Lecture 5: Language Modelling in Information Retrieval and Classification

Information Retrieval
Computer Science Tripos Part II

Ronan Cummins[1]

Natural Language and Information Processing (NLIP) Group
UNIVERSITY OF CAMBRIDGE

ronan.cummins@cl.cam.ac.uk

[1]Adapted from Simone Teufel's original slides

Upcoming today

- Query-likelihood method in IR

Upcoming today

- Query-likelihood method in IR
- Document Language Modelling

Upcoming today

- Query-likelihood method in IR
- Document Language Modelling
- Smoothing

Upcoming today

- Query-likelihood method in IR
- Document Language Modelling
- Smoothing
- Classification

Overview

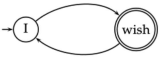
1. Query Likelihood
2. Estimating Document Models
3. Smoothing
4. Naive Bayes Classification

Language Model

- A model for how humans generate language
- Used in many language orientated-tasks (MT, word prediction, IR)
- Usually probabilistic in nature (e.g. multinomial, neural)

Language Model

- A model for how humans generate language
- Used in many language orientated-tasks (MT, word prediction, IR)
- Usually probabilistic in nature (e.g. multinomial, neural)



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

CANNOT GENERATE: wish I wish

▶ **Figure 12.1** A simple finite automaton and some of the strings in the language it generates. → shows the start state of the automaton and a double circle indicates a (possible) finishing state.

What is a document language model?

- A model for how an author generates a document on a particular topic
- The document itself is just one sample from the model (i.e. ask the author to write the document again and he/she will invariably write something similar, but not exactly the same)
- A probabilistic generative model for documents

Two Document Models

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

► **Figure 12.3** Partial specification of two unigram language models.

$$\sum_{t \in V} P(t|M_d) = 1 \tag{1}$$

Query Likelihood Method (I)

- Users often pose queries by thinking of words that are likely to be in *relevant* documents
- The query likelihood approach uses this idea as a principle for ranking documents
- Given a query string q, we rank documents by the likelihood of their document *models* M_d generating q

Query Likelihood Method (II)

$$P(d|q) = P(q|d)P(d)/P(q) \tag{2}$$

Query Likelihood Method (II)

$$P(d|q) = P(q|d)P(d)/P(q) \tag{2}$$

$$P(d|q) \propto P(q|d)P(d) \tag{3}$$

Query Likelihood Method (II)

$$P(d|q) = P(q|d)P(d)/P(q) \tag{2}$$

$$P(d|q) \propto P(q|d)P(d) \tag{3}$$

where if we have a uniform prior over $P(d)$ then

$$P(d|q) \propto P(q|d) \tag{4}$$

Query Likelihood Method (II)

$$P(d|q) = P(q|d)P(d)/P(q) \qquad (2)$$

$$P(d|q) \propto P(q|d)P(d) \qquad (3)$$

where if we have a uniform prior over $P(d)$ then

$$P(d|q) \propto P(q|d) \qquad (4)$$

Note: $P(d)$ is uniform if we have no reason a priori to favour one document over another. Useful priors (based on aspects such as authority, length, novelty, freshness, popularity, click-through rate) could easily be incorporated.

An Example (I)

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

► **Figure 12.3** Partial specification of two unigram language models.

$$P(frog\ said\ that\ toad\ likes\ frog|M_1) = \tag{5}$$

$$\tag{6}$$

An Example (I)

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

▶ **Figure 12.3** Partial specification of two unigram language models.

$$P(\textit{frog said that toad likes frog} \mid M_1) = (0.01 \times 0.03 \times 0.04 \times 0.01 \times 0.02 \times 0.01) \quad (5)$$

$$(6)$$

An Example (I)

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

▶ **Figure 12.3** Partial specification of two unigram language models.

$$P(\mathit{frog\ said\ that\ toad\ likes\ frog}|M_1) = (0.01 \times 0.03 \times 0.04 \times 0.01 \times 0.02 \times 0.01) \quad (5)$$

$$P(\mathit{frog\ said\ that\ toad\ likes\ frog}|M_2) = \quad (6)$$

An Example (I)

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

► **Figure 12.3** Partial specification of two unigram language models.

$$P(\mathit{frog\ said\ that\ toad\ likes\ frog}|M_1) = (0.01 \times 0.03 \times 0.04 \times 0.01 \times 0.02 \times 0.01) \quad (5)$$

$$P(\mathit{frog\ said\ that\ toad\ likes\ frog}|M_2) = (0.0002 \times 0.03 \times 0.04 \times 0.0001 \times 0.04 \times 0.0002) \quad (6)$$

An Example (II)

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

► **Figure 12.3** Partial specification of two unigram language models.

$$P(q|M_1) > P(q|M_2) \tag{7}$$

Overview

1. Query Likelihood
2. Estimating Document Models
3. Smoothing
4. Naive Bayes Classification

Documents as samples

- We now know how to rank document models in a theoretically principled manner.
- But how do we estimate the document model for each document?

