d_1 and d_2
- d_1 : Xerox reports a profit but revenue is down
- d_2 : Lucene narrows quarter loss but revenue decreases further
- Query q : revenue down
- Use mixture model with $\lambda = 1/2$

Exercise: Compute ranking

- Collection: d_1 and d_2
- d_1 : Xerox reports a profit but revenue is down
- d_2 : Lucene narrows quarter loss but revenue decreases further
- Query q : revenue down
- Use mixture model with $\lambda = 1/2$

Exercise: Compute ranking

- Collection: d_1 and d_2
- d_1 : Xerox reports a profit but revenue is down
- d_2 : Lucene narrows quarter loss but revenue decreases further
- Query q : revenue down
- Use mixture model with $\lambda = 1/2$

Exercise: Compute ranking

- Collection: d_1 and d_2
- d_1 : Xerox reports a profit but revenue is down
- d_2 : Lucene narrows quarter loss but revenue decreases further
- Query q : revenue down
- Use mixture model with $\lambda = 1/2$
- $P(q|d_1) = [(1/8 + 2/16)/2] \cdot [(1/8 + 1/16)/2] = 1/8 \cdot 3/32 = 3/256$
- $P(q|d_2) = [(1/8 + 2/16)/2] \cdot [(0/8 + 1/16)/2] = 1/8 \cdot 1/32 = 1/256$
- Ranking: $d_1 > d_2$