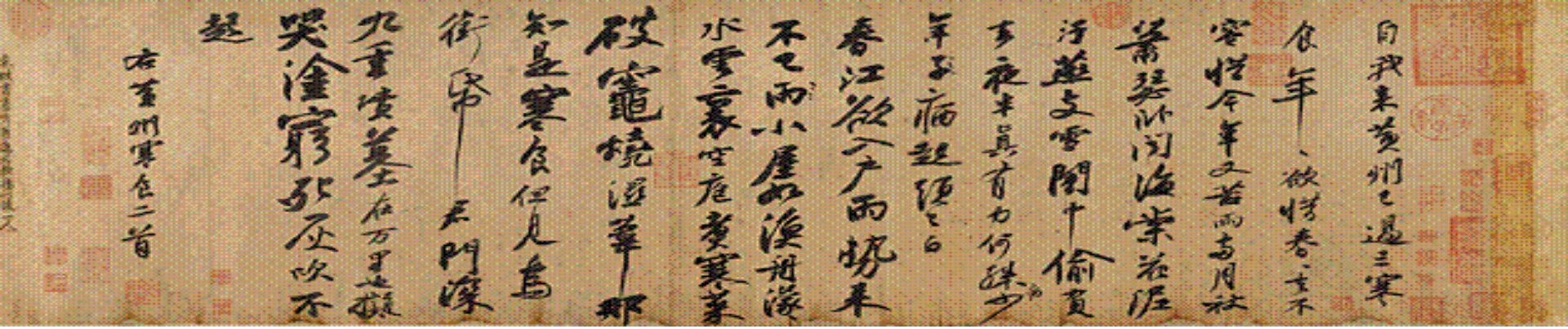


信息检索
Information Retrieval

教师：孙茂松

Tel:62781286

Email:sms@tsinghua.edu.cn

TA：**胡锦毅**

Email:hu-jy21@mails.tsinghua.edu.cn

郑重声明

- 此课件仅供选修清华大学计算机系本科生课《信息检索》(40240372)的学生个人学习使用，所以只允许学生将其下载、存贮在自己的电脑中未经孙茂松本人同意，任何人不得以任何方式扩散之（包括放到9#服务器上）。否则，由此可能引起的一切涉及知识产权的法律责任，概由该人负责。
- 此课件仅限孙茂松本人讲课使用。除孙茂松本人外，凡授课过程中，PTT文件显示此《郑重声明》之情形，即为侵权使用。

第三章 文本分析及自动标引 (Part 1)

3.0 Ranked retrieval

- Thus far, our queries have all been Boolean.
  - Documents either match or don't.
- Good for expert users with precise understanding of their needs and the collection.
  - Also good for some applications: Applications can easily consume 1000s of results.
- Not good for the majority of users.
  - Most users incapable of writing Boolean queries (or they are, but they think it's too much work).
  - Most users don't want to wade through 1000s of results.
    - This is particularly true of web search.

Problem with Boolean search: feast or famine

- Boolean queries often result in either too few (=0) or too many (1000s) results.
- Query 1: "*standard user*" → 200,000 hits
- Query 2: "*standard user dlink*": 0 hits
- It takes a lot of skill to come up with a query that produces a manageable number of hits.
  - AND gives too few; OR gives too many

Ranked retrieval models

- Rather than a set of documents satisfying a query expression, in ranked retrieval models, the system returns an ordering over the (top) documents in the collection with respect to a query
- Free text queries: Rather than a query language of operators and expressions, the user's query is just one or more words in a human language
- In principle, there are two separate choices here, but in practice, ranked retrieval models have normally been associated with free text queries and vice versa

Scoring as the basis of ranked retrieval

- We wish to return in order the documents most likely to be useful to the searcher
- How can we rank-order the documents in the collection with respect to a query?
- Assign a score to each document
- This score measures how well document and query "match".
- Assign a score to each term in each document

3.1 标引

- Indexing: The most crucial and probably the most difficult one consists in assigning appropriate terms or content identifiers capable of representing the content of the collection items.
- Manual(trained experts) vs. automatic indexing
- Controlled vs. uncontrolled indexing terms

3.1 标引

- Exhaustivity and specificity
- Evaluation: recall vs precision

Recall: the proportion of the relevant information actually retrieved in response to a search

Precision: the proportion of retrieved items actually relevant

$$recall = \frac{the-number-of-relevant-items-actually-obtained}{the-total-number-of-relevant-items-contained-in-the-coll}$$

$$precision = \frac{the-number-of-relevant-items-actually-obtained}{the-total-number-of-retrieved-items}$$

3.1 标引

- Substantial evidence indicates that simple automatic indexing methods are fast and inexpensive, and produce a recall and precision performance at least equivalent to that obtainable in manual, controlled term environments.