Documents as samples

- We now know how to rank document models in a theoretically principled manner.
- But how do we estimate the document model for each document?

document 1

click go the shears boys click click click

Documents as samples

- We now know how to rank document models in a theoretically principled manner.
- But how do we estimate the document model for each document?

document 1

click go the shears boys click click click

Maximum likelihood estimates

click=0.5, go=0.125, the=0.125, shears=0.125, boys=0.125,

Zero probability problem (over-fitting)

- When using maximum likelihood estimates, documents that do not contain *all* query terms will receive a score of zero

Zero probability problem (over-fitting)

- When using maximum likelihood estimates, documents that do not contain *all* query terms will receive a score of zero

Maximum likelihood estimates

click=0.5, go=0.125, the=0.125, shears=0.125, boys=0.125

Zero probability problem (over-fitting)

- When using maximum likelihood estimates, documents that do not contain *all* query terms will receive a score of zero

Maximum likelihood estimates

click=0.5, go=0.125, the=0.125, shears=0.125, boys=0.125

Sample query

$P(shears\ boys\ hair|M_d) =$

Zero probability problem (over-fitting)

- When using maximum likelihood estimates, documents that do not contain *all* query terms will receive a score of zero

Maximum likelihood estimates

click=0.5, go=0.125, the=0.125, shears=0.125, boys=0.125

Sample query

$P(shears\ boys\ hair|M_d) = 0.0$

What if the query is long?

Make sure no non-zero probabilities

- Only assign a zero probability when something *cannot* happen
- Remember that the document model is a generative explanation
- If a person was to rewrite the document he/she may include *hair* or indeed some other words

Make sure no non-zero probabilities

- Only assign a zero probability when something *cannot* happen
- Remember that the document model is a generative explanation
- If a person was to rewrite the document he/she may include *hair* or indeed some other words

Maximum likelihood estimates

click=0.5, go=0.125, the=0.125, shears=0.125, boys=0.125

Make sure no non-zero probabilities

- Only assign a zero probability when something *cannot* happen
- Remember that the document model is a generative explanation
- If a person was to rewrite the document he/she may include *hair* or indeed some other words

Maximum likelihood estimates

click=0.5, go=0.125, the=0.125, shears=0.125, boys=0.125

Some type of smoothing

click=0.4, go=0.1, the=0.1, shears=0.1, boys=0.1, hair=0.01, man=0.01, the=0.001, bacon=0.0001, ...

Overview

How to smooth

ML estimates

$$\hat{P}(t|M_d) = \frac{tf_t}{|d|} \tag{8}$$

Maximum likelihood estimates

click=0.5, go=0.125, the=0.125, shears=0.125, boys=0.125

How to smooth

ML estimates

$$\hat{P}(t|M_d) = \frac{tf_t}{|d|} \tag{8}$$

Maximum likelihood estimates

click=0.5, go=0.125, the=0.125, shears=0.125, boys=0.125

Linear Smoothing

$$\hat{P}(t|M_d) = \lambda \frac{tf_t}{|d|} + (1-\lambda)\hat{P}(t|M_c) \tag{9}$$

where λ is a smoothing parameter between 0 and 1, and $\hat{P}(t|M_c) = \frac{cf_t}{|c|}$ is the estimated probability of seeing t in general (i.e. ct_t is the frequency of t in the entire document collection of $|c|$ tokens).

Linear Smoothing

$$\hat{P}(t|M_d) = \lambda \frac{tf_t}{|d|} + (1 - \lambda) \frac{cf_t}{|c|} \tag{10}$$

How to smooth

Linear Smoothing

$$\hat{P}(t|M_d) = \lambda \frac{tf_t}{|d|} + (1 - \lambda)\frac{cf_t}{|c|} \tag{10}$$

Dirichlet Smoothing has been found to be more effective in IR where λ is $\frac{|d|}{\alpha+|d|}$. Plugging this in yields:

$$\hat{P}(t|M_d) = \frac{|d|}{\alpha + |d|}\frac{tf_t}{|d|} + \frac{\alpha}{\alpha + |d|}\frac{cf_t}{|c|} \tag{11}$$

where α is interpreted as the background mass (pseudo-counts).

