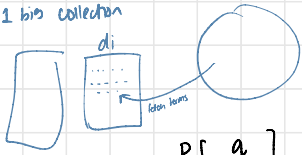
Language Model

L generative models → generates data

assumes there is 1 big collection



$$P\left[\frac{q}{MD_i}\right]$$

Assumptions

1. d and q are of same type
2. d are generated by a random experiment
3. q generation process which is similar to d generation
4. terms are conditionally independent

d we have → features generated from some random Problem

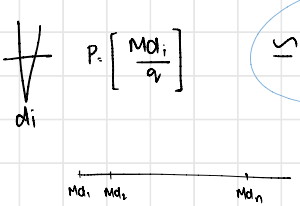
Steps

1. Model d → Md_i (each d is treated as a LM)
2. Query Generation Process
   - Swim MLK
3. Rank on their Probability

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

$D = \{Md_1, Md_2 \ldots\ldots Md_n\}$

$\theta = \{q\}$



$P(d)$ = Prior P of hypothesis A — PRIOR

$P(q)$: Prior P of query q — EVIDENCE

$P(q|d)$: P of q given d — LIKELIHOOD

$P(d|q)$: P of d given q — POSTERIOR

A1: $P(q)$ is same for all d → ignore

A2: $P(d)$ is uniform

So d can be ranked according to relevance to q

ranking according to $P(d|q)$ and $P(q|d)$ is equivalent

$\{H_1, H_2, H_3 \ldots H_n\}$

$\{T_3 \_\_\_\_\_\}$

$\_\_\_\_\_$

$U1 = \{H_1 \_\_\_\_\_\}$

$P(U1)_{ML}$

Assumptions of LM

- Simplifying assumption: Queries and documents are objects of same type. Not true!
  - There are other LMs for IR that do not make this assumption.
  - The vector space model makes the same assumption.
- Simplifying assumption: Terms are conditionally independent.
  - Again, vector space model (and Naive Bayes) makes the same assumption.
- Cleaner statement of assumptions than vector space
- Thus, better theoretical foundation than vector space
  - ... but "pure" LMs perform much worse than "tuned" LMs.

Query Likelihood Model

↳ basic method for LM

↳ construct M_d (LM) from each document d

↳ goal → rank d by $P(q|d)$

where $P(d)$ is interpreted as likelihood

Bayesian Rule

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

A1: $P(q)$ is same for all d → ignored

A2: $P(d)$ is uniform → also ignored

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

↳ but we can implement genuine prior

which could include criteria like

- ↳ authority
- ↳ length
- ↳ genre
- ↳ newness

but given these simplifications, we return results ranked by simply

$P(q|d)$

MLE

MAP → you have prior info

Genuine Prior

attempts to model a generation process

Language model approach:

↳ d are ranked by the P that a q would be observed as a random sample from the respective d model

Query Likelihood Model

The intuition of the basic model is that the user has a prototype document in mind and generates a query based on words that appear in this document. Often, users have a reasonable idea of terms that are likely to occur in documents of interest and they will choose query terms that distinguish these documents from others in the collection.[3] Collection statistics are an integral part of the language model, rather than being used heuristically as in many other approaches.

Multinomial Unigram language model

↳ most common way for LM approach

↳ equivalent to multinomial naive Bayes model → where d are classes, each treated in the estimation as a separate language

$$P(q|M_d) = K_q \prod_{t \in V} P(t|M_d)^{tf_{t,q}}$$

K_q: multinomial coefficient for query q → ignore as constant for a particular q

↳ In LM we treat generation of queries as a random process

STEPS

1. infer LM for each d
2. Estimate $P(q|M_{d1})$ → P of generating the query according to each of these d models
3. Rank d according to these probabilities

HOW to estimate $P(q|M_d)$ using MLE

→ maximum likelihood estimation

$$\hat{P}(q|M_d) = \prod_{t \in q} \hat{P}_{MLE}(t|M_d) = \prod_{t \in q} \frac{tf_{t,d}}{L_d}$$

$\hat{P}$ → estimation symbol

$\hat{P}_{MLE}(t|M_d)$: language model of d

$tf_{t,d}$ → raw term frequency of term t in doc d

L_d → no of tokens in d

→ count how often each word occured / total no of words in d

CON

↳ if 0 comes everything becomes 0

so we do soothing

Query Likelihood Model

The classic problem with using language models is one of estimation (the ^ symbol on the P's is used above to stress that the model is estimated): terms appear very sparsely in documents. In particular, some words will not have appeared in the document at all, but are possible words for the information need, which the user may have used in the query. If we estimate $\hat{P}(t|M_d) = 0$ for a term missing from a document d, then we get a strict conjunctive semantics: documents will only give a query non-zero probability if all of the query terms appear in the document. Zero probabilities are clearly a problem in other uses of language models, such as when predicting the next word in a speech recognition application, because many words will be sparsely represented in the training data. It may seem rather less clear whether this is problematic in an IR application. This could be thought of as a human-computer interface issue: vector space systems have generally preferred more lenient matching, though recent web search developments have tended more in the direction of doing searches with such conjunctive semantics. Regardless of the approach here, there is a more general problem of estimation: occurring words are also badly estimated; in particular,

Query Likelihood Model

counts and renormalizing to give a probability distribution.[4] In this section we will mention a couple of other smoothing methods, which involve combining observed counts with a more general reference probability distribution. The general approach is that a non-occurring term should be possible in a query, but its probability should be somewhat close to but no more likely than would be expected by chance from the whole collection. That is, if $tf_{t,d} = 0$ then

$$\hat{P}(t|M_d) \leq cf_t/T$$

where cf_t is the raw count of the term in the collection, and T is the raw size (number of tokens) of the entire collection. A simple idea that works well in practice is to use a mixture between a document-specific multinomial distribution and a multinomial distribution estimated from the entire collection:

$$\hat{P}(t|d) = \lambda \hat{P}_{mle}(t|M_d) + (1 - \lambda)\hat{P}_{mle}(t|M_c)$$

Query Likelihood Model

where $0 < \lambda < 1$ and M_c is a language model built from the entire document collection. This mixes the probability from the document with the general collection frequency of the word. Such a model is referred to as a *linear interpolation* language model.[5] Correctly setting λ is important to the good performance of this model.

An alternative is to use a language model built from the whole collection as a prior distribution in a *Bayesian updating process* (rather than a uniform distribution, as we saw in Section 11.3.2). We then get the following equation:

$$\hat{P}(t|d) = \frac{\mathrm{tf}_{t,d} + \alpha \hat{P}(t|M_c)}{L_d + \alpha}$$

Both of these smoothing methods have been shown to perform well in IR experiments; we will stick with the linear interpolation smoothing method for the rest of this section. While different in detail, they are both conceptually similar: in both cases the probability estimate for a word present in the document combines a discounted MLE and a fraction of the estimate of its prevalence in the whole collection, while for words not present in a document, the estimate is just a fraction of the estimate of the prevalence of the word in the whole collection.

Query Likelihood Model

of the models. The reason for this is explored in Exercise 12.8. The extent of smoothing in these two models is controlled by the λ and α parameters: a small value of λ or a large value of α means more smoothing. This parameter can be tuned to optimize performance using a line search (or, for the linear interpolation model, by other methods, such as the expectation maximimization algorithm; see Section 16.5, page 368). The value need not be a constant. One approach is to make the value a function of the query size. This is useful because a small amount of smoothing (a "conjunctive-like" search) is more suitable for short queries, while a lot of smoothing is more suitable for long queries.

To summarize, the retrieval ranking for a query q under the basic LM for IR we have been considering is given by:

$$P(d|q) \propto P(d) \prod_{t \in q} ((1-\lambda)P(t|M_c) + \lambda P(t|M_d))$$

This equation captures the probability that the document that the user had in mind was in fact d.

Example

Example 12.3: Suppose the document collection contains two documents:

- d_1: Xyzzy reports a profit but revenue is down
- d_2: Quorus narrows quarter loss but revenue decreases further

The model will be MLE unigram models from the documents and collection, mixed with $\lambda = 1/2$.

Suppose the query is *revenue down*. Then:

$$\begin{aligned} P(q|d_1) &= [(1/8 + 2/16)/2] \times [(1/8 + 1/16)/2] \\ &= 1/8 \times 3/32 = 3/256 \\ P(q|d_2) &= [(1/8 + 2/16)/2] \times [(0/8 + 1/16)/2] \\ &= 1/8 \times 1/32 = 1/256 \end{aligned}$$

So, the ranking is $d_1 > d_2$.

revenue in collection = 2

down in collection = 1

total collection = 16

total words d_1 = 8

total words d_2 = 8

	tf revenue	tf down
d_1	1	1
d_2	1	0

$$P(q|d_1) = \left[\frac{\frac{1}{8} + \frac{2}{16}}{2}\right] \times \left[\frac{\frac{1}{8} + \frac{1}{16}}{2}\right]$$

revenue: $\frac{2}{16}$ — in document; down: $\frac{1}{16}$ — in collection

Smoothing Language Model

1. Jelinek-Mercer Smoothing
2. Dirichlet Smoothing
3. Good Turing Smoothing

b. Suppose we have a collection that consists of the 4 documents given below:
D1: click go the shears boys click click click
D2: click click
D3: metal here
D4: metal shears click here

Build a query likelihood language model for this document collection. Assume a mixture model between the documents and the collection, with both weighted at 0.5. Maximum likelihood estimation (mle) is used to estimate both as unigram models. Calculate the model probabilities of the queries click, shears, and hence click shears for each document, and use those probabilities to rank the documents returned by each query. [10]

	click	go	the	shears	boys	metal	here
Mc	7/16	1/16	1/16	2/16	1/16	2/16	2/16
P(q/D1)	4/8	1/8	1/8	1/8	1/8	0	0
P(q/D2)	2/2	0	0	1	0	0	0
P(q/D3)	0	0	0	1	0	1/2	1/2
P(q/D4)	1/4	0	0	1/4	0	1/4	1/4

$P(q|d_1)$

$\lambda = 0.5 = \frac{1}{2}$

	D1	D2	D3	D4
click	1/2*(4/8+7/16)	1/2*(2/2+7/16)	1/2*(0+7/16)	1/2*(1/4+7/16)
shears	1/2*(1/8+2/16)	1/2*(0+2/16)	1/2*(0+2/16)	1/2*(1/8+2/16)
click shears	0.059	0.045	0.014	0.043

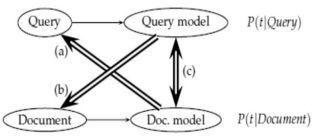
Click → D2, D1, D4, D3
Shears → D4, D1 D2 D3
Click Shear → D4, D1, D2, D3

Extended Language model

Is Probability of M_d (query LM) generating document

- Rather than looking at the probability of a document language model M_d generating the query, you can look at the probability of a query language model M_q generating the document.
- The main reason that doing things in this direction and creating a document likelihood model is less appealing is that there is much less text available to estimate a language model based on the query text, and so the model will be worse estimated. (Smoothing factors)

Extended Language Model



► **Figure 12.5** Three ways of developing the language modeling approach: (a) query likelihood, (b) document likelihood, and (c) model comparison.

Kullback Leibler Divergence Model

KL divergence is an asymmetric divergence measure originating in information theory, which measures how bad the probability distribution M_q is at modeling M_d (Cover and Thomas 1991, Manning and Schütze 1999). Lafferty and Zhai (2001) present results suggesting that a model comparison approach outperforms both query-likelihood and document-likelihood approaches. One disadvantage of using KL divergence as a ranking function is that scores are not comparable across queries. This does not matter for ad hoc retrieval, but is important in other applications such as topic tracking. Kraaij and Spitters (2003) suggest an alternative proposal which models similarity as a normalized log-likelihood ratio (or, equivalently, as a difference between cross-entropies).

KL Divergence Model

$$R(d;q) = KL(M_d||M_q) = \sum_{t \in V} P(t|M_q) \log \frac{P(t|M_q)}{P(t|M_d)}$$

TEXT CLASSIFICATION

CHP 13

Ad hoc Query

↳ Users have transient information needs that they try to address by posting 1 or more queries to search engine

Standing Query

↳ It is periodically executed on a collection to which new docs are incrementally added overtime

Text classification

↳ process of assigning a predefined set of classes to new instance of text

1. Manual
2. Supervised
3. Semi Supervised
4. Un-supervised

- *Input*:
  - a text piece/ a document d
  - a fixed set of classes $C = \{c_1, c_2, \ldots, c_j\}$
- *Output*: a predicted class $c_i \in C$

1. Manual Hand coded rules

↳ rules based on combinations of words/other features

e.g. spam. black-list. address OR ("dollars" AND "have been selected")

↳ accuracy can be high

↳ if rules are carefully defined by expert

CON

↳ expensive to build and maintain rules

2. Supervised

↳ input

↳ doc d

↳ fixed set of classes $C = \{c_1, c_2, \ldots c_n\}$

↳ training set of m hand labeled docs $\{d_1, c_1\} \ldots \{d_m, c_m\}$

↳ output

↳ a learned classifier c

3. Semi Supervised

↳ started as supervised

↳ tune in such a way that it continues to learn from the functionality of it works

<classification>

4. Unsupervised

↳ It learns on its own how to classify the doc by implicitly learned features

Different Type of Text Classifications

- Predictive
- Usually supervised
- Inputs (independent variable) vs predictors (dependent or response variable)
- Several configurations

Number of outputs	Output type	Classification kind
1 per instance	Binary	Binary
1 per instance	Multivalued	Multiclass
n per instance	Binary	Multilabel
n per instance	Multivalued	Multidimensional
1 per n instances	Binary/Multivalued	Multiinstance