3.2 Term的自动抽取及其加权

- Indexing task:
  (1) terms(or concepts): content identifiers
  (2) weight of terms

- frequency of word types (distinct words)
- Zipf's law: the Harvard linguistic

  George Kingsley Zipf (1902-1950)

  frequency * rank ≈ constant

3.2 Term的自动抽取及其加权

Zipf's law: If the terms in a collection are ranked (r) by their frequency (f_r), they roughly fit the relation $r * f\_r \approx C$. Different collections have different constants C, but in English text, C tends to be about $N / 10$, where N is the number of words in the collection.

$p\_r = f\_r / N$ is the probability that a randomly chosen term (with frequency f_r) will have rank r.

$r * p\_r = A$, where A tends to be about 0.1 in English text.

3.2 Term的自动抽取及其加权

Statistics from the TIME collection, a 1.6 MB collection of 423 short TIME magazine articles (245,412 term occurrences). Top 50 terms:

Word	f_t	$r_t * p_r$	Word	f_t	$r_t * p_r$	Word	f_t	$r_t * p_r$
the	15861	0.065	it	1290	0.095	week	793	0.113
of	7239	0.059	from	1228	0.095	they	697	0.102
to	6331	0.077	but	1138	0.093	govern	687	0.104
a	5878	0.096	u	955	0.082	all	672	0.104
and	5614	0.114	had	940	0.084	year	672	0.107
in	5294	0.129	last	930	0.087	its	620	0.101
that	2507	0.072	be	915	0.089	britain	89	0.098
for	2228	0.073	have	914	0.093	when	579	0.099
was	2149	0.079	who	894	0.095	out	577	0.101
with	1839	0.075	not	882	0.097	would	577	0.103
his	1815	0.081	has	880	0.100	new	572	0.105
is	1810	0.089	an	873	0.103	up	559	0.105
he	1700	0.090	s	865	0.106	been	554	0.106
as	1581	0.090	were	848	0.107	more	540	0.106
on	1551	0.095	their	815	0.106	which	539	0.108
by	1467	0.096	are	812	0.109	into	518	0.106
at	1333	0.092	one	811	0.112			

3.2 Term的自动抽取及其加权

Statistics from the WSJ87 collection, a 131.6 MB collection of 46,449 newspaper articles (19 million term occurrences). Top 50 terms:

Word	f_t	$r_t * p_r$	Word	f_t	$r_t * p_r$	Word	f_t	$r_t * p_r$
the	1130021	0.059	from	96900	0.092	or	54958	0.101
of	547311	0.058	he	94585	0.095	about	53713	0.102
to	516635	0.082	million	3515	0.098	market	52110	0.101
a	464736	0.098	year	90104	0.100	they	51359	0.103
in	390819	0.103	its	86774	0.100	this	50933	0.105
and	387703	0.122	be	85588	0.104	would	50828	0.107
that	204351	0.075	was	83398	0.105	u	49281	0.106
for	199340	0.084	company	3070	0.109	which	48273	0.107
is	152483	0.072	an	76974	0.105	bank	47940	0.109
said	148302	0.078	has	74405	0.106	stock	47401	0.110
it	134323	0.078	are	74097	0.109	trade	47310	0.112
on	121173	0.077	have	73132	0.112	his	47116	0.114
by	118863	0.081	but	71887	0.114	more	46244	0.114
as	109135	0.080	will	71494	0.117	who	42142	0.106
at	101779	0.080	say	66807	0.113	one	41635	0.107
mr	101679	0.086	new	64456	0.112	their	40910	0.108
with	101210	0.091	share	63925	0.114			

3.2 Term的自动抽取及其加权

Does Real Data Fit Zipf's Law?

A law of the form $y = kx^c$ is called a power law.