Bayesian Intuition

We should have more trust (belief) in ML estimates that are derived from longer documents (see the $\frac{|d|}{\alpha+|d|}$ factor).

Putting this all together

Rank documents according to:

$$P(q|d) = \prod_{t \in q} \left(\frac{|d|}{\alpha + |d|} \frac{tf_t}{|d|} + \frac{\alpha}{\alpha + |d|} \frac{cf_t}{|c|} \right) \qquad (12)$$

or

Putting this all together

Rank documents according to:

$$P(q|d) = \prod_{t \in q} \left(\frac{|d|}{\alpha + |d|} \frac{tf_t}{|d|} + \frac{\alpha}{\alpha + |d|} \frac{cf_t}{|c|} \right) \tag{12}$$

or

$$log\ P(q|d) = \sum_{t \in q} log \left(\frac{|d|}{\alpha + |d|} \frac{tf_t}{|d|} + \frac{\alpha}{\alpha + |d|} \frac{cf_t}{|c|} \right) \tag{13}$$

Pros and Cons

- It is principled, intuitive, simple, and extendable
- Aspects of *tf* and *idf* are incorporated quite naturally
- It is computationally efficient for large scale corpora
- More complex language models (markov-models) can be adopted and priors can be added
- But more complex models usually involve storing more parameters (and doing more computation)

Pros and Cons

- It is principled, intuitive, simple, and extendable
- Aspects of *tf* and *idf* are incorporated quite naturally
- It is computationally efficient for large scale corpora
- More complex language models (markov-models) can be adopted and priors can be added
- But more complex models usually involve storing more parameters (and doing more computation)

- Both documents and queries are modelled as simple strings of symbols
- No formal treatment of relevance
- Therefore model does not handle relevance feedback automatically

Extensions

- Relevance-based language models (very much related to Naive-Bayes classification) incorporate the idea of relevance and are useful for capturing feedback
- Treating the query as being drawn from a query model (useful for long queries)
- Markov-chain models for document modelling
- Use different generative distributions (e.g. replacing the multinomial with neural models)

Overview

The Naive Bayes classifier

- The Naive Bayes classifier is a probabilistic classifier.

The Naive Bayes classifier

- The Naive Bayes classifier is a probabilistic classifier.
- We compute the probability of a document d being in a class c as follows:

$$P(c|d) \propto P(c) \prod_{1 \leq k \leq n_d} P(t_k|c)$$

The Naive Bayes classifier

- The Naive Bayes classifier is a probabilistic classifier.
- We compute the probability of a document d being in a class c as follows:

$$P(c|d) \propto P(c) \prod_{1 \leq k \leq n_d} P(t_k|c)$$

- n_d is the length of the document. (number of tokens)

The Naive Bayes classifier

- The Naive Bayes classifier is a probabilistic classifier.
- We compute the probability of a document d being in a class c as follows:

$$P(c|d) \propto P(c) \prod_{1 \leq k \leq n_d} P(t_k|c)$$

- n_d is the length of the document. (number of tokens)
- $P(t_k|c)$ is the conditional probability of term t_k occurring in a document of class c

The Naive Bayes classifier

- The Naive Bayes classifier is a probabilistic classifier.
- We compute the probability of a document d being in a class c as follows:

$$P(c|d) \propto P(c) \prod_{1 \leq k \leq n_d} P(t_k|c)$$

- n_d is the length of the document. (number of tokens)
- $P(t_k|c)$ is the conditional probability of term t_k occurring in a document of class c
- $P(t_k|c)$ as a measure of how much evidence t_k contributes that c is the correct class.

The Naive Bayes classifier

- The Naive Bayes classifier is a probabilistic classifier.
- We compute the probability of a document d being in a class c as follows:

$$P(c|d) \propto P(c) \prod_{1 \leq k \leq n_d} P(t_k|c)$$

- n_d is the length of the document. (number of tokens)
- $P(t_k|c)$ is the conditional probability of term t_k occurring in a document of class c
- $P(t_k|c)$ as a measure of how much evidence t_k contributes that c is the correct class.
- $P(c)$ is the prior probability of c.

The Naive Bayes classifier

- The Naive Bayes classifier is a probabilistic classifier.
- We compute the probability of a document d being in a class c as follows:

$$P(c|d) \propto P(c) \prod_{1 \leq k \leq n_d} P(t_k|c)$$

- n_d is the length of the document. (number of tokens)
- $P(t_k|c)$ is the conditional probability of term t_k occurring in a document of class c
- $P(t_k|c)$ as a measure of how much evidence t_k contributes that c is the correct class.
- $P(c)$ is the prior probability of c.
- If a document's terms do not provide clear evidence for one class vs. another, we choose the c with highest $P(c)$.