Naïve Bayes Classification

aka Multinomial Naïve Bayes / Multinomial NB model

↳ Supervised learning method

↳ a probabilistic learning method

- The probability of a document d being in class c is computed as

$$\hat{y} = \arg\max_{y \in Y} p(y \mid x_1, \ldots, x_n)$$
$$= \arg\max_{y \in Y} \frac{p(y)p(x_1, \ldots, x_n \mid y)}{p(x_1, \ldots, x_n)}$$
$$= \arg\max_{y \in Y} p(y)p(x_1, \ldots, x_n \mid y)$$
$$= \arg\max_{y \in Y} p(y) \prod_{i=1}^{n} p(x_i \mid y)$$

Naïve Bayes Classification

maximum likelihood estimates

- simply use the frequencies in the data

$$\hat{P}(c_j) = \frac{doccount(C = c_j)}{N_{doc}}$$

$$\hat{P}(w_i \mid c_j) = \frac{count(w_i, c_j)}{\sum_{w \in V} count(w, c_j)}$$

$$\hat{P}(w_i \mid c) = \frac{count(w_i, c) + 1}{\sum_{w \in V} (count(w, c) + 1)}$$
$$= \frac{count(w_i, c) + 1}{\left(\sum_{w \in V} count(w, c)\right) + |V|}$$

Naïve Bayes

- Very Fast, low storage requirements
- Robust to Irrelevant Features
  - Irrelevant Features cancel each other without affecting results
- Very good in domains with many equally important features
  - Decision Trees suffer from *fragmentation* in such cases – especially if little data
- Optimal if the independence assumptions hold: If assumed independence is correct, then it is the Bayes Optimal Classifier for problem
- A good dependable baseline for text classification

Variations of Naive Bayesian

↳ three NB classifiers

1. Multinomial NB
2. Multinomial Bernouli model

1. Mutinomial NB

↳ Priors

$$\text{Prior}(c) = \frac{\text{no of docs in } c}{\text{total no of docs}}$$

↳ Conditional Probability

$$P(\text{chinese}|c) = \frac{\text{TF of chinese in } c + 1}{\text{Total terms of } c + V}$$

(+1 → smoothing; V → no of words in vocabulary)

↳ choosin a class

$$P(c|s) = \text{Prior}(c) \times P(\text{chinese}|c) \times P(\text{Tokyo}|c) \times P(\text{Japan}|c) \times \ldots$$

(chinese → df chinese; Tokyo → df chinese; Japan → df Tokyo; … → terms in ds)

Example

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

$$\hat{P}(w|c) = \frac{count(w,c)+1}{count(c)+|V|}$$

	Doc	Words	Class
Training	1	Chinese Beijing Chinese	c
	2	Chinese Chinese Shanghai	c
	3	Chinese Macao	c
	4	Tokyo Japan Chinese	j
Test	5	Chinese Chinese Chinese Tokyo Japan	?

Priors:

$P(c) = 3/4$

$P(j) = 1/4$

Conditional Probabilities: (smoothing)

P(Chinese|c) = (5+1) / (8+6) = 6/14 = 3/7
P(Tokyo|c) = (0+1) / (8+6) = 1/14
P(Japan|c) = (0+1) / (8+6) = 1/14
P(Chinese|j) = (1+1) / (3+6) = 2/9
P(Tokyo|j) = (1+1) / (3+6) = 2/9
P(Japan|j) = (1+1) / (3+6) = 2/9

Choosing a class: (chinese, tokyo, japan)

P(c|d5) = 3/4 * (3/7)³ * 1/14 * 1/14 ≈ 0.0003

P(j|d5) = 1/4 * (2/9)³ * 2/9 * 2/9 ≈ 0.0001

Multinomial NB

- Training

TrainMultinomialNB(C, D)
1 $V \leftarrow$ ExtractVocabulary(D)
2 $N \leftarrow$ CountDocs(D)
3 for each $c \in C$
4 do $N_c \leftarrow$ CountDocsInClass(D, c)
5 $prior[c] \leftarrow N_c/N$
6 $text_c \leftarrow$ ConcatenateTextOfAllDocsInClass(D, c)
7 for each $t \in V$
8 do $T_{ct} \leftarrow$ CountTokensOfTerm($text_c$, t)
9 for each $t \in V$
10 do $condprob[t][c] \leftarrow \frac{T_{ct}+1}{\sum_{t'}(T_{ct'}+1)}$
11 return V, prior, condprob

Multinomial NB

- Testing

ApplyMultinomialNB(C, V, prior, condprob, d)
1 $W \leftarrow$ ExtractTokensFromDoc(V, d)
2 for each $c \in C$
3 do $score[c] \leftarrow \log prior[c]$
4 for each $t \in W$
5 do $score[c] += \log condprob[t][c]$
6 return $\arg\max_{c \in C} score[c]$

2. Bernouli Model

Priors

Prior (c): $\frac{\text{no of docs in c}}{\text{total no of docs}}$

Conditional Probability

$P(\text{chinese}|c)$: $\frac{\text{df of chinese in c} + 1}{\text{Total docs of c} + 2}$ → smoothing

Choosing class

$P(c|d_5) =$

$= \text{Prior}(c) \times P(\text{chinese}|c) \times P(\text{Tokyo}|c) \times P(\text{Japan}|c) \times$ → terms in d_5

$(1 - P(\text{Beijing}|c)) \cdot (1 - P(\text{Shanghai}|c)) \times (1 - P(\text{Macao}|c))$ → terms not in d_5

Example

	Doc	Words	Class
Training	1	Chinese Beijing Chinese	c
	2	Chinese Chinese Shanghai	c
	3	Chinese Macao	c
	4	Tokyo Japan Chinese	j
Test	5	Chinese Chinese Chinese Tokyo Japan	?

Choosing a class:

$P(c|d_5) \propto P(c) \cdot P(\text{Chinese}|c) \cdot P(\text{Japan}|c) \cdot P(\text{Tokyo}|c) \cdot (1 - P(\text{Beijing}|c)) \cdot (1 - P(\text{Shanghai}|c)) \cdot (1 - P(\text{Macao}|c))$

$= 3/4 \cdot 3/5 \cdot 1/5 \cdot 1/5 \cdot (1-2/5) \cdot (1-2/5) \cdot (1-2/5)$

≈ 0.005

Priors:

$P(c) = 3/4$

$P(\bar{c}) = 1/4$

Conditional Probabilities:

The conditional probabilities are:

$P(\text{Chinese}|c) = (3+1)/(3+2) = 4/5$

$P(\text{Japan}|c) = P(\text{Tokyo}|c) = (0+1)/(3+2) = 1/5$

$P(\text{Beijing}|c) = P(\text{Macao}|c) = P(\text{Shanghai}|c) = (1+1)/(3+2) = 2/5$

$P(\text{Chinese}|\bar{c}) = (1+1)/(1+2) = 2/3$

$P(\text{Japan}|\bar{c}) = P(\text{Tokyo}|\bar{c}) = (1+1)/(1+2) = 2/3$

$P(\text{Beijing}|\bar{c}) = P(\text{Macao}|\bar{c}) = P(\text{Shanghai}|\bar{c}) = (0+1)/(1+2) = 1/3$

$P(\bar{c}|d_5) \propto 1/4 \cdot 2/3 \cdot 2/3 \cdot 2/3 \cdot (1-1/3) \cdot (1-1/3) \cdot (1-1/3)$

≈ 0.022

Bernoulli model

- Training of Model

```
TrainBernoulliNB(C, 𝔻)
1  V ← ExtractVocabulary(𝔻)
2  N ← CountDocs(𝔻)
3  for each c ∈ ℂ
4  do N_c ← CountDocsInClass(𝔻, c)
5     prior[c] ← N_c/N
6     for each t ∈ V
7     do N_ct ← CountDocsInClassContainingTerm(𝔻, c, t)
8        condprob[t][c] ← (N_ct + 1)/(N_c + 2)
9  return V, prior, condprob
```

Bernoulli model

- Testing

```
ApplyBernoulliNB(ℂ, V, prior, condprob, d)
1  V_d ← ExtractTermsFromDoc(V, d)
2  for each c ∈ ℂ
3  do score[c] ← log prior[c]
4     for each t ∈ V
5     do if t ∈ V_d
6           then score[c] += log condprob[t][c]
7           else score[c] += log(1 − condprob[t][c])
8  return arg max_{c∈ℂ} score[c]
```

< 500 is a small doc

Multinomial Vs. Bernoulli

Table 13.3 Multinomial versus Bernoulli model.

	multinomial model	Bernoulli model
event model	generation of token	generation of document
random variable(s)	$X = t$ iff t occurs at given pos	$U_t = 1$ iff t occurs in doc
document representation	$d = \langle t_1, \ldots, t_k, \ldots, t_{n_d} \rangle, t_k \in V$	$d = \langle e_1, \ldots, e_t, \ldots, e_M \rangle, e_t \in \{0, 1\}$
parameter estimation	$\hat{P}(X = t\|c)$	$\hat{P}(U_t = e\|c)$
decision rule: maximize	$\hat{P}(c) \prod_{1 \le k \le n_d} \hat{P}(X = t_k\|c)$	$\hat{P}(c) \prod_{t \in V} \hat{P}(U_t = e_t\|c)$
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 term the	$\hat{P}(X = \text{the}\|c) \approx 0.05$	$\hat{P}(U_{\text{the}} = 1\|c) \approx 1.0$

text, generally improvement can be performed with relaxing the assumptions.

5. How do we treat the words(features) that do not appear in training document for NB Classifiers? [5]

In a Naïve Bayes classifier, the training data never complete and hence there are terms(features/words) that do not appear in training data. The maximum likelihood estimate for such term is zero and hence we cannot be able to the compute probability for a given class. One of the solution to this problem is Laplace smoothing which simply adds one to each count, which is treated as uniform prior.

EVALUATION OF CLASSIFICATION TASK

↳ email classification - simple binary classification
↳ classes: {spam, ham}
↳ training data set = 120 emails
↙ 50 spam ↘ 70 ham

1. Binary classification

	Spam	Ham
Predicted Spam	35	16
Predicted Ham	15	54

$$\text{Accuracy} = \frac{35 + 54}{120}$$
(35: correctly predicted spam; 54: correctly predicted ham; 120: total emails)
Accuracy: fraction of correctly classified items

$$\text{Error} = \frac{15 + 16}{120}$$
(15: incorrect spam prediction; 16: incorrect ham prediction; 120: total emails)
Error: fraction of incorrectly classified items

2. Multiclass classification

- Multiclass Classification

	C1	C2	C3	C4	C5
Pre C1	10	1	0	1	0
Pre C2	2	22	1	1	1
Pre C3	0	2	20	2	2
Pre C4	2	0	12	21	0
Pre C5	0	0	0	0	25

$$\text{Recall} = \frac{c_{ii}}{\sum_j c_{ij}}$$
Recall: fraction of docs in class i classified correctly

$$\text{Precision} = \frac{c_{ii}}{\sum_j c_{ji}}$$
Precision: fraction of docs assigned class i that are actually about class i

$$\text{Accuracy (1 - error)} = \frac{\sum_i c_{ii}}{\sum_j \sum_i c_{ij}}$$
Accuracy: fraction of docs classified correctly

Recall

C1: $\frac{10}{12} = 0.83$ (10 → diagonal; 12 → row sum)
C2: $\frac{22}{27} = 0.81$
C3: $\frac{20}{26} = 0.77$
C4: $\frac{21}{35} = 0.6$
C5: $\frac{25}{25} = 1$

Precision

C1: $\frac{10}{14} = 0.71$ (10 → diagonal; 14 → column sum)
C2: $\frac{22}{25} = 0.88$
C3: $\frac{20}{33} = 0.60$
C4: $\frac{21}{25} = 0.84$
C5: $\frac{25}{28} = 0.89$

Accuracy

$$= \frac{10 + 22 + 20 + 21 + 25}{136} = \frac{98}{136} = 0.72$$
(98: sum of all diagonals; 136: sum of whole table)

3. Macro averaging

↳ when more than 1 class

↳ for combining multiple performance measures into quality

Class 1

	Truth: yes	Truth: no
Classifier: yes	10	10
Classifier: no	10	970

Class 2

	Truth: yes	Truth: no
Classifier: yes	90	10
Classifier: no	10	890

Micro Ave. Table

	Truth: yes	Truth: no
Classifier: yes	100	20
Classifier: no	20	1860

c1 c2: 10 + 10 | 10 + 890

- Microaveraged score is dominated by score on common classes

Precision of class 1: $\frac{10}{10+10} = 0.5$

Precision of class 2: $\frac{90}{90+10}$

Macro averaged precision $= \frac{0.5 + 0.9}{2} = 0.7$ (0.5 = Precision C1, 0.9 = Precision C2, 2 → N = total no of classes)

↳ compute performance for each class then average

Micro averaged precision $= \frac{100}{120} = 0.83$ (100 = correctly classified emails, 120 → total emails)

↳ collect decisions for all classes, compute contingency table, evaluate

Supervised learning datasets

↳ Training / validation / Testing

↳ Model fitting

↳ feature selection

↳ Validation set

↳ Hold out → fixed splits 70/30

↳ cross validation (K-fold) → 10-fold

↳ Variance Vs Bias

↳ error due to Bias

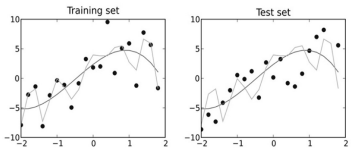
↳ diff b/w expected and actual

↳ error due to variance

↳ variability of model

↳ bulls-eye

Example



A training set (left) and a test set (right) from the same statistical population are shown as blue points. Two predictive models are fit to the training data. Both fitted models are plotted with both the training and test sets. In the training set, the MSE of the fit shown in orange is 4 whereas the MSE for the fit shown in green is 9. In the test set, the MSE for the fit shown in orange is 15 and the MSE for the fit shown in green is 13. The orange curve severely overfits the training data, since its MSE increases by almost a factor of four when comparing the test set to the training set. The green curve overfits the training data much less, as its MSE increases by less than a factor of 2.

Variance vs. Bias

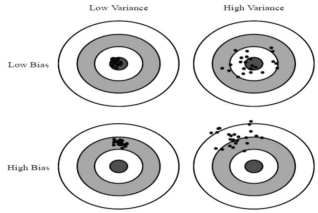


Fig. 1 Graphical illustration of bias and variance.