Zipf's law is a power law with $c = -1$

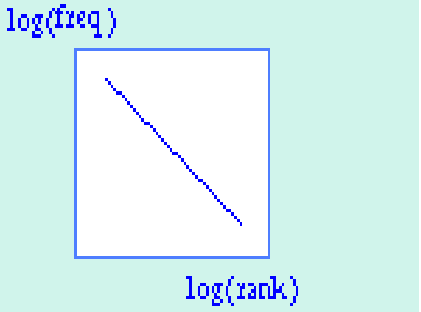
On a log-log plot, power laws give a straight line with slope c.

$$\log(y) = \log(kx^c) = \log k + c\log(x)$$

Zipf is quite accurate except for very high and low rank.

3.2 Term的自动抽取及其加权

"Principle of least effort"



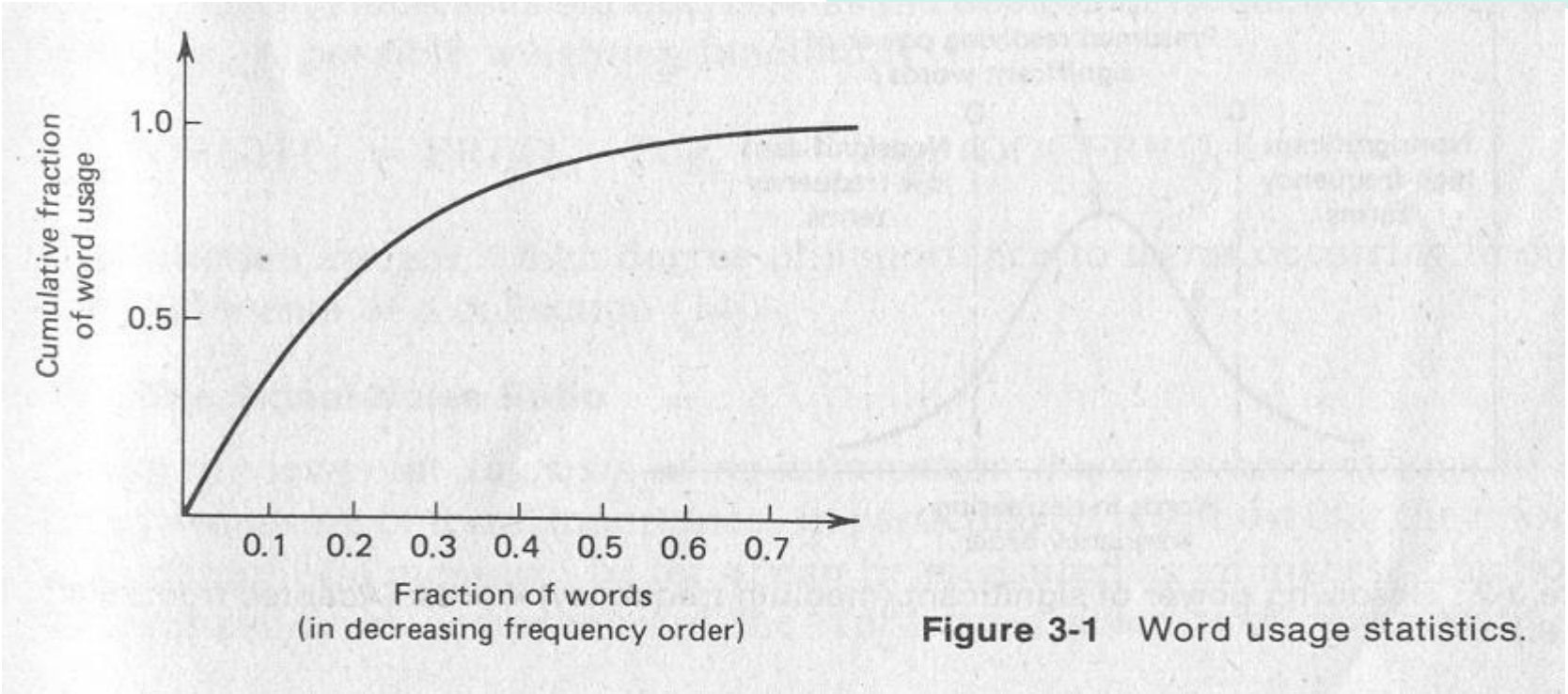
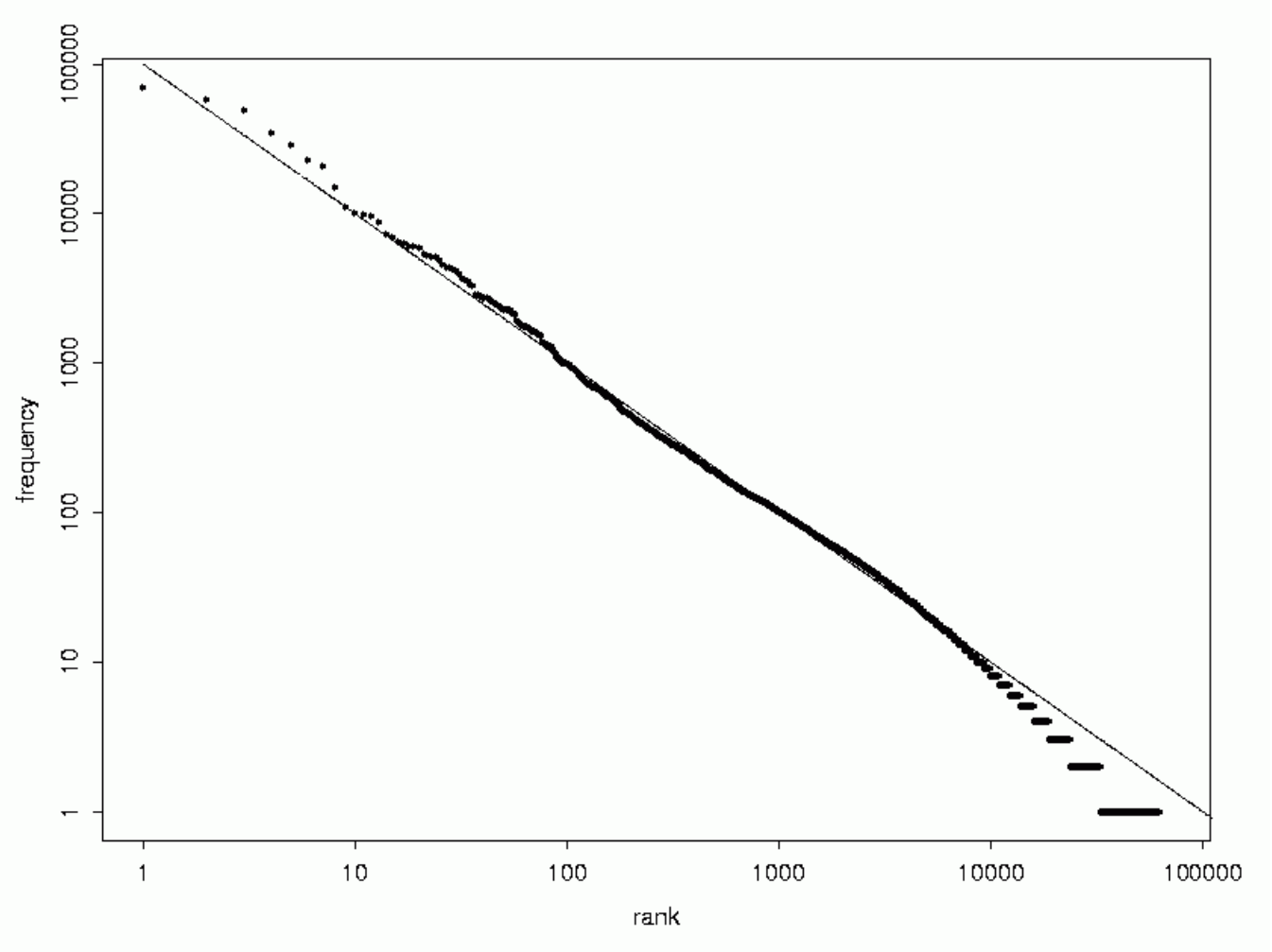


Figure 3-1 Word usage statistics.