Maximum a posteriori class

- Our goal in Naive Bayes classification is to find the “best” class.

Maximum a posteriori class

- Our goal in Naive Bayes classification is to find the "best" class.
- The best class is the most likely or maximum a posteriori (MAP) class c_{map}:

$$c_{\text{map}} = \arg\max_{c \in \mathbb{C}} \hat{P}(c|d) = \arg\max_{c \in \mathbb{C}} \ \hat{P}(c) \prod_{1 \leq k \leq n_d} \hat{P}(t_k|c)$$

Taking the log

Taking the log

- Multiplying lots of small probabilities can result in floating point underflow.

Taking the log

- Multiplying lots of small probabilities can result in floating point underflow.
- Since $\log(xy) = \log(x) + \log(y)$, we can sum log probabilities instead of multiplying probabilities.

Taking the log

- Multiplying lots of small probabilities can result in floating point underflow.
- Since $\log(xy) = \log(x) + \log(y)$, we can sum log probabilities instead of multiplying probabilities.
- Since log is a monotonic function, the class with the highest score does not change.

Taking the log

- Multiplying lots of small probabilities can result in floating point underflow.
- Since $\log(xy) = \log(x) + \log(y)$, we can sum log probabilities instead of multiplying probabilities.
- Since log is a monotonic function, the class with the highest score does not change.
- So what we usually compute in practice is:

$$c_{\text{map}} = \underset{c \in \mathbb{C}}{\arg\max} \, [\log \hat{P}(c) + \sum_{1 \leq k \leq n_d} \log \hat{P}(t_k|c)]$$

Naive Bayes classifier

Naive Bayes classifier

- Classification rule:

$$c_{\text{map}} = \underset{c \in \mathbb{C}}{\arg\max} \, [\log \hat{P}(c) + \sum_{1 \leq k \leq n_d} \log \hat{P}(t_k|c)]$$

Naive Bayes classifier

- Classification rule:

$$c_{\mathrm{map}} = \underset{c \in \mathbb{C}}{\arg\max} \, [\log \hat{P}(c) + \sum_{1 \leq k \leq n_d} \log \hat{P}(t_k|c)]$$

- Simple interpretation:

Naive Bayes classifier

- Classification rule:

$$c_{\text{map}} = \arg\max_{c \in \mathbb{C}} \, [\log \hat{P}(c) + \sum_{1 \leq k \leq n_d} \log \hat{P}(t_k|c)]$$

- Simple interpretation:
  - Each conditional parameter $\log \hat{P}(t_k|c)$ is a weight that indicates how good an indicator t_k is for c.

Naive Bayes classifier

- Classification rule:

$$c_{\text{map}} = \arg\max_{c \in \mathbb{C}} \, [\log \hat{P}(c) + \sum_{1 \leq k \leq n_d} \log \hat{P}(t_k|c)]$$

- Simple interpretation:
  - Each conditional parameter $\log \hat{P}(t_k|c)$ is a weight that indicates how good an indicator t_k is for c.
  - The prior $\log \hat{P}(c)$ is a weight that indicates the relative frequency of c.

- Classification rule:

$$c_{\text{map}} = \arg\max_{c \in \mathbb{C}} \, [\log \hat{P}(c) + \sum_{1 \leq k \leq n_d} \log \hat{P}(t_k|c)]$$

- Simple interpretation:
  - Each conditional parameter $\log \hat{P}(t_k|c)$ is a weight that indicates how good an indicator t_k is for c.
  - The prior $\log \hat{P}(c)$ is a weight that indicates the relative frequency of c.
  - The sum of log prior and term weights is then a measure of how much evidence there is for the document being in the class.

Naive Bayes classifier

- Classification rule:

$$c_{\text{map}} = \arg\max_{c \in \mathbb{C}} \,[\log \hat{P}(c) + \sum_{1 \leq k \leq n_d} \log \hat{P}(t_k|c)]$$

- Simple interpretation:
  - Each conditional parameter $\log \hat{P}(t_k|c)$ is a weight that indicates how good an indicator t_k is for c.
  - The prior $\log \hat{P}(c)$ is a weight that indicates the relative frequency of c.
  - The sum of log prior and term weights is then a measure of how much evidence there is for the document being in the class.
  - We select the class with the most evidence.

Parameter estimation take 1: Maximum likelihood

Parameter estimation take 1: Maximum likelihood

- Estimate parameters $\hat{P}(c)$ and $\hat{P}(t_k|c)$ from train data: How?