Textual Features

↳ textual docs are rich in features

↳ words/tokens

↳ phrases/Bi-grams

↳ graphs

↳ NLP based features

Features

↳ an individual measurable property

↳ are usually numeric

↳ but structural features (strings, graphs) are used in syntatic pattern recognition

Feature Selection

↳ process of selecting a sub-set features for model training and using only this subset as feature in text classification

↳ selects only relavent and non redundant features

PROS

↳ reduce the dimensionality (decreases the size of effective vocabulary)

↳ simplify model

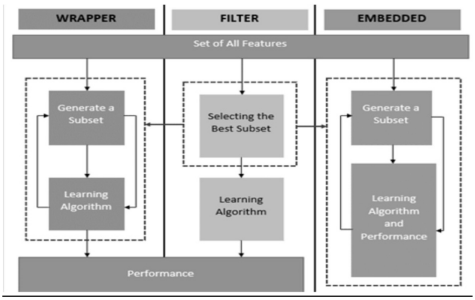
↳ shorter training time

↳ improve model accuracy (noise features are eliminated)

↳ 3 categories of Feature Selection

1. Filter methods
2. Wrapper methods
3. Embedded methods

Feature Selection



Feature Selection

	Filter method	Wrapper method	Embedded method
What is it?	Uses proxy measure	Uses predictive model	Feature selection is embedded in the model building phase
Speed	Computationally faster	Slower	Medium
Overfitting	Avoids overfitting	Prone to overfitting	Less prone to overfitting
Performance	Sometimes may fail to select best features	Better performance	Good performance

↳ 2 types of feature selection in machine learning

1. Wrappers

↳ use classification accuracy of some learning algo as their evaluation function

CONS

↳ more time consuming — as they have to train a classifier for each feature subset to be evaluated

↳ essply when no of features is hiah

↳ hence not suitable for text classification

2. Filters

↳ perform feature selection independently of the learning algo that will use the selected features

↳ they use an evaluation metric that measures the ability of the feature to diff each class

PROS

↳ much less time consuming

↳ widely used in text classification

How Feature Selection Works

- How it Works
  - We can view feature selection as a method for replacing a complex classifier (using all features) with a simpler one (using a subset of the features).
  - The purpose of a feature selection algorithm is to select only those features that For a given class c, we compute a utility measure A(t, c) for each term of the vocabulary and select the k terms that have the highest values of A(t, c).

Features Selection

```
SELECTFEATURES(D, c, k)
1  V ← EXTRACTVOCABULARY(D)
2  L ← []
3  for each t ∈ V
4  do A(t, c) ← COMPUTEFEATUREUTILITY(D, t, c)
5     APPEND(L, ⟨A(t, c), t⟩)
6  return FEATURESWITHLARGESTVALUES(L, k)
```

▶ Figure 13.6 Basic feature selection algorithm for selecting the k best features.

↳ 3 approaches to feature selection

1. Mutual information $A(t,c) = I(U_t ; C_c)$
2. Chi squared method $A(t,c) = X^2(t,c)$ — X^2
3. Frequency based features $A(t,c) = N(t,c)$

1. Mutal Information $A(t,c) = I(U_t ; C_c)$

↳ compute $A(t,c)$ as the expected MI (Mutual Information)
of term t and class c

↳ It measure how much information
the presence/absence of a term
contributes to making correct classification
decision on c

$$I(U;C) = \sum_{e_t \in \{1,0\}} \sum_{e_c \in \{1,0\}} P(U = e_t, C = e_c) \log_2 \frac{P(U = e_t, C = e_c)}{P(U = e_t)P(C = e_c)},$$

$$\begin{aligned} I(U;C) &= \frac{N_{11}}{N} \log_2 \frac{NN_{11}}{N_1.N_{.1}} + \frac{N_{01}}{N} \log_2 \frac{NN_{01}}{N_0.N_{.1}} \\ &\quad + \frac{N_{10}}{N} \log_2 \frac{NN_{10}}{N_1.N_{.0}} + \frac{N_{00}}{N} \log_2 \frac{NN_{00}}{N_0.N_{.0}} \end{aligned}$$

Mutual Information

Example 13.3: Consider the class *poultry* and the term export in Reuters-RCV1. The counts of the number of documents with the four possible combinations of indicator values are as follows:

	$e_c = e_{poultry} = 1$	$e_c = e_{poultry} = 0$
$e_t = e_{export} = 1$	$N_{11} = 49$	$N_{10} = 27{,}652$
$e_t = e_{export} = 0$	$N_{01} = 141$	$N_{00} = 774{,}106$

After plugging these values into Equation (13.17) we get: (sum of these)

$$\begin{aligned} I(U;C) &= \frac{49}{801{,}948} \log_2 \frac{801{,}948 \cdot 49}{(49+27{,}652)(49+141)} \\ &\quad + \frac{141}{801{,}948} \log_2 \frac{801{,}948 \cdot 141}{(141+774{,}106)(49+141)} \\ &\quad + \frac{27{,}652}{801{,}948} \log_2 \frac{801{,}948 \cdot 27{,}652}{(49+27{,}652)(27{,}652+774{,}106)} \\ &\quad + \frac{774{,}106}{801{,}948} \log_2 \frac{801{,}948 \cdot 774{,}106}{(141+774{,}106)(27{,}652+774{,}106)} \\ &\approx 0.0001105 \end{aligned}$$

Mutual Information – Example

UK		China		poultry	
london	0.1925	china	0.0997	poultry	0.0013
uk	0.0755	chinese	0.0523	meat	0.0008
british	0.0596	beijing	0.0444	chicken	0.0006
stg	0.0555	yuan	0.0344	agriculture	0.0005
britain	0.0469	shanghai	0.0292	avian	0.0004
plc	0.0357	hong	0.0198	broiler	0.0003
england	0.0238	kong	0.0195	veterinary	0.0003
pence	0.0212	xinhua	0.0155	birds	0.0003
pounds	0.0149	province	0.0117	inspection	0.0003
english	0.0126	taiwan	0.0108	pathogenic	0.0003
coffee		**elections**		**sports**	
coffee	0.0111	election	0.0519	soccer	0.0681
bags	0.0042	elections	0.0342	cup	0.0515
growers	0.0025	polls	0.0339	match	0.0441
kg	0.0019	voters	0.0315	matches	0.0408
colombia	0.0018	party	0.0303	played	0.0388
brazil	0.0016	vote	0.0299	league	0.0386
export	0.0014	poll	0.0225	beat	0.0301
exporters	0.0013	candidate	0.0202	game	0.0299
exports	0.0013	campaign	0.0202	games	0.0284
crop	0.0012	democratic	0.0198	team	0.0264

► **Figure 13.7** Features with high mutual information scores for six Reuters-RCV1 classes.

2. Chi Square Method $A(t,c) = X^2(t,c)$

↳ 2 events are the occurence of the
term and class

↳ terms are ranked w.r.t the following quantity

$$X^2(\mathbb{D}, t, c) = \sum_{e_t \in \{0,1\}} \sum_{e_c \in \{0,1\}} \frac{(N_{e_t e_c} - E_{e_t e_c})^2}{E_{e_t e_c}}$$

↳ X^2 is a measure of how much
expected counts E and observed counts N → O
deviate from each other

↳ high value of X^2
↳ indicates the hypothesis of independence
↳ which implies that expected and observed counts are similar, is incorrect

Steps

↳ find each RT and CT

↳ $E = \frac{RT \times CT}{T}$

↳ sum of all $\frac{(O-E)^2}{E}$

MUTUAL INFO

Example 13.3: Consider the class *poultry* and the term export in Reuters-RCV1. The counts of the number of documents with the four possible combinations of indicator values are as follows:

	$e_c = e_{poultry} = 1$	$e_c = e_{poultry} = 0$
$e_t = e_{export} = 1$	$N_{11} = 49$	$N_{10} = 27{,}652$
$e_t = e_{export} = 0$	$N_{01} = 141$	$N_{00} = 774{,}106$

After plugging these values into Equation (13.17) we get:

$$\begin{aligned} I(U;C) &= \frac{49}{801{,}948}\log_2\frac{801{,}948 \cdot 49}{(49+27{,}652)(49+141)} \\ &+ \frac{141}{801{,}948}\log_2\frac{801{,}948 \cdot 141}{(141+774{,}106)(49+141)} \\ &+ \frac{27{,}652}{801{,}948}\log_2\frac{801{,}948 \cdot 27{,}652}{(49+27{,}652)(27{,}652+774{,}106)} \\ &+ \frac{774{,}106}{801{,}948}\log_2\frac{801{,}948 \cdot 774{,}106}{(141+774{,}106)(27{,}652+774{,}106)} \\ &\approx 0.0001105 \end{aligned}$$

8)

	Poultry	$\overline{\text{Poultry}}$	row Total RT
export	49	27 652	27,701
$\overline{\text{export}}$	141	774106	774,247
Column Total CT	190	801,758	801,948

Export	Poultry	O (Observed Value)	$\frac{O}{T}\log_2\frac{O \times T}{RT \times CT}$
E	P	49	$\frac{49}{801948}\log_2\frac{49 \times 801948}{27701 \times 190} = 5.334 \times 10^{-5}$
E	$\overline{P}$	27,652	$\frac{27652}{801948}\log_2\frac{27652 \times 801948}{27701 \times 801758} = -2.651 \times 10^{-5}$
$\overline{E}$	P	141	$\frac{141}{801948}\log_2\frac{141 \times 801948}{774247 \times 190} = -2.009 \times 10^{-5}$
$\overline{E}$	$\overline{P}$	774,106	$\frac{774106}{801948}\log_2\frac{774106 \times 801948}{774247 \times 801758} = -7.635 \times 10^{-5}$

?

CHI SQUARE METHOD

8)

	Poultry	$\overline{\text{Poultry}}$	row Total RT
export	49	27 652	27,701
$\overline{\text{export}}$	141	774106	774,247
Column Total CT	190	801,758	801,948

$\frac{CT \times RT}{T}$

Export	Poultry	O (Observed Value)	E (Expected Value)	$(O-E)^2$	$(O-E)^2/E$
E	P	49	$\frac{27{,}701 \times 190}{801{,}948} = 6.6$	1797.76	272.38
E	$\overline{P}$	27,652	$\frac{27{,}701 \times 801{,}758}{801{,}948} = 27{,}694.4$	1797.76	0.065
$\overline{E}$	P	141	$\frac{774{,}247 \times 190}{801{,}948} = 183.4$	1797.76	9.802
$\overline{E}$	$\overline{P}$	774,106	$\frac{774247 \times 801758}{801948} = 774063.6$	1849	2×10^{-3}
					284

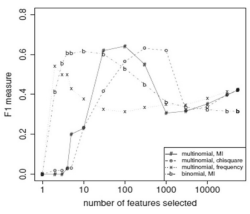
ANS

3. Frequency based features

↳ selecting terms that are most common in classes
  ↳ It selects some frequent terms → that have no specific info about the class

↳ frequency can be defined as
  ↳ doc frequency → no of d in class c that contain the term t → more appropriate for Bernoulli model
  OR
  ↳ collection frequency → no of token of t that occur in docs in C → more appropriate for multinomial model

Features Selection – Experiment



► **Figure 13.8** Effect of feature set size on accuracy for multinomial and Bernoulli models.

Feature Generation

↳ idea is to generate high level features → more abstract from low level features

e.g. sentence dependency graph from sentence

Naïve Bayes

- Very Fast, low storage requirements
- Robust to Irrelevant Features

  Irrelevant Features cancel each other without affecting results

- Very good in domains with many equally important features

  Decision Trees suffer from *fragmentation* in such cases – especially if little data

- Optimal if the independence assumptions hold: If assumed independence is correct, then it is the Bayes Optimal Classifier for problem
- A good dependable baseline for text classification

Vector Space Model

- A document is represented as a feature vector in feature-dimensional space.
- Distance is generally the notions of similarity for this model.
- General assumptions for classification:
  - Contiguity hypothesis: Documents in the same class form a contiguous region and regions of different classes do not overlap.

Classification –Vector Space Model

- Corpus-Classification Dataset
  - Pre-processing
- Feature Selection & weighting
- Similarity function
- Classification algorithm
- Performance evaluation

Rocchio's Algorithm

↳ can be used for text classification

↳ It divides vector space into regions centered on centroids/prototypes → center of mass of all docs in the class

one for each class

PROS

↳ simple and efficient

CON

↳ inaccurate if classes are not approximately spheres with similar radii

Rocchio's Algorithm

TRAINROCCHIO(C, D)
1 **for each** $c_j \in C$
2 **do** $D_j \leftarrow \{d : \langle d, c_j \rangle \in D\}$
3 $\quad \vec{\mu}_j \leftarrow \frac{1}{|D_j|} \sum_{d \in D_j} \vec{v}(d)$
4 **return** $\{\vec{\mu}_1, \ldots, \vec{\mu}_J\}$

APPLYROCCHIO($\{\vec{\mu}_1, \ldots, \vec{\mu}_J\}, d$)
1 **return** $\arg\min_j |\vec{\mu}_j - \vec{v}(d)|$

Rocchio's Algorithm Example

	Doc	Words	Class
Training	1	Chinese Beijing Chinese	c
	2	Chinese Chinese Shanghai	c
	3	Chinese Macao	c
	4	Tokyo Japan Chinese	j
Test	5	Chinese Chinese Chinese Tokyo Japan	?