3.2 Term的自动抽取及其加权

Zipf's law log-log plot



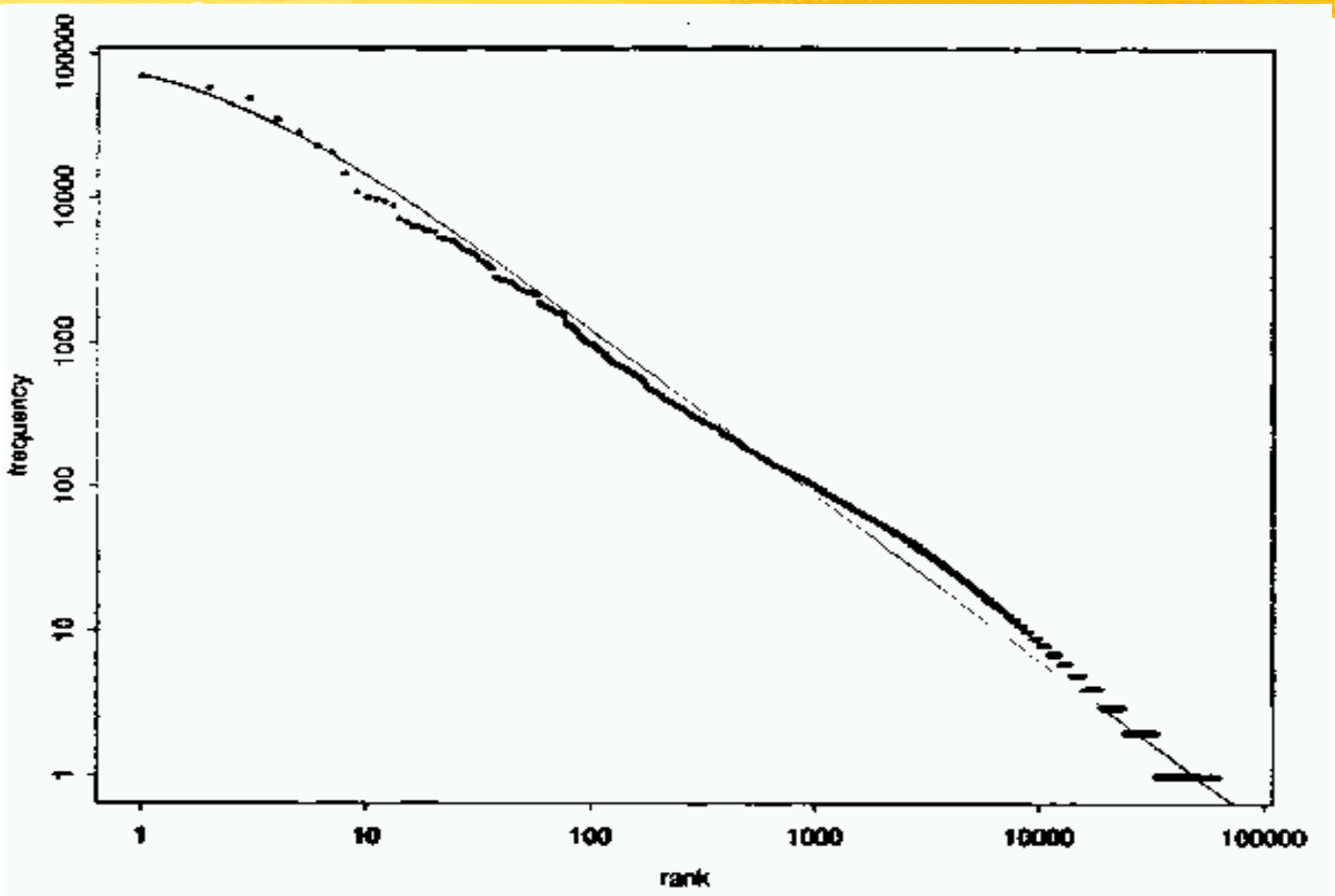
3.2 Term的自动抽取及其加权

Mandelbrot (1954) Correction

The following more general form gives a bit better fit:

$$f = P(r + \rho)^{-B} \qquad \text{For constants } P, B, \rho$$

3.2 Term的自动抽取及其加权



Mandelbrot's function on Brown corpus

$P = 10^{5.4}$, $B = 1.15$, $\rho = 100$

3.2 Term的自动抽取及其加权

Zipf's Law Impact on IR

Good News: Stopwords will account for a large fraction of text so eliminating them greatly reduces inverted-index storage costs.

Bad News: For most words, gathering sufficient data for meaningful statistical analysis (e.g. for correlation analysis for query expansion) is difficult since they are extremely rare.

3.2 Term的自动抽取及其加权

Predicting Word Frequency Distribution

By Zipf's Law, a word appearing *n* times has rank $r_n = AN/n$

Several words may occur *n* times, assume rank r_n applies to the last of these.