Parameter estimation take 1: Maximum likelihood

- Estimate parameters $\hat{P}(c)$ and $\hat{P}(t_k|c)$ from train data: How?
- Prior:

$$\hat{P}(c) = \frac{N_c}{N}$$

Parameter estimation take 1: Maximum likelihood

- Estimate parameters $\hat{P}(c)$ and $\hat{P}(t_k|c)$ from train data: How?
- Prior:

$$\hat{P}(c) = \frac{N_c}{N}$$

- N_c: number of docs in class c; N: total number of docs

Parameter estimation take 1: Maximum likelihood

- Estimate parameters $\hat{P}(c)$ and $\hat{P}(t_k|c)$ from train data: How?
- Prior:

$$\hat{P}(c) = \frac{N_c}{N}$$

- N_c: number of docs in class c; N: total number of docs
- Conditional probabilities:

$$\hat{P}(t|c) = \frac{T_{ct}}{\sum_{t' \in V} T_{ct'}}$$

Parameter estimation take 1: Maximum likelihood

- Estimate parameters $\hat{P}(c)$ and $\hat{P}(t_k|c)$ from train data: How?
- Prior:

$$\hat{P}(c) = \frac{N_c}{N}$$

- N_c: number of docs in class c; N: total number of docs
- Conditional probabilities:

$$\hat{P}(t|c) = \frac{T_{ct}}{\sum_{t' \in V} T_{ct'}}$$

- T_{ct} is the number of tokens of t in training documents from class c (includes multiple occurrences)

Parameter estimation take 1: Maximum likelihood

- Estimate parameters $\hat{P}(c)$ and $\hat{P}(t_k|c)$ from train data: How?
- Prior:

$$\hat{P}(c) = \frac{N_c}{N}$$

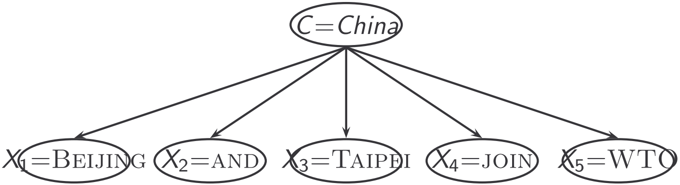
- N_c: number of docs in class c; N: total number of docs
- Conditional probabilities:

$$\hat{P}(t|c) = \frac{T_{ct}}{\sum_{t' \in V} T_{ct'}}$$

- T_{ct} is the number of tokens of t in training documents from class c (includes multiple occurrences)
- We've made a Naive Bayes independence assumption here: $\hat{P}(t_{k_1}|c) = \hat{P}(t_{k_2}|c)$, independent of positions k_1, k_2

The problem with maximum likelihood estimates: Zeros

The problem with maximum likelihood estimates: Zeros

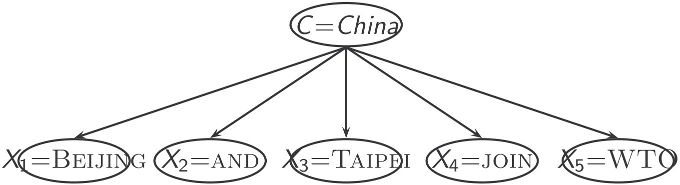


$$\begin{aligned} P(China|d) \quad \propto \quad & P(China) \cdot P(\text{BEIJING}|China) \cdot P(\text{AND}|China) \\ & \cdot P(\text{TAIPEI}|China) \cdot P(\text{JOIN}|China) \cdot P(\text{WTO}|China) \end{aligned}$$

- If WTO never occurs in class China in the train set:

$$\hat{P}(\text{WTO}|China) = \frac{T_{China,\text{WTO}}}{\sum_{t' \in V} T_{China,t'}} = \frac{0}{\sum_{t' \in V} T_{China,t'}} = 0$$

The problem with maximum likelihood estimates: Zeros



$$
\begin{aligned}
P(\mathit{China}|d) \quad \propto \quad & P(\mathit{China}) \cdot P(\textrm{Beijing}|\mathit{China}) \cdot P(\textrm{and}|\mathit{China}) \\
& \cdot P(\textrm{Taipei}|\mathit{China}) \cdot P(\textrm{join}|\mathit{China}) \cdot P(\textrm{WTO}|\mathit{China})
\end{aligned}
$$

- If WTO never occurs in class China in the train set:

$$
\hat{P}(\textrm{WTO}|\mathit{China}) = \frac{T_{\mathit{China},\textrm{WTO}}}{\sum_{t' \in V} T_{\mathit{China},t'}} = \frac{0}{\sum_{t' \in V} T_{\mathit{China},t'}} = 0
$$