Dictionary = <bejing, chinese, japan, macao,shanghai,tokyo>
Doc#1 < 1,2,0,0,0,0>
Doc#2 <0,2,0,0,1,0>
Doc#3 <0,1,0,1,0,0>
Doc#4 <0,1,1,0,0,1>
Doc#5 <0,3,0,0,0,1> ← term vectors

μ_c = 1/3 (doc1 +doc2+doc3)
μ_c = <1/3,5/3,0,1/3,1/3,0>
μ_j = doc4 = < 0,1,1,0,0,1>
Doc5 = < 0,3,0,0,0,1>

Distance (Doc5, μ_c) = 5
Distance (Doc5, μ_j) = 4
Doc5 belong to class μ_j

smallest

μ_c: sum of all docs of c / no of docs in c

dot product, with no magnitude division

$(Ds, \mu_c) \cdot \frac{5}{3} \times 3 = 5$

$(Ds, \mu_j) \cdot 3 \times 1 + 1 \times 1 = 4$

1	2	0	0	0	0	
0	2	0	0	1	0	
0	1	0	1	0	0	
1	5	0	1	1	0	/3

k-Nearest Neighbour Learning (KNN)

↳ example based learning

↳ memory based learning

↳ where learning is storing the representation of the training eg in D

↳ Testing instance x

↳ compute similarity b/w x and all eg in D

↳ assign x the category of most similar eg in D

↳ It doesn't explicitly calculate a class prototype descriptor

PROS

↳ simple, easy to implement

↳ only 1 hyper parameter

↳ robust to noisy training

CON

↳ sensitive to

↳ value of k

↳ distance function

↳ noisy data

↳ lazy learner → no precompute model

↳ large computation time for large dataset → but no training time

k-NN Classification Algorithm

- Training
  - For each training example $<d_i, C_j> \varepsilon D_{train}$
  - Compute the corresponding Feature vector -> d_i, for document d_i
- Testing
  - Computer vector for d_j using the same feature vector
  - For each $<d_i, C_j> \varepsilon D_{train}$ calculate x[i]=Cosine(d_j,d_i), sort x[] by decreasing value.
  - Let N be the closest (i.e. first) k examples in D. (get k most similar neighbors) Return the majority class of examples in N

k-NN Classification

TRAIN-KNN(C, $\mathbb{D}$)
1 $\mathbb{D}' \leftarrow$ PREPROCESS($\mathbb{D}$)
2 $k \leftarrow$ SELECT-K(C, $\mathbb{D}'$)
3 **return** $\mathbb{D}', k$

APPLY-KNN(C, $\mathbb{D}', k, d$)
1 $S_k \leftarrow$ COMPUTENEARESTNEIGHBORS($\mathbb{D}', k, d$)
2 **for each** $c_j \in \mathbb{C}$
3 **do** $p_j \leftarrow |S_k \cap c_j|/k$
4 **return** $\arg\max_j p_j$

Example

	Doc	Words	Class
Training	1	Chinese Beijing Chinese	c
	2	Chinese Chinese Shanghai	c
	3	Chinese Macao	c
	4	Tokyo Japan Chinese	j
Test	5	Chinese Chinese Chinese Tokyo Japan	?

Dictionary = <bejing, chinese, japan, macao,shanghai,tokyo>
Doc#1 < 1,2,0,0,0,0>
Doc#2 <0,2,0,0,1,0>
Doc#3 <0,1,0,1,0,0>
Doc#4 <0,1,1,0,0,1>
Doc#5 <0,3,0,0,0,1>
↳ term vector

cosine similarity
Cos(d1,d5) = dot-product (d1,d5) ÷ |d1| |d5|
Cos(d1,d5) = 0.848
Cos(d2,d5) =0.848
Cos(d3,d5) =0.424
Cos(d4,d5) =0.953
3 largest

For 1-NN d5 will belong to j
For 3-NN d5 will belong to c
big

1-NN:
Imagine Boundry of 1
Label around D5 (which is D4 i.e. the nearest)
so it belongs to J

3-NN:
Imagine a boundary of 3 labels around D5 (which are D1, D2, D4 i.e. again nearest)
so class C docs are more than C' so it belongs to C

Agenda

- Ad hoc vs. Standing Query
- Text/Document Classification
- Naïve Bayes Classification
- Variations of Naïve Bayes Classification
- Evaluation of Classification
- Classification model & Accuracy
- Feature Selection
  - Mutual Information
  - Chi Square Method
  - Frequency based feature
- Conclusion

Conclusion

- Classification Model learning is a challenging task.
- Textual documents are rich in features.
- Feature Selection is helpful in simplifying the model and improving accuracy of the model.
- Evaluation of classification task is challenging as well.
- Chapter 13 and 14 from the textbook.

Sentiment → small part of emotion

emotion

↳ Binary classification

↳ Multiclass classification — News → Business / Political / entertainment

↳ fixed classes

↳ Multilabel classification — image → mountain, sea

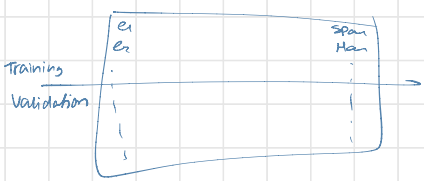
↳ more than one

↳ simultaneous

↳ Multi instance classification

↳ Extreme classification

↳ many labels



YOLO

assume training set is balanced
50% spam
50% ham

TEXT CLUSTERING CHP 16,17

Clustering

↳ unsupervised ML technique

↳ It automatically seperates similar objects in a heterogenous collection

↳ Objective function used to

↳ Optimise the similarity b/w objects in a cluster

↳ reduce the similarity b/w different clusters

Clustering

- It is a challenging problem, as we will be doing dual optimization: High Intra-cluster similarity and Low Inter-cluster similarity
- Document clustering is a special problem, where objects to clusters are documents.
  - Natural Language Text
  - Knowing exact number of distinct groups(clusters)
  - Producing an optimal clustering arrangement
  - Label the groups
  - Evaluation of clustering results

Clustering VS Classification

↳ unsupervised learning

↳ Clusters are inferred from the data without human input

↳ supervised learning

↳ classes are human defined and part of input to the learning algo

Clustering Hypothesis

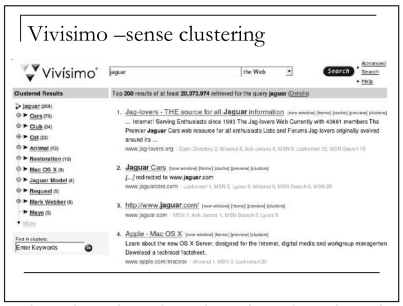
↳ docs in the same cluster behave similarly w.r.t relevance to info needs

- All applications of clustering in IR are based (directly or indirectly) on the cluster hypothesis.
- Van Rijsbergen's original wording: "closely associated documents tend to be relevant to the same requests".

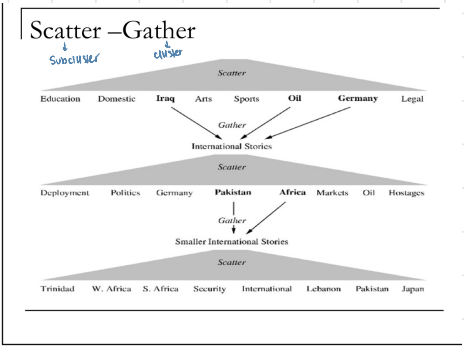
Clustering Applications

Application	What is clustered?	Benefit
Search result clustering	search results	more effective information presentation to user
Scatter-Gather	(subsets of) collection	alternative user interface: "search without typing"
Collection clustering	collection	effective information presentation for exploratory browsing
Cluster-based retrieval	collection	higher efficiency: faster search

Vivisimo –sense clustering



Scatter –Gather



Hard clustering
- each doc only belongs to "1" cluster

Soft clustering
- a doc can belong to more than 1 cluster
- Fuzzy membership in more than 1 cluster

Flat clustering Algos
- usually start with a random partitioning of docs in groups
- refine iteratively
- Main Algo: K-means

METHOD
- compute a partion of N docs into a set of K clusters

GIVEN:
- a set of docs
- the number k

FIND:
- a partition into k clusters that optimises the chosen partitioning criterion

GLOBAL OPTIMIZATION:
- exhaustively enumerate partions
- Pick optimal one → an NP hard

EFFETIVE HEURISTIC METHOD:
- k-means Algo

VS

Hierarchial clustering Algos
- creates a hierarchy
  - all possible cluster levels
- bottom up, agglomerative
- top down, divisive
- main Algo: HAC

K-mean clustering

best clustering algo

- used as baseline for clustering docs
- there are 3 main steps
  1. Representation of docs
  2. Similarity measure
  3. clustering Approach

Similarity measure

aka similarity function

- it maps 2 objects into a real value b/w (0-1)
  - identical objs → value 1
  - totally diff pair → lower value

- **Equal self-similarity.** d(A, A) = d(B, B) for all points A and B. Therefore, s(A, A) = s(B, B) for all stimuli A and B.
- **Minimality.** d(A, B) > d(A, A) for all points A ≠ B. Therefore, s(A, B) < s(A, A) for all stimuli A ≠ B.
- **Symmetry.** d(A, B) = d(B, A) for all points A and B. Therefore, s(A, B) = s(B, A) for all stimuli A and B.
- **Triangle Inequality.** d(A, B) + d(B, C) ≥ d(A, C) for all points A, B, and C.

K-mean DOC clustering

- it uses a VSM to represent docs in feature dimensional space
- uses Euclidean distance to measure similarity among docs

almost equivalent to cosine similarity

METHOD

- start with an initial k-docs
- for each other doc compute distance from these k docs
- the smallest distance doc from the k-doc is assigned to cluster

k-Mean

K-MEANS$(\{\vec{x}_1, \ldots, \vec{x}_N\}, K)$
1 $(\vec{s}_1, \vec{s}_2, \ldots, \vec{s}_K) \leftarrow$ SELECTRANDOMSEEDS$(\{\vec{x}_1, \ldots, \vec{x}_N\}, K)$
2 **for** $k \leftarrow 1$ **to** K
3 **do** $\vec{\mu}_k \leftarrow \vec{s}_k$
4 **while** stopping criterion has not been met
5 **do for** $k \leftarrow 1$ **to** K
6 **do** $\omega_k \leftarrow \{\}$
7 **for** $n \leftarrow 1$ **to** N
8 **do** $j \leftarrow \arg\min_{j'} |\vec{\mu}_{j'} - \vec{x}_n|$
9 $\omega_j \leftarrow \omega_j \cup \{\vec{x}_n\}$ *(reassignment of vectors)*
10 **for** $k \leftarrow 1$ **to** K
11 **do** $\vec{\mu}_k \leftarrow \frac{1}{|\omega_k|} \sum_{\vec{x} \in \omega_k} \vec{x}$ *(recomputation of centroids)*
12 **return** $\{\vec{\mu}_1, \ldots, \vec{\mu}_K\}$

Convergence of K-Means

First, there are at most k^N ways to partition N data points into k clusters; each such partition can be called a "clustering". This is a large but finite number. For each iteration of the algorithm, we produce a new clustering based *only* on the old clustering. Notice that

1. if the old clustering is the same as the new, then the next clustering will again be the same.
2. If the new clustering is different from the old then the newer one has a lower cost

Since the algorithm iterates a function whose domain is a finite set, the iteration must eventually enter a cycle. The cycle cannot have length greater than 1, because otherwise by (2) you would have some clustering which has a lower cost than itself which is impossible. Hence the cycle must have length exactly 1. Hence k-means converges in a finite number of iterations.

3 stopping criteria for k-mean

1. few iteration
2. if no some clusters candidate want to change cluster
3. using threshold on intra compact on clusters

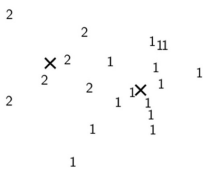
K-means is guaranteed to converge: Proof

- RSS = sum of all squared distances between document vector and closest centroid
- RSS decreases during each reassignment step.
  - because each vector is moved to a closer centroid
- RSS decreases during each recomputation step.
  - see next slide
- There is only a finite number of clusterings.
- Thus: We must reach a fixed point.
- Assumption: Ties are broken consistently.

Worked Example: Random selection of initial centroids

Exercise: (i) Guess what the optimal clustering into two clusters is in this case; (ii) compute the centroids of the clusters

Worked Ex.: Centroids and assignments after convergence



K-means is guaranteed to converge: Proof

- RSS = sum of all squared distances between document vector and closest centroid
- RSS decreases during each reassignment step.
  - because each vector is moved to a closer centroid
- RSS decreases during each recomputation step.
  - see next slide
- There is only a finite number of clusterings.
- Thus: We must reach a fixed point.
- Assumption: Ties are broken consistently.

Optimality of *K*-means

- Convergence does not mean that we converge to the optimal clustering!
- This is the great weakness of K-means.
- If we start with a bad set of seeds, the resulting clustering can be horrible.

Initialization of *K*-means

- Random seed selection is just one of many ways K-means can be initialized.
- Random seed selection is not very robust: It's easy to get a suboptimal clustering.
- Better ways of computing initial centroids:
  - Select seeds not randomly, but using some heuristic (e.g., filter out outliers or find a set of seeds that has "good coverage" of the document space)
  - Use hierarchical clustering to find good seeds
  - Select i (e.g., $i = 10$) different random sets of seeds, do a K-means clustering for each, select the clustering with lowest RSS

Time complexity of *K*-means

- Computing one distance of two vectors is $O(M)$.
- Reassignment step: $O(KNM)$ (we need to compute KN document-centroid distances)
- Recomputation step: $O(NM)$ (we need to add each of the document's < M values to one of the centroids)
- Assume number of iterations bounded by I
- Overall complexity: $O(IKNM)$ – linear in all important dimensions
- However: This is not a real worst-case analysis.
- In pathological cases, complexity can be worse than linear.

How many clusters?

- Number of clusters *K* is given in many applications.
  - E.g., there may be an external constraint on *K*. Example: In the case of Scatter-Gather, it was hard to show more than 10–20 clusters on a monitor in the 90s.
- What if there is no external constraint? Is there a "right" number of clusters?
- One way to go: define an optimization criterion
  - Given docs, find *K* for which the optimum is reached.
  - What optimiation criterion can we use?
  - We can't use RSS or average squared distance from centroid as criterion: always chooses *K* = *N* clusters.

Exercise

- Your job is to develop the clustering algorithms for a competitor to news.google.com
- You want to use *K*-means clustering.
- How would you determine *K*?