Therefore, r_n words occur *n* or more times and r_{n+1} words occur $n+1$ or more times.

So, the number of words appearing **exactly** *n* times is:

$$I_n = r_n - r_{n+1} = \frac{AN}{n} - \frac{AN}{n+1} = \frac{AN}{n(n+1)}$$

3.2 Term的自动抽取及其加权

Predicting Word Frequency Distribution

Assume highest ranking term occurs once and therefore has rank $D = AN/1$

Fraction of words with frequency n is:

$$\frac{I_n}{D} = \frac{1}{n(n+1)}$$

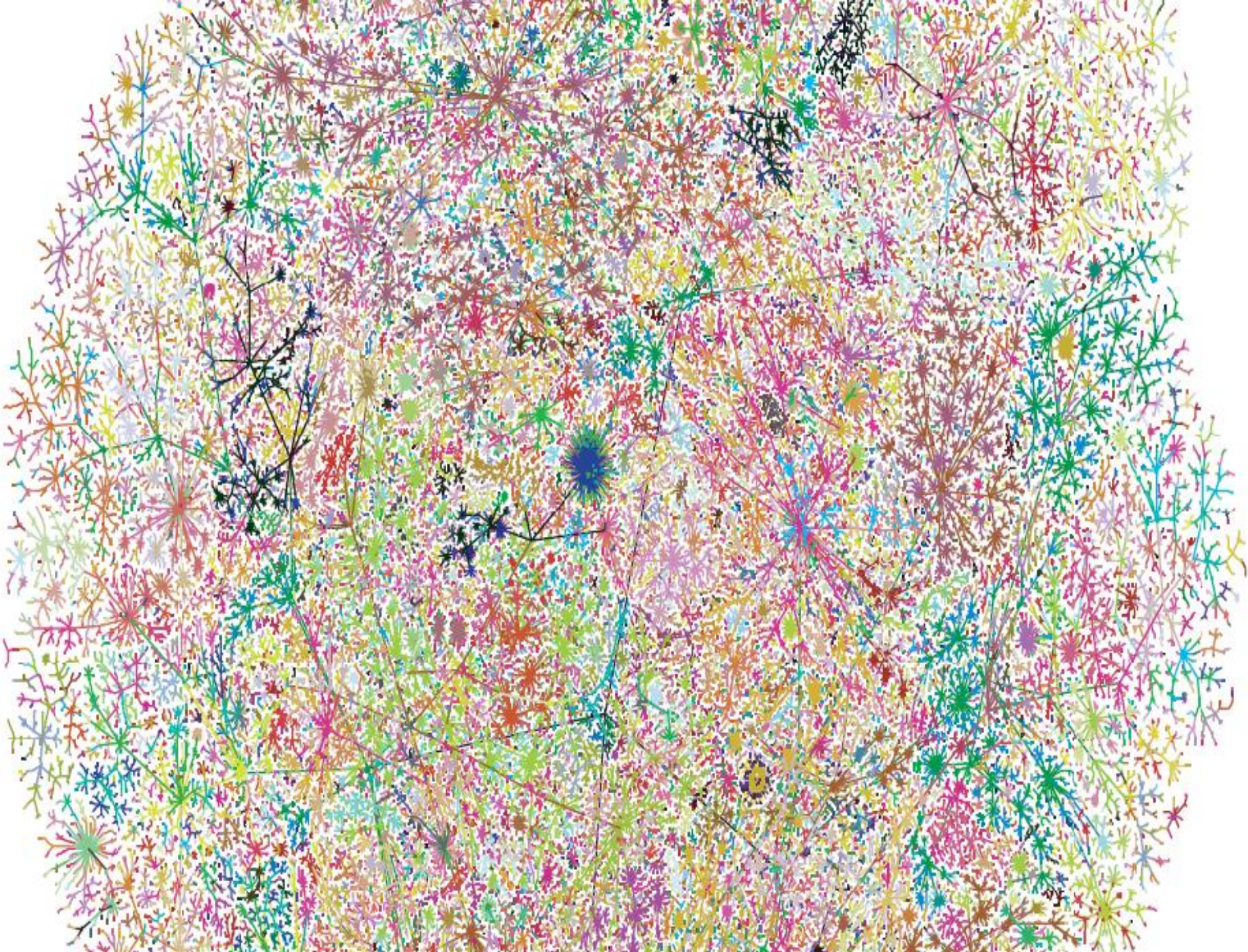
Fraction of words appearing only once is therefore ½.