The problem with maximum likelihood estimates: Zeros

- If there are no occurrences of WTO in documents in class China, we get a zero estimate:

$$\hat{P}(\mathrm{WTO}|\mathit{China}) = \frac{T_{\mathit{China},\mathrm{WTO}}}{\sum_{t' \in V} T_{\mathit{China},t'}} = 0$$

The problem with maximum likelihood estimates: Zeros

- If there are no occurrences of WTO in documents in class China, we get a zero estimate:

$$\hat{P}(\mathrm{WTO}|China) = \frac{T_{China,\mathrm{WTO}}}{\sum_{t' \in V} T_{China,t'}} = 0$$

- $\rightarrow$ We will get $P(China|d) = 0$ for any document that contains WTO!

To avoid zeros: Add-one smoothing

To avoid zeros: Add-one smoothing

- Before:

$$\hat{P}(t|c) = \frac{T_{ct}}{\sum_{t' \in V} T_{ct'}}$$

To avoid zeros: Add-one smoothing

- Before:

$$\hat{P}(t|c) = \frac{T_{ct}}{\sum_{t' \in V} T_{ct'}}$$

- Now: Add one to each count to avoid zeros:

$$\hat{P}(t|c) = \frac{T_{ct} + 1}{\sum_{t' \in V}(T_{ct'} + 1)} = \frac{T_{ct} + 1}{(\sum_{t' \in V} T_{ct'}) + B}$$

To avoid zeros: Add-one smoothing

- Before:

$$\hat{P}(t|c) = \frac{T_{ct}}{\sum_{t' \in V} T_{ct'}}$$

- Now: Add one to each count to avoid zeros:

$$\hat{P}(t|c) = \frac{T_{ct} + 1}{\sum_{t' \in V} (T_{ct'} + 1)} = \frac{T_{ct} + 1}{(\sum_{t' \in V} T_{ct'}) + B}$$

- B is the number of bins – in this case the number of different words or the size of the vocabulary $|V| = M$

Example

	docID	words in document	in $c = China$?
training set	1	Chinese Beijing Chinese	yes
	2	Chinese Chinese Shanghai	yes
	3	Chinese Macao	yes
	4	Tokyo Japan Chinese	no
test set	5	Chinese Chinese Chinese Tokyo Japan	?

- Estimate parameters of Naive Bayes classifier
- Classify test document

$|text_c| = 8$
$|text_{\overline{c}}| = 3$
B=6 (vocabulary)

Example: Parameter estimates

Example: Parameter estimates

Priors: $\hat{P}(c) = 3/4$ and $\hat{P}(\overline{c}) = 1/4$

Conditional probabilities:

$$
\begin{aligned}
\hat{P}(\text{CHINESE}|c) &= (5+1)/(8+6) = 6/14 = 3/7 \\
\hat{P}(\text{TOKYO}|c) = \hat{P}(\text{JAPAN}|c) &= (0+1)/(8+6) = 1/14 \\
\hat{P}(\text{CHINESE}|\overline{c}) &= (1+1)/(3+6) = 2/9 \\
\hat{P}(\text{TOKYO}|\overline{c}) = \hat{P}(\text{JAPAN}|\overline{c}) &= (1+1)/(3+6) = 2/9
\end{aligned}
$$

The denominators are $(8+6)$ and $(3+6)$ because the lengths of $text_c$ and $text_{\overline{c}}$ are 8 and 3, respectively, and because the constant B is 6 as the vocabulary consists of six terms.

Example: Classification

Example: Classification

$$\begin{aligned}\hat{P}(c|d_5) &\propto 3/4 \cdot (3/7)^3 \cdot 1/14 \cdot 1/14 \approx 0.0003 \\ \hat{P}(\overline{c}|d_5) &\propto 1/4 \cdot (2/9)^3 \cdot 2/9 \cdot 2/9 \approx 0.0001\end{aligned}$$

Thus, the classifier assigns the test document to $c = China$. The reason for this classification decision is that the three occurrences of the positive indicator CHINESE in d_5 outweigh the occurrences of the two negative indicators JAPAN and TOKYO.