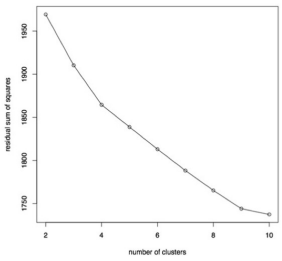
Simple objective function for *K* (1)

- Basic idea:
  - Start with 1 cluster (*K* = 1)
  - Keep adding clusters (= keep increasing *K*)
  - Add a penalty for each new cluster
- Trade off cluster penalties against average squared distance from centroid
- Choose the value of *K* with the best tradeoff

Simple objective function for *K* (2)

- Given a clustering, define the cost for a document as (squared) distance to centroid
- Define total distortion RSS(K) as sum of all individual document costs (corresponds to average distance)
- Then: penalize each cluster with a cost λ
- Thus for a clustering with *K* clusters, total cluster penalty is *K*λ
- Define the total cost of a clustering as distortion plus total cluster penalty: RSS(K) + *K*λ
- Select K that minimizes (RSS(K) + *K*λ)
- Still need to determine good value for λ . . .

Finding the "knee" in the curve



Pick the number of clusters where curve "flattens". Here: 4 or 9.

K-Mean Clustering

- Performance is determined
  - by initialization
  - appropriate distance measure

CONS

- Sensitive to the initial seeds (if we start with bad seed, resulting clustering can be horrible)
- difficult to compare results
- fixed no of clusters
  - difficult to predict actual no's from the data set

K-mean variants to overcome weakness

1. K-Medoids → resistance to noise/outliers
2. K-Modes → extention to categorical data clustering analysis
3. CLARA → extention to deal with large data sets
4. Mixture Models → handling uncertainty of clusters
   - EM Algo

PROS

- forms good clusters in less iterations

Evaluating Clustering Task

(It is challenging)

- Internal
  - only data is used to evaluate clustering results
- External
  - evaluation w.r.t human defined classification (gold standard query)

Good Clustering

Internal criteria

- Produce high quality clusters in which
  - intra class similarity is high
  - inter class similarity is low

e.g. RSS in KMeans

- The measured quality of a clustering depends on both the document representation and the similarity measure used

External criteria

- based on gold standard dataset
- Purity
- Rand
- F-Measure
- NMI

- Based on a gold standard data set, e.g., the Reuters collection we also used for the evaluation of classification
- Goal: Clustering should reproduce the classes in the gold standard
- (But we only want to reproduce how documents are divided into groups, not the class labels.)
- First measure for how well we were able to reproduce the classes: Purity

3 ways to evaluate cluster

- Inter cluster
- Intra cluster → Internal
- Gold standard → External

Purity

→ evaluates clustering using ground truth

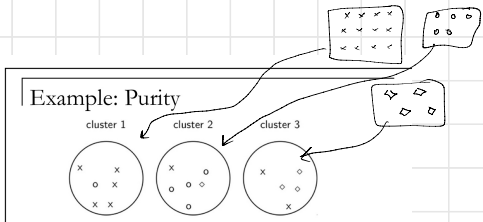
↳ make cluster of data → K mean

↳ is between 0 and 1

↳ 0 → all clusters diff

↳ 1 → clusters found ground truth

$$\text{Purity} = \frac{1}{\text{no of docs}} \times \left(\sum \max(\text{ground truth in cluster}) \right)$$



- To compute purity:
  - 5 = $\max_j |\omega_1 \cap c_j|$ (class x, cluster 1);
  - 4 = $\max_j |\omega_2 \cap c_j|$ (class o, cluster 2); and
  - 3 = $\max_j |\omega_3 \cap c_j|$ (class ◇, cluster 3).
- Purity is (1/17) × (5 + 4 + 3) ≈ 0.71.

External criterion: Purity

- $\Omega = \{\omega_1, \omega_2, \ldots, \omega_K\}$ is the set of clusters and $C = \{c_1, c_2, \ldots, c_J\}$ is the set of classes.
- For each cluster ω_k : find class c_j with most members n_{kj} in ω_k
- Sum all n_{kj} and divide by total number of points

$$\text{purity}(\Omega, C) = \frac{1}{N} \sum_k \max_j |\omega_k \cap c_j|$$

Random index

→ no need for ground truth

↳ alternate view to Purity

↳ view it as a series of decisions

one for each of the $N(N-1)/2$ pairs of docs in the collection

↳ True Positive (TP)

↳ assigns 2 similar docs to the same cluster

↳ True Negative (TN)

↳ assign 2 dissimilar docs to diff clusters

↳ 2 types of possible error

1. FP

↳ decision assigning 2 dissimilar docs to same cluster

2. FN

↳ decision assign 2 similar docs to diff clusters

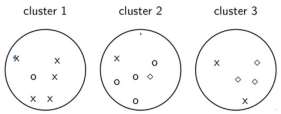
Random Index

- Definition: $\text{RI} = \frac{\text{TP}+\text{TN}}{\text{TP}+\text{FP}+\text{FN}+\text{TN}}$
- Based on 2x2 contingency table of all pairs of documents:

	same cluster	different clusters
same class	true positives (TP)	false negatives (FN)
different classes	false positives (FP)	true negatives (TN)

- TP+FN+FP+TN is the total number of pairs.
- There are $\binom{N}{2}$ pairs for N documents.
- Example: $\binom{17}{2}$ = 136 in o/◇/x example
- Each pair is either positive or negative (the clustering puts the two documents in the same or in different clusters) . . .
- . . . and either "true" (correct) or "false" (incorrect): the clustering decision is correct or incorrect.

Example: Random Index



- Consider the same example once again, this time we need to calculate Random Index (RI)

$$TP + FP: \quad C\left(\frac{\text{total doc in cluster}}{N-1}\right) + \ldots$$

total no of docs

$$TP: \quad C\left(\frac{\text{total doc with similar pair}}{N-1}\right) + \ldots$$

$$FP: TP + FP - TP$$

Example: Random Index

As an example, we compute RI for the o/◇/x example. We first compute TP + FP. The three clusters contain 6, 6, and 5 points, respectively, so the total number of "positives" or pairs of documents that are in the same cluster is:

combination

$$TP + FP = \binom{6}{2} + \binom{6}{2} + \binom{5}{2} = 40$$

remaining cluster

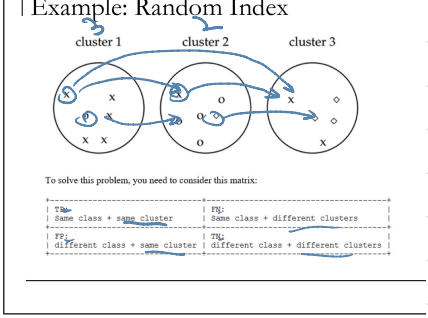
Of these, the x pairs in cluster 1, the o pairs in cluster 2, the ◇ pairs in cluster 3, and the x pair in cluster 3 are true positives:

$$TP = \binom{5}{2} + \binom{4}{2} + \binom{3}{2} + \binom{2}{2} = 20$$

Thus, FP = 40 − 20 = 20. FN and TN are computed similarly.

5:20

Example: Random Index



To solve this problem, you need to consider this matrix:

TP: Same class + same cluster	FN: Same class + different clusters
FP: different class + same cluster	TN: different class + different clusters

TP SS
FP DS
FN SD
TN DD

Example: Random Index

Same class and in the same cluster

$$TP = \binom{5}{2} + \binom{4}{2} + \binom{2}{2} + \binom{3}{2} = 10 + 6 + 1 + 3 = 20$$

Same classes, but different clusters

$$FN = \binom{5}{1}\binom{1}{1} + \binom{5}{1}\binom{2}{1} + \binom{1}{1}\binom{4}{1} + \binom{2}{1}\binom{2}{1} + \binom{1}{1}\binom{3}{1} = 5 + 10 + 4 + 2 + 3 = 24$$

Different classes but in the same cluster

$$FP = \binom{5}{1}\binom{1}{1} + \binom{4}{1}\binom{1}{1} + \binom{4}{1}\binom{1}{1} + \binom{1}{1}\binom{1}{1} + \binom{3}{1}\binom{2}{1} = 5 + 4 + 4 + 1 + 6 = 20$$

c1 c2 c3

Rand measure for the o/◇/x example

	same cluster	different clusters
same class	TP = 20	FN = 24
different classes	FP = 20	TN = 72

RI is then

(20 + 72)/(20 + 20 + 24 + 72) ≈ 0.68.

Evaluation results for the o/◇/x example

	purity	NMI	RI	F_5
lower bound	0.0	0.0	0.0	0.0
maximum	1.0	1.0	1.0	1.0
value for example	0.71	0.36	0.68	0.46

All four measures range from 0 (really bad clustering) to 1 (perfect clustering).

External evaluation measures

Normalized mutual information (NMI)

↳ how much info clustering contains about the classification

↳ singleton clusters → no of clusters = no of docs

↳ have max MI

therefore normalize by entropy of clusters and classes

F measure

↳ like Rand, BUT

Precision and recall can be weighted

Hierarchical Agglomerative clustering

→ bottom up

↳ assumes similarity functions

to determine similarity of two instances

Step?

↳ starts with all instances in a seperate cluster

↳ repeatedly joins the two clusters that are most similar

until there is only 1 cluster

↳ The history of merging forms a

binary tree / hierarchy

Example

```
SIMPLEHAC(d_1, ..., d_N)
 1  for n ← 1 to N
 2  do for i ← 1 to N
 3     do C[n][i] ← SIM(d_n, d_i)
 4     I[n] ← 1  (keeps track of active clusters)
 5  A ← []  (collects clustering as a sequence of merges)
 6  for k ← 1 to N − 1
 7  do (i, m) ← arg max_{(i,m): i≠m ∧ I[i]=1 ∧ I[m]=1} C[i][m]
 8     A.APPEND((i, m))  (store merge)
 9     for j ← 1 to N
10     do C[i][j] ← SIM(i, m, j)
11        C[j][i] ← SIM(i, m, j)
12     I[m] ← 0  (deactivate cluster)
13  return A
```

Figure 17.2 A simple, but inefficient HAC algorithm.

Agglomerative clustering

↳ bottoms up method

↳ starts with each eg in its own cluster

Iteratively combines mem

↳ to form larger clusters

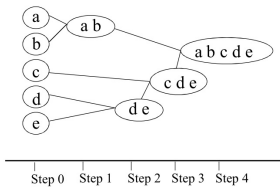
Divisive clustering

↳ Partitional / top down method

↳ seperates all eg immediately into clusters

Hierarchical Clustering

- Agglomerative approach

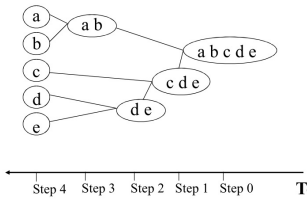


Initialization:
Each object is a cluster
Iteration:
Merge two clusters which are most similar to each other;
Until all objects are merged into a single cluster

bottom-up

Hierarchical Clustering

- Divisive Approaches

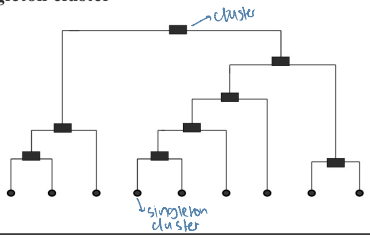


Initialization:
All objects stay in one cluster
Iteration:
Select a cluster and split it into two sub clusters
Until each leaf cluster contains only one object

Top-down

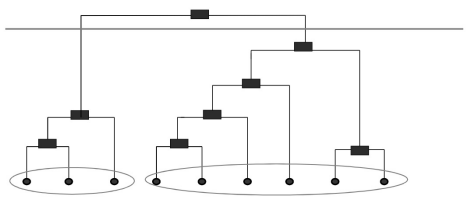
Dendrogram

- A binary tree that shows how clusters are merged/split hierarchically
- Each node on the tree is a cluster; each leaf node is a singleton cluster



Dendrogram

- A clustering of the data objects is obtained by cutting the *dendrogram* at the desired level, then each connected component forms a cluster

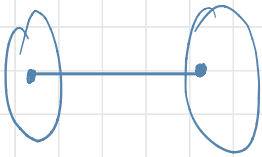


VARIATIONS OF HAC

1. Single link clustering

aka connectivity

↳ considers distance b/w 2 clusters to be equal
to the shortest distance from any member of 1 cluster
to any member of the other cluster



↳ If data has similarities
the similarity b/w a pair of clusters = greatest similarity from any member of 1 cluster to any member of the other cluster

2. Complete link clustering

aka the diameter

↳ considers distance b/w 2 clusters to be equal
to the longest distance from any member of 1 cluster
to any member of the other cluster

3. Average link clustering

aka minimum variance method

↳ considers distance b/w 2 clusters to be equal
to the average distance from any member of 1 cluster
to any member of the other cluster

Variations of HAC

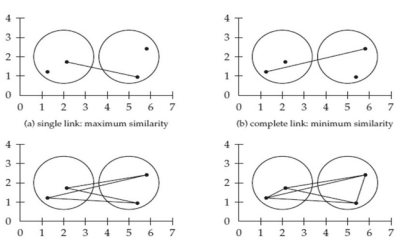


Figure 17.3 The different notions of cluster similarity used by the four HAC algorithms. An *inter-similarity* is a similarity between two documents from different clusters.

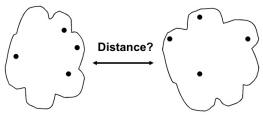
Computational Complexity

- In the first iteration, all HAC methods need to compute similarity of all pairs of n individual instances which is $O(n^2)$.
- In each of the subsequent $n-2$ merging iterations, compute the distance between the most recently created cluster and all other existing clusters.
- In order to maintain an overall $O(n^2)$ performance, computing similarity to each cluster must be done in constant time.
  - Else $O(n^2 \log n)$ or $O(n^3)$ if done naively

How to Merge Clusters?