3.2 Term的自动抽取及其加权

Word Frequency Data (from B. Croft, UMass)

Number of Occurrences (n)	Predicted Proportion of Occurrences 1/n(n+1)	Actual Proportion occurring n times I_n/D	Actual Number of Words occurring n times
1	.500	.402	204,357
2	.167	.132	67,082
3	.083	.069	35,083
4	.050	.046	23,271
5	.033	.032	16,332
6	.024	.024	12,421
7	.018	.019	9,766
8	.014	.016	8,200
9	.011	.014	6,907
10	.009	.012	5,893

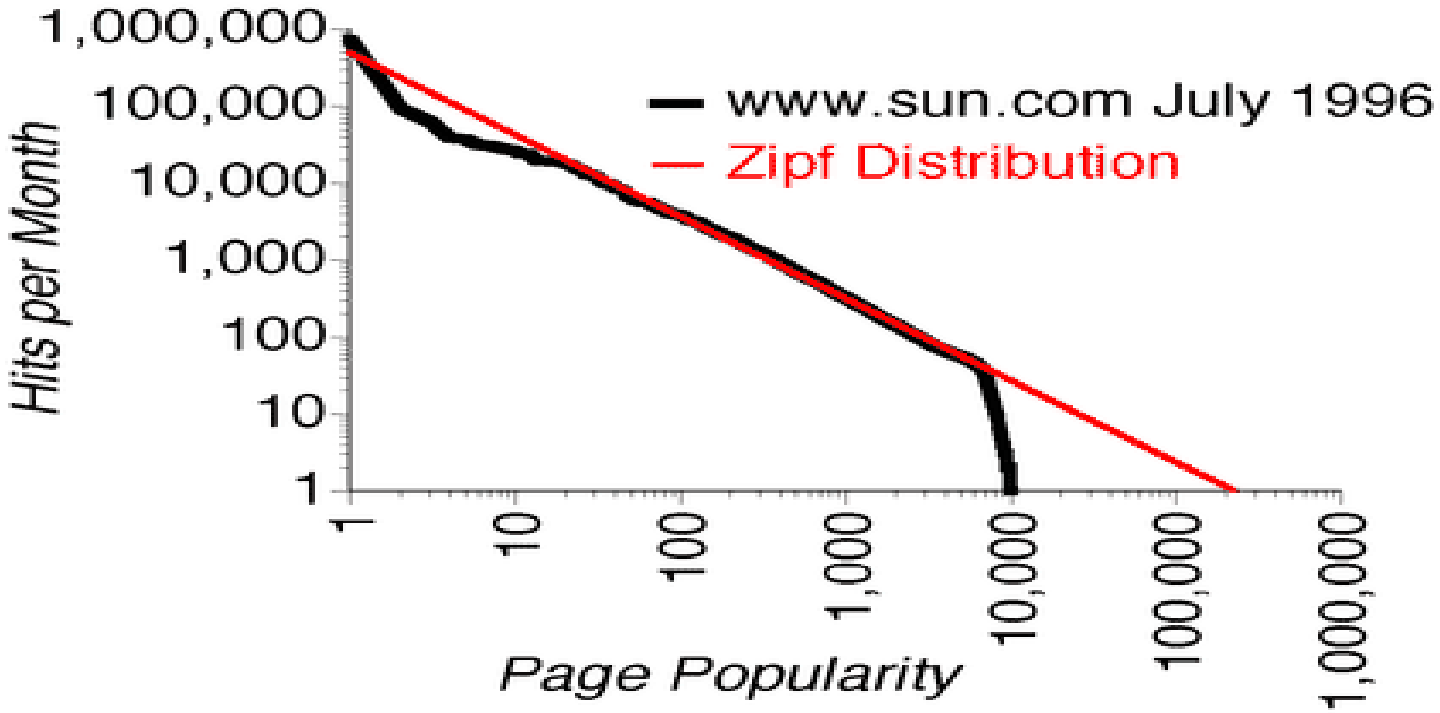
Frequencies from 336,310 documents in the 1GB TREC Volume 3 Corpus
125,720,891 total word occurrences; 508,209 unique words



3.2 Term的自动抽取及其加权

http://www.useit.com/alertbox/zipf.html

Zipf Curves and Website Popularity



Comparing empirical log data from Sun's website with a theoretical Zipf distribution. .