Time complexity of Naive Bayes

Time complexity of Naive Bayes

mode	time complexity
training	$\Theta(\lvert\mathbb{D}\rvert L_{\mathrm{ave}} + \lvert\mathbb{C}\rvert\lvert V\rvert)$
testing	$\Theta(L_{\mathrm{a}} + \lvert\mathbb{C}\rvert M_{\mathrm{a}}) = \Theta(\lvert\mathbb{C}\rvert M_{\mathrm{a}})$

- L_{ave}: average length of a training doc, L_{a}: length of the test doc, M_{a}: number of distinct terms in the test doc, $\mathbb{D}$: training set, V: vocabulary, $\mathbb{C}$: set of classes

Time complexity of Naive Bayes

mode	time complexity
training	$\Theta(\lvert\mathbb{D}\rvert L_{\mathrm{ave}} + \lvert\mathbb{C}\rvert\lvert V\rvert)$
testing	$\Theta(L_{\mathrm{a}} + \lvert\mathbb{C}\rvert M_{\mathrm{a}}) = \Theta(\lvert\mathbb{C}\rvert M_{\mathrm{a}})$

- L_{ave}: average length of a training doc, L_{a}: length of the test doc, M_{a}: number of distinct terms in the test doc, $\mathbb{D}$: training set, V: vocabulary, $\mathbb{C}$: set of classes
- $\Theta(\lvert\mathbb{D}\rvert L_{\mathrm{ave}})$ is the time it takes to compute all counts. Note that $\lvert\mathbb{D}\rvert L_{\mathrm{ave}}$ is T, the size of our collection.

Time complexity of Naive Bayes

mode	time complexity
training	$\Theta(\lvert\mathbb{D}\rvert L_{\text{ave}} + \lvert\mathbb{C}\rvert\lvert V\rvert)$
testing	$\Theta(L_{\text{a}} + \lvert\mathbb{C}\rvert M_{\text{a}}) = \Theta(\lvert\mathbb{C}\rvert M_{\text{a}})$

- L_{ave}: average length of a training doc, L_{a}: length of the test doc, M_{a}: number of distinct terms in the test doc, $\mathbb{D}$: training set, V: vocabulary, $\mathbb{C}$: set of classes
- $\Theta(\lvert\mathbb{D}\rvert L_{\text{ave}})$ is the time it takes to compute all counts. Note that $\lvert\mathbb{D}\rvert L_{\text{ave}}$ is T, the size of our collection.
- $\Theta(\lvert\mathbb{C}\rvert\lvert V\rvert)$ is the time it takes to compute the conditional probabilities from the counts.

Time complexity of Naive Bayes

mode	time complexity
training	$\Theta(\lvert\mathbb{D}\rvert L_{\mathrm{ave}} + \lvert\mathbb{C}\rvert\lvert V\rvert)$
testing	$\Theta(L_{\mathrm{a}} + \lvert\mathbb{C}\rvert M_{\mathrm{a}}) = \Theta(\lvert\mathbb{C}\rvert M_{\mathrm{a}})$

- L_{ave}: average length of a training doc, L_{a}: length of the test doc, M_{a}: number of distinct terms in the test doc, $\mathbb{D}$: training set, V: vocabulary, $\mathbb{C}$: set of classes
- $\Theta(\lvert\mathbb{D}\rvert L_{\mathrm{ave}})$ is the time it takes to compute all counts. Note that $\lvert\mathbb{D}\rvert L_{\mathrm{ave}}$ is T, the size of our collection.
- $\Theta(\lvert\mathbb{C}\rvert\lvert V\rvert)$ is the time it takes to compute the conditional probabilities from the counts.
- Generally: $\lvert\mathbb{C}\rvert\lvert V\rvert < \lvert\mathbb{D}\rvert L_{\mathrm{ave}}$

Time complexity of Naive Bayes

mode	time complexity
training	$\Theta(\lvert\mathbb{D}\rvert L_{\mathrm{ave}} + \lvert\mathbb{C}\rvert\lvert V\rvert)$
testing	$\Theta(L_{\mathrm{a}} + \lvert\mathbb{C}\rvert M_{\mathrm{a}}) = \Theta(\lvert\mathbb{C}\rvert M_{\mathrm{a}})$

- L_{ave}: average length of a training doc, L_{a}: length of the test doc, M_{a}: number of distinct terms in the test doc, $\mathbb{D}$: training set, V: vocabulary, $\mathbb{C}$: set of classes
- $\Theta(\lvert\mathbb{D}\rvert L_{\mathrm{ave}})$ is the time it takes to compute all counts. Note that $\lvert\mathbb{D}\rvert L_{\mathrm{ave}}$ is T, the size of our collection.
- $\Theta(\lvert\mathbb{C}\rvert\lvert V\rvert)$ is the time it takes to compute the conditional probabilities from the counts.
- Generally: $\lvert\mathbb{C}\rvert\lvert V\rvert < \lvert\mathbb{D}\rvert L_{\mathrm{ave}}$
- Test time is also linear (in the length of the test document).