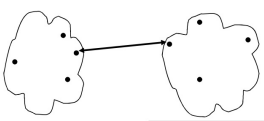
- How to measure the distance between clusters?

- Single-link
- Complete-link
- Average-link
- Centroid distance



Hint: ***Distance between clusters*** is usually defined on the basis of ***distance between objects.***

How to Define Inter-Cluster Distance

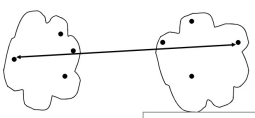


- Single-link
- Complete-link
- Average-link
- Centroid distance

$$d_{\min}(C_i, C_j) = \min_{p \in C_i, q \in C_j} d(p,q)$$

The distance between two clusters is represented by the distance of the *closest pair of data objects* belonging to different clusters.

How to Define Inter-Cluster Distance

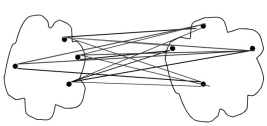


- Single-link
- Complete-link
- Average-link
- Centroid distance

$$d_{\min}(C_i, C_j) = \max_{p \in C_i, q \in C_j} d(p,q)$$

The distance between two clusters is represented by the distance of the *farthest pair of data objects* belonging to different clusters.

How to Define Inter-Cluster Distance

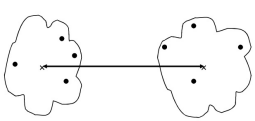


- Single-link
- Complete-link
- Average-link
- Centroid distance

$$d_{\min}(C_i, C_j) = avg_{p \in C_i, q \in C_j} d(p,q)$$

The distance between two clusters is represented by the *average* distance of *all pairs of data objects* belonging to different clusters.

How to Define Inter-Cluster Distance



m_i,m_j are the means of C_i, C_j,

- Single-link
- Complete-link
- Average-link
- Centroid distance

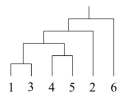
$$d_{mean}(C_i, C_j) = d(m_i, m_j)$$

The distance between two clusters is represented by the distance between *the means of the cluters.*

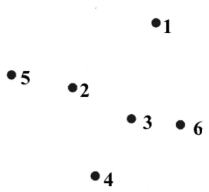
An Example of the Agglomerative Hierarchical Clustering Algorithm

- For the following data set, we will get different clustering results with the single-link and complete-link algorithms.

Result of the Single-Link algorithm → shortest

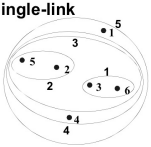


Result of the Complete-Link algorithm → longest

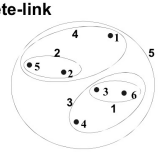


Hierarchical Clustering: Comparison

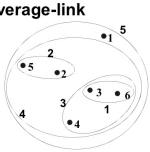
Single-link



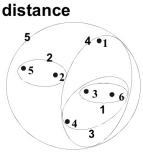
Complete-link



Average-link

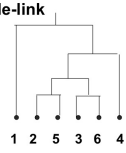


Centroid distance

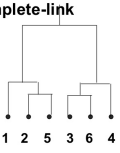


Compare Dendrograms

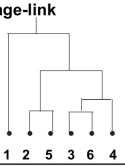
Single-link



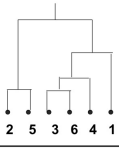
Complete-link



Average-link



Centroid distance



a. Differentiate between k-mean and HAC approaches to clustering. [5]

K-Mean	HAC
It is a partition clustering method. It produces a non-overlapping partition.	It is a hierarchical clustering method. It produces a hierarchy of clusters.
It takes k as number of clusters to be produced.	It performs merging of clusters to produce hierarchy, you can cut at a desire level to get the clusters.
The running time is proportional to number of data points for clustering.	The running time is polynomial.

Question No. 2

Consider the following data points in 2D space.

P1(3,1), P2(1,2), P3(1,1), P4(8,1), P5(7,3), P6(6,5) taking P2 and P5 as initial seeds – run vanilla k-means algorithm for two iterations to find the resultant clusters C1 and C2.

First Iteration:

C1= P2(1,2)
C2= P5 (7,3)
P1 is close to P2 than P5, similarly P3 is close to P2 than P5
P4 is close to P5 than P2. Hence

C1={ P1,P2, P3} = {5/3, 4/3)
C2={ P4, P5, P6} =(7, 3)

Distance Formula

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

First Iteration:

C1 (5/3, 4/3)
C2= (7,3)
P1 is close to C1 than C2, similarly P3 is close to C1 than C2
P4 is close to P5 than P2. Hence

C1={ P1, P2, P3} = {5/3, 4/3)
C2={ P4,P5, P6} = (7, 3)

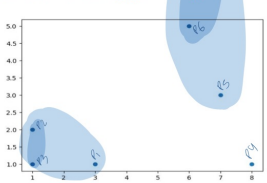
Question No. 2

Consider the following data points in 2D space.

P1(3,1), P2(1,2), P3(1,1), P4(8,1), P5(7,3), P6(6,5), P7(6,7)

Apply Bottom-Up version of Hierarchical Agglomerative Clustering (HAC) for the given data points, using Single Link approach for merging. Show each step of the algorithm

Consider the following plot in2 D space with the given points:



For Bottom Up Hierarchal clustering, Point P3 and P2 are more similar and will be merged first.

Second Point P1 will be merged with the cluster C1(P2, P3) to produced C2(P1, C1).

Thirdly, we will merge P6 and P7 to produced C3(P6, P7) It will merge later P5 to produced C4(C3, P5). Similarly, it will merge the P4 to it to produced C5(C4, P4) and in the final merge C2 and C5 will be merged.

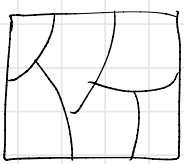
clustering

K Mean Demo + HAC Demo SLIDES MISSING

cluster and object membership is hard → hard clustering
cluster and object membership is numerical → soft cluster → ?

Clustering

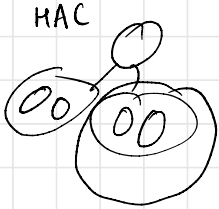
Partition
K mean



↳ no pts common
↳ no object in same cluster

CONS
↳ Partitions can be 2^n which is alot

Hierarchal
HAC



↳ commons in this

Algo Method
↳ take subset of data
↳ plot on k-
↳

Hypothesis Clustering
↳ if 1 doc matches this cluster, then it matches all in this cluster

a. Differentiate between k-Mean and Agglomerative Hierarchical clustering algorithms.

1. Hierarchical and Partitional(k-Mean) Clustering have key differences in running time, assumptions, input parameters and resultant clusters.
2. k-Mean clustering is faster than hierarchical clustering.
3. Hierarchical clustering requires only a similarity measure, while k-Mean clustering requires stronger assumptions such as number of clusters and the initial centers.
4. Hierarchical clustering does not require any input parameters, while k-Mean clustering algorithms require the number of clusters to start running.
5. Hierarchical clustering returns a much more meaningful and subjective division of clusters but k-Mean clustering results in exactly k clusters.
6. Hierarchical clustering algorithms are more suitable for categorical data as long as a similarity measure can be defined accordingly.

ishma hafeez
notes
repsht

WEB SEARCH BASICS & WEB CRAWLING CHP 19, 20

Web Search Basics

- Internet is a Client Server Architecture which provides a bunch of services
- Client
  - a browser
- Server
  - It communicates with the client via HTTP protocol
  - It is lightweight, simple
  - It asynchronously carries a variety of payload (text, images, audio, video) encoded in HTML
  - HTML
    - a markup language for the web
    - connects diff pages and content
- Browser
- HTTP
  - an application protocol for distributed, collaborative systems
  - HTTP header contains a lot of fies for effective transfer of info
  - HTTP/2 is now supported by major webservers and browsers

Web Search Basics

1. Top links to specialized searches
2. Search box
3. Click to search
4. Click to retrieve a single result
5. Link to Advanced Search
6. Click to set search preferences
7. Link to Google's language tools
8. Click to set Google as your browser home page

- Browser
  - The first browser was developed by Tim Berners-Lee in 1990- very limited functionality
  - Mosaic was first GUI based browser in 1993 by Marc Andreesen
  - Marc started Netscape in 1994 and launch Netscape Navigator
  - Microsoft started IE in 1995 for free. 95% market share in 2002
  - Marc started Mozilla foundation and started Firefox in 2004 reached 23% market share in 2011

Web Search – Client Server

Comparison of protocol stack changes delivered with each new version after HTTP/1.0

HTTP/1.1	HTTP/2	HTTP/3
• Some methods and response codes are added. • "Keep-Alive" becomes officially supported. "Host" header becomes supported for Virtual Domain. • Syntax and semantics are separated.	• Support of parallel request transmission by "stream" (elimination of ***HTTP*** HoL Blocking). • Addition of flow-control and prioritization function in units of "stream". • Addition of server-push function (send related file without request.)	• Lower protocol changes from TCP+TLS to UDP+QUIC • Streams and flow-control function are moved to QUIC. • Parallel request transmission is supported by QUIC stream (eliminating ***TCP*** HoL Blocking).

HTTP

↳ 5 groups of status code, which are grouped by 1st digit

1xx → informational

2xx → request was successful

3xx → client redirected to a diff source

4xx → request contains an error

5xx → server encountered an error fulfulling the request

HTTPS

Hyper Text Transfer Protocol Secure

↳ secure version of HTTP

↳ the communication protocol is encrypted using TLS or SSL

TLS: Transport Layer Security

SSL: Secure Socket Layer

pros

↳ customer info is encrypted, cant be intercepted

e.g. credit cards

↳ visitors can verify you are a registered business and that you own the domain

↳ Customers know they are not suppose to visit sites without HTTPS, and therefore, they are more likely to trust and complete purchases from sites that use HTTPS.

Server Side Scripting

↳ a no of server side scripting avl

Client Side Scripting

↳ UI and interaction with local machine

mostly JavaScript

CSS (Cascading Style Sheet)

↳ a language

↳ describes the style of an HTML doc

- HTML
  - HTML 2.0 -1995; HTML 3.0 1997; HTML 4.0 1997
  - HTML 5.0 2014; XHTML vs. XML

HTTP Injection (aka Cross Site Scripting)

↳ a security vulnerability

↳ It allows an attacker to inject HTML code into webpages that are viewed by other users

HTTP Cross Site Scripting

↳ Session Fixation

HTTP Response Splitting

↳ Web application Vulnerability

↳ Web Cache Poisoning

↳ Cross User Defacement

Client-Side Vs. Server Side Scripting

Difference between client-side scripting vs. Server side scripting

Client Side Scripting	Server Side Scripting
The client-side environment used to run scripts is usually a browser.	The **server-side environment** that runs a scripting language is a web server.
The source code is transferred from the web server to the user's computer over the internet and run directly in the browser.	A user's request is fulfilled by running a script directly on the web server to generate dynamic HTML pages. This HTML is then sent to the client browser.
Advantages to client-side scripting including faster response times, a more interactive application, and less overhead on the web server.	The primary advantage to server-side scripting is the ability to highly customize the response based on the user's requirements, access rights, or queries into data stores.
The Disadvantages of client-side scripting are that scripting languages require more time and effort, while the client's browser must support that scripting language.	The disadvantage of server-side processing is the page postback: it can introduce processing overhead that can decrease performance and force the user to wait for the page to be processed and recreated. Once the page is posted back to the server, the client must wait for the server to process the request and send the page back to the client.

Example
`<script>`
`document.getElementById('hello').innerHTML = 'Hello';`
`</script>`
	Example:
	`<h1 id="hello"><?php echo 'Hello'; ?></h1>`

Directories

↳ Taxonomies populated with web pages in categories e.g Yahoo

↳ User to browser through a hierarchical tree of category labels

Search Engines

↳ full text index search engines

e.g Altavista, Excite, Infoseek → Pure searches

↳ user with a keyboard search interface supported by

↳ inverted index

↳ ranking mechanisms → google identified

↳ 2 principles that helped it grow traffic and revenue → by word of mouth

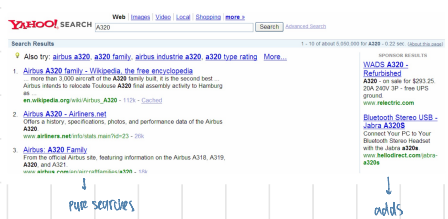
↳ Relevance → focus on precision over recall

↳ Simple interface → gives good user experience

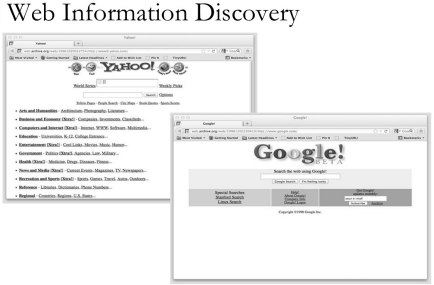
Directories Vs. Search Engines

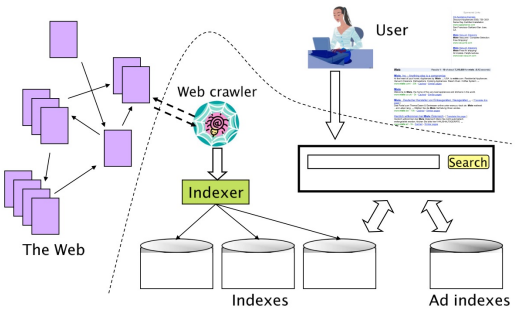
- A directory allows you to explore and get what you want eventually.
- Use a directory to find cooking-related websites.
- Use a directory to find travel guides in a country.
- A search engine brings you to the exact page on the words or phrases you are looking for.
- Use a search engine to find a specific recipe, by providing the name of the ingredients.
- Use a search engine to find the transport trains schedule in Germany

Directory Search (Yahoo)	Text Search (Google)
- Yahoo style, directory search maintains a large hierarchical directory of terms, a user need to follow the topic based hierarchical path to get the information.	- Google style, simple text based search, autonomously crawl and index pages on text terms. The query terms are process through index and relevant pages are returned as result.
- It is very challenging to maintain such large directory	- Indexing is easy as compared to directory maintenance
- Relevant results from exploration of the hierarchical path.	- Relevant results as per the retrieval approach (pagerank).