Time complexity of Naive Bayes

mode	time complexity
training	$\Theta(\lvert\mathbb{D}\rvert L_{\text{ave}} + \lvert\mathbb{C}\rvert\lvert V\rvert)$
testing	$\Theta(L_{\text{a}} + \lvert\mathbb{C}\rvert M_{\text{a}}) = \Theta(\lvert\mathbb{C}\rvert M_{\text{a}})$

- L_{ave}: average length of a training doc, L_{a}: length of the test doc, M_{a}: number of distinct terms in the test doc, $\mathbb{D}$: training set, V: vocabulary, $\mathbb{C}$: set of classes
- $\Theta(\lvert\mathbb{D}\rvert L_{\text{ave}})$ is the time it takes to compute all counts. Note that $\lvert\mathbb{D}\rvert L_{\text{ave}}$ is T, the size of our collection.
- $\Theta(\lvert\mathbb{C}\rvert\lvert V\rvert)$ is the time it takes to compute the conditional probabilities from the counts.
- Generally: $\lvert\mathbb{C}\rvert\lvert V\rvert < \lvert\mathbb{D}\rvert L_{\text{ave}}$
- Test time is also linear (in the length of the test document).
- Thus: Naive Bayes is linear in the size of the training set (training) and the test document (testing). This is optimal.

Naive Bayes is not so naive

- Multinomial model violates two independence assumptions and yet...

Naive Bayes is not so naive

- Multinomial model violates two independence assumptions and yet...
- Naive Bayes has won some competitions (e.g., KDD-CUP 97; prediction of most likely donors for a charity)

Naive Bayes is not so naive

- Multinomial model violates two independence assumptions and yet...
- Naive Bayes has won some competitions (e.g., KDD-CUP 97; prediction of most likely donors for a charity)
- More robust to nonrelevant features than some more complex learning methods

Naive Bayes is not so naive

- Multinomial model violates two independence assumptions and yet...
- Naive Bayes has won some competitions (e.g., KDD-CUP 97; prediction of most likely donors for a charity)
- More robust to nonrelevant features than some more complex learning methods
- More robust to concept drift (changing of definition of class over time) than some more complex learning methods

Naive Bayes is not so naive

- Multinomial model violates two independence assumptions and yet...
- Naive Bayes has won some competitions (e.g., KDD-CUP 97; prediction of most likely donors for a charity)
- More robust to nonrelevant features than some more complex learning methods
- More robust to concept drift (changing of definition of class over time) than some more complex learning methods
- Better than methods like decision trees when we have many equally important features

Naive Bayes is not so naive

- Multinomial model violates two independence assumptions and yet...
- Naive Bayes has won some competitions (e.g., KDD-CUP 97; prediction of most likely donors for a charity)
- More robust to nonrelevant features than some more complex learning methods
- More robust to concept drift (changing of definition of class over time) than some more complex learning methods
- Better than methods like decision trees when we have many equally important features
- A good dependable baseline for text classification (but not the best)

Naive Bayes is not so naive

- Multinomial model violates two independence assumptions and yet...
- Naive Bayes has won some competitions (e.g., KDD-CUP 97; prediction of most likely donors for a charity)
- More robust to nonrelevant features than some more complex learning methods
- More robust to concept drift (changing of definition of class over time) than some more complex learning methods
- Better than methods like decision trees when we have many equally important features
- A good dependable baseline for text classification (but not the best)
- Optimal if independence assumptions hold (never true for text, but true for some domains)

Naive Bayes is not so naive

- Multinomial model violates two independence assumptions and yet...
- Naive Bayes has won some competitions (e.g., KDD-CUP 97; prediction of most likely donors for a charity)
- More robust to nonrelevant features than some more complex learning methods
- More robust to concept drift (changing of definition of class over time) than some more complex learning methods
- Better than methods like decision trees when we have many equally important features
- A good dependable baseline for text classification (but not the best)
- Optimal if independence assumptions hold (never true for text, but true for some domains)
- Very fast; low storage requirements

Not covered

- Derivation of NB formula
- Evaluation of text classification

Summary

- Query-likelihood as a general principle for ranking documents in an unsupervised manner
  - Treat queries as strings
  - Rank documents according to their models
- Document language models
  - Know the difference between the document and the document model
  - Multinomial distribution is simple but effective
- Smoothing
  - Reasons for, and importance of, smoothing
  - Dirichlet (Bayesian) smoothing is very effective
- Classification
  - Text classification is supervised learning
  - Naive Bayes: simple baseline text classifier

Reading

- Manning, Raghavan, Schütze: Introduction to Information Retrieval (MRS), chapter 12: Language models for information retrieval
- MRS chapters 13.1-13.4 for text classification