Web Information Discovery





► **Figure 19.7** The various components of a web search engine.

Web Characteristics

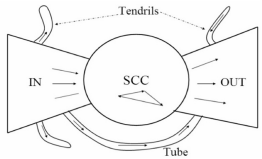
↳ Web User Interaction

↳ Web Graph

↳ Web Spam

Web Characteristics

- Web as a Graph



- There are three major categories of web pages that are IN, OUT and SCC

Web Economic Model

↳ Advertisment Model for Revenue

↳ Unit of measure

↳ CPM, CPC, CPI, CPD, CPP

↳ Complex Advertisment Models

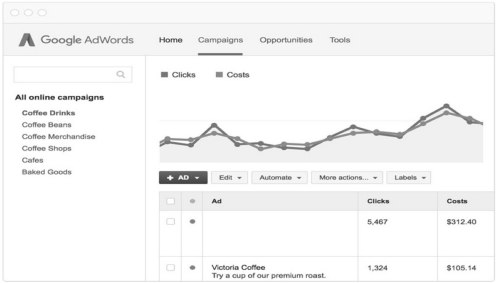
↳ Ad words

↳ Ads

↳ Search Terms

↳ Daily budget

AdWords



User needs

↳ Common web search queries can be grouped in

↳ Informational: general info on a broad topic

e.g. Lukemia, covid

↳ Navigational: seek the website

e.g. United Airlines

↳ Transactional: gives results listing services that provide interface for such transactions

e.g. • Purchase a product
• download a file
• make a reservation

↳ Gray area

- **Gray areas**
  - Find a good hub — Car rental Brazil
  - Exploratory search "see what's there"

User Experience

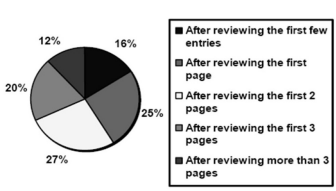
↳ User Queries

↳ 3-4 keywords

↳ rarely uses syntax operators → Free Text Queries

How far do people look for results?

"When you perform a search on a search engine and don't find what you are looking for, at what point do you typically either revise your search, or move on to another search engine? (Select one)"



(Source: iprospect.com WhitePaper_2006_SearchEngineUserBehavior.pdf)

INDEX SIZE & ESTIMATE

Capture/Recapture method

- Suppose that we could pick a random page from the index of E1 and test whether it is in E2's index and symmetrically, test whether a random page from E2 is in E1.
- These experiments give us fractions x and y such that our estimate is that a fraction x of the pages in E1 are in E2, while a fraction y of the pages in E2 are in E1.
- Then, letting |Ei| denote the size of the index of search engine Ei, we have x|E1| ≈ y|E2|, from which we have the form we will use |E1|/|E2| ≈ y/x

Sampling Methods

- Random Searches
- Random IP addresses
- Random Walks
- Random Queries

- Actual Estimate is quite challanging

Duplicate/Near Duplicate Detection

- Web contains multiple copies of the same content
  - for redundancy and high availability

  hence while indexing for web search engine we may come up for duplicate
- Duplicates
  - checksum is used to detect duplicate
- Near Duplicate
  - not identical, but portion is common
  - Pre-set threshold is used to filter out near duplicates
    - shingling
- Shingling
  - detects near duplicates

- Shingling - Given a positive integer k and a sequence of terms in a document d, define the k-shingles of d to be the set of all consecutive sequences of k terms in d.

Shingling

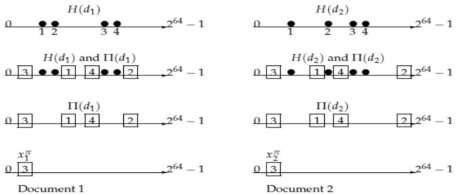
- To find a near duplicate, a shingling approach is used. If there are many common shingling for some k in a pair of documents, its contents will be the same.
- Consider a sentence below

  a rose is a rose is a rose.
- Its shingling set Z = {a rose is a ; rose is a rose ; is a rose is ; a rose is a ; rose is a rose } , which has |Z|=5
- Overlap, by Jaccard = 2/5

Near Dublicate scaled Approach

- A pair-wise approach seems unavoidable for using shingling overlap to detect near duplicate.
- We can perform better, by using a large integer Hash Function and doing Hashing for shingling patterns.

Near-Duplicate Scaled Approach



► **Figure 19.8** Illustration of shingle sketches. We see two documents going through four stages of shingle sketch computation. In the first step (top row), we apply a 64-bit hash to each shingle from each document to obtain $H(d_1)$ and $H(d_2)$ (circles). Next, we apply a random permutation Π to permute $H(d_1)$ and $H(d_2)$, obtaining $\Pi(d_1)$ and $\Pi(d_2)$ (squares). The third row shows only $\Pi(d_1)$ and $\Pi(d_2)$, while the bottom row shows the minimum values x_1^π and x_2^π for each document.

Web crawler → aka Spider

↳ a process by which we gather Pages from the web to index them and support a search engine

↳ It allows to quickly and efficiently gather as many useful web Pages as possible together with the link structure that interconnects them

Features a web crawler should provide

- Robustness
  - to deal with large no. of linked pages from website
  - Sometimes server traps a crawler
    the crawler must identify these traps
- Politeness
  - implicit and explicit webserver policies
    regulating the rate at which a crawler can visit them
- Distributed
  - be able to execute in a distributed fashion
    across multiple machines
- Scalable
  - crawler architecture should permit
    scaling up the crawl rate
    by adding extra machines and bandwidth
- Performance and Efficiency
  - make efficient use of various system resources including
    - Processor
    - storage
    - network bandwidth
- Quality
  - crawler should be biased towards
    fetching useful pages first
- Freshness
  - It should operate in continous mode
  - It should obtain fresh copies of previously fetched pages
- Extensible
  - demands crawler architech to be modular by
    - to cope with new data formats
    - to fetch new protocols

Web crawler 'MUST' provide these features

c. Outline at least three challenges of a modern web-crawler and suggest one solution to overcome each of them. [3]

The web-sites generally block access through automatic crawlers, the crawler must be polite in accessing these websites.

There can be several issues with physical access, network access and application layer of the web-host, a crawler must be robust to all these problem.

Crawler should be distributed and scalable as it need to access millions of pages per unit time.

- Quality: Given that a significant fraction of all web pages are of poor utility for serving user query needs, the crawler should be biased toward fetching "useful" pages first.

- Extensible: Crawlers should be designed to be extensible in many ways – to cope with new data formats, new fetch protocols, and so on. This demands that the crawler architecture be modular.

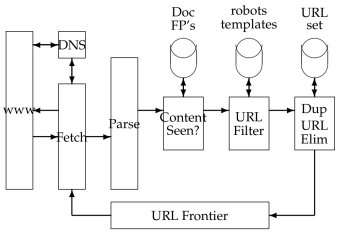
Architecture of a crawler

- URL Frontier
  - contains URLs yet to be fetched in the current crawl

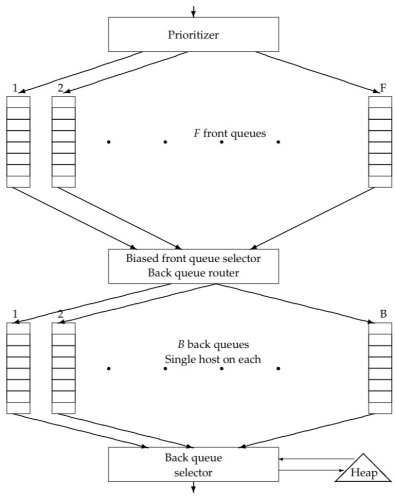
steps

- a seed set is stored in URL Frontier
- crawler begins by taking a URL from the seed set

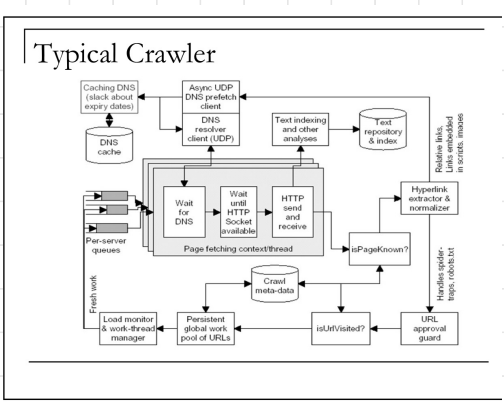
- DNS → Domain Name Service Resolution
  - looks up IP address for domain names
- Fetch
  - uses HTTP protocol to fetch the URL
- Parse
  - the page is parsed
  - text/images/videos/links are extracted
- Distributed indexes
  - by term → Global Indexes
  - by document → Local Indexes
- Connectivity Server
  - URL are transformed into integer values
  - In-Link and Out-Link states are maintained
  - ordering of URL based on
    - Host
    - Lexiographic ordering
    - etc.



► **Figure 20.1** The basic crawler architecture.



► **Figure 20.3** The URL frontier. URLs extracted from already crawled pages flow in at the top of the figure. A crawl thread requesting a URL extracts it from the bottom of the figure. En route, a URL flows through one of several *front queues* that manage its priority for crawling, followed by one of several *back queues* that manage the crawler's politeness.



LINK ANALYSIS

CHP 21

Link Analysis

↳ used by websearch engines
to compute a composite score for a webpage on any given query

Link Analysis

- The analysis of hyperlinks and the graph structure of the Web has been instrumental in the development of web search.
- In this chapter we focus on the use of hyperlinks for ranking web search results.

Web Page

<in Link> <out-link>

- Journal of the ACM.

↳ Anchor text

↳ Extended anchor text

Web Page as a Graph

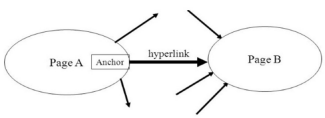
↳ web pages are connected with in and out links

A1: a hyperlink b/w pages
denotes a conferral of authority → quality signal

A2: the text in the anchor of the hyperlink describes the target page

The Web as a graph

- Web pages are connected with in- and out- links
- Our study of link analysis builds on two intuitions:
  - The anchor text pointing to page B is a good description of page B.
  - The hyperlink from A to B represents an endorsement of page B, by the creator of page A.



Page Rank

↳ technique for link analysis

↳ assigns every node in the web graph
a numerical score b/w 0 and 1 → Page Rank

↳ The Page rank of a node depends on the link structure of the web graph

- Given a query, a web search engine computes a composite score for each web page that combines hundreds of features such as cosine similarity and term proximity, together with the PageRank score.

Page Rank

- Consider a random surfer who begins at a web page (a node of the web graph) and executes a random walk on the Web as follows.
  - At each time step, the surfer proceeds from his current page A to a randomly chosen web page that A hyperlinks to.
  - As the surfer proceeds in this random walk from node to node, he visits some nodes more often than others; intuitively, these are nodes with many links coming in from other frequently visited nodes.
  - In the teleport operation the surfer jumps from a node to any other node in the web graph. This could happen because he types an address into the URL bar of his browser.
  - Teleporting is uniformly performed.

PAGE RANK

PROS

1. the algo is robust against spam as its difficult for web owner to add link to their page from other imp pages
2. It is a global measure and is query independent

CONS

1. It favors older pages b/c a new page will not have many links unless it is part of an existing site
2. It can easily be increased by 'link farms' However, while indexing, the search actively tries to find these flaws

Teleporting

↳ the surfer jumps from a node to any other node in the web graph

↳ Its used to assign page rank score to each node

teleport operation can be done in 2 ways

1. when at a node with no out-links the surfer invokes the teleportation operation
2. At any node that has outgoing links the sufer invokes the teleportation operation
   ↳ with probability $0 < \alpha < 1$
   ↳ and standard random walk

teleport operation with probability $0 < \alpha < 1$ and the standard random walk (follow an out-link chosen uniformly at random with probability $1 - \alpha$, where α is a fixed parameter chosen in advance. Typically, α might be 0.1.

markov chain

↳ a discrete time stochastic process

↳ a process that occurs in a series of time steps in each of which a random choice is made

↳ It consists on N states

↳ each web page corresponds to a state in markov chain

↳ It is characterized by N×N transition probability matrix P each of whose entries is in the interval [0, 1] the entries in each row of P adds up to 1

Markov Chain

- The Markov chain can be in one of the N states at any given timestep; then, the entry Pij tells us the probability that the state at the next timestep is j, conditioned on the current state being i.
- Each entry Pij is known as a transition probability and depends only on the current state i; this is known as the Markov property.

MARKOV CHAIN AS

Stochastic matrix

↳ a matrix with non ve enteries that satisfies

↳ It has a principal left eigen vector corresponding to its largest eigen value, which is 1

- An N-dimensional probability vector each of whose components corresponds to one of the N states of a Markov chain can be viewed as a probability distribution over its states
- Markov Properties

$$\forall i, j, P_{ij} \in [0, 1] \qquad \forall i, \sum_{j=1}^{N} P_{ij} = 1.$$

Markov Chain Probability Matrix

- How to get the probability matrix?
  - If a row of A has no 1's, then replace each element by 1/N.
  - For all other rows proceed as follows:
  - Divide each 1 in A by the number of 1's in its row. Thus, if there is a row with three m's, then each of them is replaced by 1/m.
  - Multiply the resulting matrix by 1 − α.
  - Add α/N to every entry of the resulting matrix, to obtain the required matrix P.

Ergodic Markov chain

- A Markov chain is said to be ergodic if there exists a positive integer T0 such that for all pairs of states i, j in the Markov chain, if it is started at time 0 in state i then for all t > T0, the probability of being in state j at time t is greater than 0.
- The random walk with teleporting results in a unique distribution of steady-state probabilities over the states of the induced Markov chain.
- This steady-state probability for a state is the PageRank of the corresponding web page.

Steps

↳ make graph

↳ make adjency matrix

↳ each row sum = 1

↳ A = A x (1−α) / A = A x d

↳ P = A + (α/N)

↳ Page Rank Algo

↳ $x_1 = x_0 * P$

↳ $x_2 = x_1 * P$

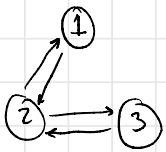
⋮

Exercise 21.6

Consider a web graph with three nodes 1, 2 and 3. The links are as follows: $1 \to 2, 3 \to 2, 2 \to 1, 2 \to 3$. Write down the transition probability matrices for the surfer's walk with teleporting, for the following three values of the teleport probability: (a) $\alpha = 0$; (b) $\alpha = 0.5$ and (c) $\alpha = 1$.

$$A: \begin{array}{c} \\ 1 \\ 2 \\ 3 \end{array} \begin{array}{c} \begin{array}{ccc} 1 & 2 & 3 \end{array} \\ \begin{bmatrix} 0 & 1 & 0 \\ 0.5 & 0 & 0.5 \\ 0 & 1 & 0 \end{bmatrix} \end{array}$$

∴ row sum = 1



a) $\alpha = 0$

↳ multiply $1-\alpha$ to every entry

$A \times (1-0)$

$$= \begin{bmatrix} 0 & 1 & 0 \\ 0.5 & 0 & 0.5 \\ 0 & 1 & 0 \end{bmatrix}$$

↳ add α/N to every entry

$A + 0/3$

$$P = \begin{bmatrix} 0 & 1 & 0 \\ 0.5 & 0 & 0.5 \\ 0 & 1 & 0 \end{bmatrix}$$

Probability Matrix

b) $\alpha = 0.5$

↳ $A \times 1 - 0.5$

$A \times 0.5$

$$\begin{bmatrix} 0 & 0.5 & 0 \\ 0.25 & 0 & 0.25 \\ 0 & 0.5 & 0 \end{bmatrix}$$

↳ $A + {}^{0.5}/_{3}$

$A + 1/6$

$$P = \begin{bmatrix} 1/6 & 2/3 & 1/6 \\ 5/12 & 1/6 & 5/12 \\ 1/6 & 2/3 & 1/6 \end{bmatrix}$$

c) $\alpha = 1$

↳ $A \times 1 - 1$

↳ $A \times 0$

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

↳ $A + 1/3$

$$P: \begin{bmatrix} 1/3 & 1/3 & 1/3 \\ 1/3 & 1/3 & 1/3 \\ 1/3 & 1/3 & 1/3 \end{bmatrix}$$

Q) Imagine that the surfer starts in state 1, corresponding to the initial probability distribution vector $\vec{x}_0 = (1\ 0\ 0)$. Then, after one step the distribution is

$$\vec{x}_0 P = \begin{pmatrix} 1/6 & 2/3 & 1/6 \end{pmatrix} = \vec{x}_1.$$

Page rank algo

$X_0 = (1\ 0\ 0)$

$X_1 = X_0 * P$ → matrix multiplication

$X_2 = X_1 * P$

⋮

The PageRank computation

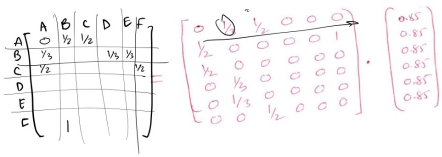
$\vec{x}_0$	1	0	0
$\vec{x}_1$	1/6	2/3	1/6
$\vec{x}_2$	1/3	1/3	1/3
$\vec{x}_3$	1/4	1/2	1/4
$\vec{x}_4$	7/24	5/12	7/24
...	...	...	...
$\vec{x}$	5/18	4/9	5/18

After two steps it is

$$\vec{x}_1 P = \begin{pmatrix} 1/6 & 2/3 & 1/6 \end{pmatrix} \begin{pmatrix} 1/6 & 2/3 & 1/6 \\ 5/12 & 1/6 & 5/12 \\ 1/6 & 2/3 & 1/6 \end{pmatrix} = \begin{pmatrix} 1/3 & 1/3 & 1/3 \end{pmatrix} = \vec{x}_2.$$

iii. Assume that the PageRank values for any page p_i at iteration 0 is $PR(p_i) = 1$ and that the damping factor for iterations is d = 0.85 Perform the PageRank algorithm and determine the rank for every page after 2 iterations. [3]

→ daping factor

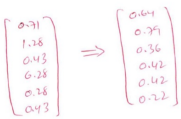


METHOD 1

$I_1 = A^T \times d =$

$I_2 = A^T \times I_1$

$I_3 = A^T \times I_2$

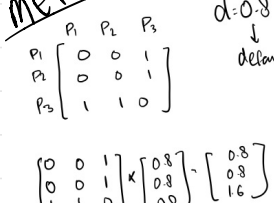


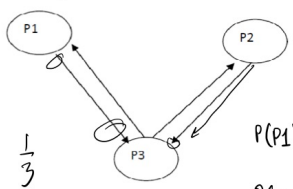
Question No. 7 [Time:30 min] [Points: 10+10]

a. Consider the following graph, which represents different web-pages that are linked together. You are required to calculate the page rank of each page using PageRank algorithm. You can use any method to solve the combination of equations. [5]

METHOD 2

METHOD 1





Let d=0.8 , we will have the general equation for each page as below

PR(P1) = 0.2 + 0.8 (PR(P3)/2)
PR(P2) = 0.2 + 0.8 (PR(P3) / 2)
PR(P3) = 0.2 + 0.8 (PR(P1) + PR(P2))

These equations can easily be solved. We get the following PageRank values for the given pages:

PR(P1)= 0.776
PR(P2) = 0.776
PR(P3) = 1.448

b. Outline the key differences between HITS and PageRank algorithms. [5]

HITS	PageRank

HITS → Hyper Link Induced Topic Search → aka Hubs and Authorities

↳ a link analysis algo

↳ It rates web pages

↳ Hubs

↳ large dictionaries → that were not authoritative in info

↳ used as compilations of a broad catalogue of info that led users directly to other authoritative pages

↳ the algo assigns 2 scores for each page

↳ its authority value

↳ which estimates the value of the content of the page

↳ its hub value

↳ which estimates the value of its links to other pages

A good hub represents

↳ a page that pointed to many other pages

A good authority represents

↳ a page that was linked by many diff hubs

HITS CONS

↳ It is query dependent

↳ the 2 scores are influenced by search terms

↳ It is executed at query time

↳ not commonly used by search engines

↳ It computes 2 scores per doc as opposed to a single score

↳ It is processed on a small set of relavent docs not all docs, which was the case in Page Rank

Example

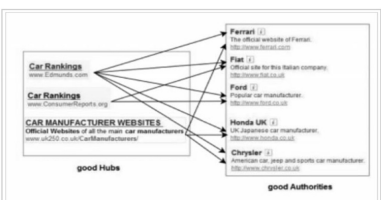


Figure 1: An example of authorities and hubs

- **It is query dependent, that is, the (Hubs and Authority) scores resulting from the link analysis are influenced by the search terms;**
- **As a corollary, it is executed at query time, not at indexing time, with the associated hit on performance that accompanies query-time processing.**

HITS Algorithm

The HITS Algorithm can be described as follows:

1) Given a search query Q, collect the top 200 webpages that contain the highest frequency of query Q.

2) Add the the collection the webpages that point to or are pointed by these top 200 webpages. Create Adjacency Matrix A among these webpages.

3) Initialize the hub and authority column vectors U and V with values of 1.

4) For a set k number of iterations, do the following:

```
a) Update the authority scores through the authority matrix V
b) Update the hub scores through the hub matrix U
c) Normalize the hub matrix and authority matrix U and V
```

5) Rank the webpages according to the authority score as reflected through authority matrix V

HITS Algorithm

The algorithm performs a series of iterations, each consisting of two basic steps:

- **Authority Update**: Update each node's *Authority score* to be equal to the sum of the *Hub Scores* of each node that points to it. That is, a node is given a high authority score by being linked to pages that are recognized as Hubs for information.
- **Hub Update**: Update each node's *Hub Score* to be equal to the sum of the *Authority Scores* of each node that it points to. That is, a node is given a high hub score by linking to nodes that are considered to be authorities on the subject.

HITS Algorithm

$\forall p$. we update $\mathbf{auth}(p)$ to be the summation:

$$\mathbf{auth}(p) = \sum_{i=1}^{n} \mathbf{hub}(i)$$

where n is the total number of pages connected to p and i is a page connected to p. That is, the Authority score of a page is the sum of all the Hub scores of pages that point to it.

~~$\forall p$. we update $\mathbf{hub}(p)$ to be the summation:~~

$$\mathbf{hub}(p) = \sum_{i=1}^{n} \mathbf{auth}(i)$$

where n is the total number of pages p connects to and i is a page which p connects to. Thus a page's Hub score is the sum of the Authority scores of all its linking pages

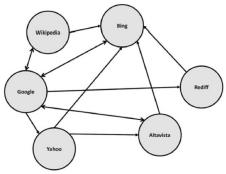
HITS Algorithm

$$a_j^{(k)} \leftarrow \sum_{(i,j)\in E} h_i^{(k-1)}, \qquad h_i^{(k)} \leftarrow \sum_{(i,j)\in E} a_j^{(k)},$$

$$\mathbf{a}^{(k)} \leftarrow \mathbf{A}^{\mathbf{T}} \cdot \mathbf{h}^{(k-1)}. \qquad \mathbf{h}^{(k)} \leftarrow \mathbf{A} \cdot \mathbf{a}^{(k)}.$$

Example:

- A subset of graph with selected Hub & Authority status.



- This is a result of resultant search result on "q"

Adjacency Matrix

	Wiki	Google	Bing	Yahoo	Altavista	Rediff
Wikipedia	0	1	1	0	0	0
Google	1	0	1	1	1	1
Bing	0	1	0	0	0	0
Yahoo	0	0	1	0	1	0
Altavista	0	1	1	0	0	0
Rediffmail	0	0	1	0	0	0

Iterative calculation of Hub & Authority

$$\mathbf{a}^{(1)} = \mathbf{A}^{\mathbf{T}} \cdot \mathbf{h}^{(0)}$$

A^T — old hub

$$= \begin{bmatrix} 0&1&1&0&0&0 \\ 1&0&1&1&1&1 \\ 0&1&0&0&0&0 \\ 0&0&1&0&1&0 \\ 0&1&1&0&0&0 \\ 0&0&1&0&0&0 \end{bmatrix}^{\mathbf{T}} \cdot \begin{bmatrix} 1\\1\\1\\1\\1\\1 \end{bmatrix}$$

$$= \begin{bmatrix} 0&1&0&0&0&0 \\ 1&0&1&0&1&0 \\ 1&1&0&1&1&1 \\ 0&1&0&0&0&0 \\ 0&1&0&1&0&0 \\ 0&1&0&0&0&0 \end{bmatrix} \cdot \begin{bmatrix} 1\\1\\1\\1\\1\\1 \end{bmatrix}$$

$$= \begin{bmatrix} 1\\3\\5\\1\\2\\1 \end{bmatrix}$$ new authority

$\sqrt{a^2+b^2+c^2+d^2+\cdots}$

Normalized

$$\mathbf{a}^{(1)} = \begin{bmatrix} \frac{1}{\sqrt{1^2+3^2+5^2+1^2+2^2+1^2}} \\ \frac{3}{\sqrt{1^2+3^2+5^2+1^2+2^2+1^2}} \\ \frac{5}{\sqrt{1^2+3^2+5^2+1^2+2^2+1^2}} \\ \frac{1}{\sqrt{1^2+3^2+5^2+1^2+2^2+1^2}} \\ \frac{2}{\sqrt{1^2+3^2+5^2+1^2+2^2+1^2}} \\ \frac{1}{\sqrt{1^2+3^2+5^2+1^2+2^2+1^2}} \end{bmatrix}$$

$$= \begin{bmatrix} \frac{1}{\sqrt{41}} \\ \frac{3}{\sqrt{41}} \\ \frac{5}{\sqrt{41}} \\ \frac{1}{\sqrt{41}} \\ \frac{2}{\sqrt{41}} \\ \frac{1}{\sqrt{41}} \end{bmatrix}$$

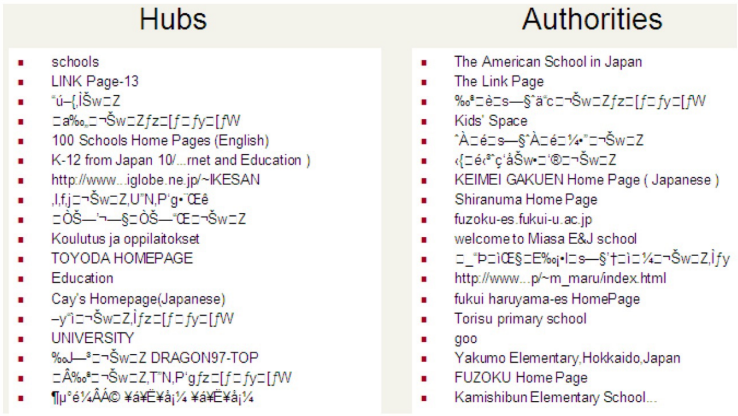
$$= \begin{bmatrix} 0.15617 \\ 0.46852 \\ 0.78087 \\ 0.15617 \\ 0.312348 \\ 0.15617 \end{bmatrix}$$ → new authority

first h and a are all 1s

new Authority: $A^T \times h^i$

new Hub: $A \times a^0$

Initially h^0 and a^0: $\begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$



► **Figure 21.6** A sample run of HITS on the query japan elementary schools.

HITS vs. PageRank

HITS	PageRank
It gives 2 scores Hub and Authority for each page.	It gives one score e.g. PageRank.
It is executed at query time	It is executed at indexing time.
Not robust against spams.	Robust against web-spams.
Never favor pages, but can be manipulated.	Favor old pages.
It is query dependent	It is query independent

real estate would rank highly. This led to the first generation of *spam*, which (in the context of web search) is the manipulation of web page content for the purpose of appearing high up in search results for selected keywords.

ishma hafeez
notes
repsht