3.2 Term的自动抽取及其加权

Zipf's Law by **Dr. Richard S. Wallace**
http://www.alicebot.org/articles/wallace/zipf.html
关于Zipf‘s law的一篇介绍性短文。
选读

http://linkage.rockefeller.edu/wli/zipf/index_ru.html
Zipf’s Law的基本描述及其在各学科领域中的应用
选读

3.2 Term的自动抽取及其加权

- Indexing based on word frequency

(1) frequency of word k in document i: TF_{ik}

(2) the total collection frequency of word k: TTF_k

$$TTF_k = \sum_{i=1}^{n} TF_{ik}$$

(3) high threshold: remove all words with a collection frequency above it

(4) low threshold: remove all words with a collection frequency below it

(5) the remaining medium-frequency words as index terms

3.2 Term的自动抽取及其加权

Luhn (1958) suggested that both extremely common and extremely uncommon words were not very useful for indexing.

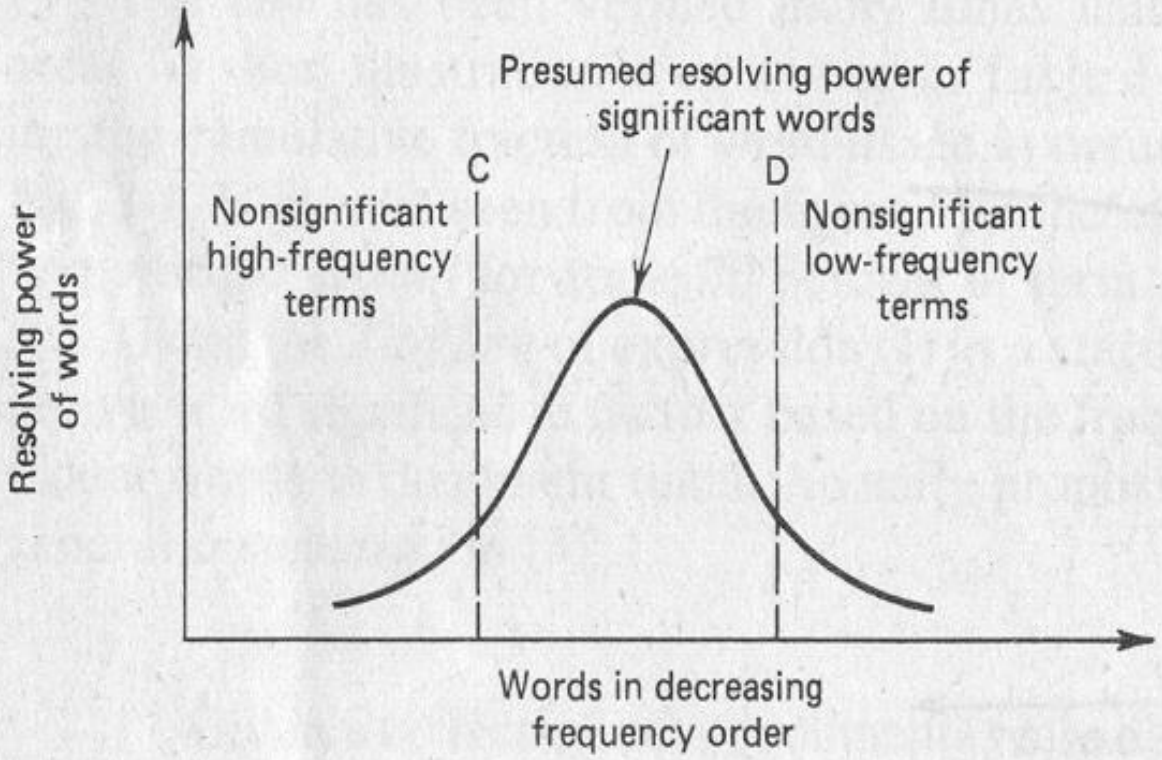


Figure 3-2 Resolving power of significant (medium-frequency) words. (*Adapted from reference 8.*)

3.2 Term的自动抽取及其加权

Zipf's Law and Term Weighting

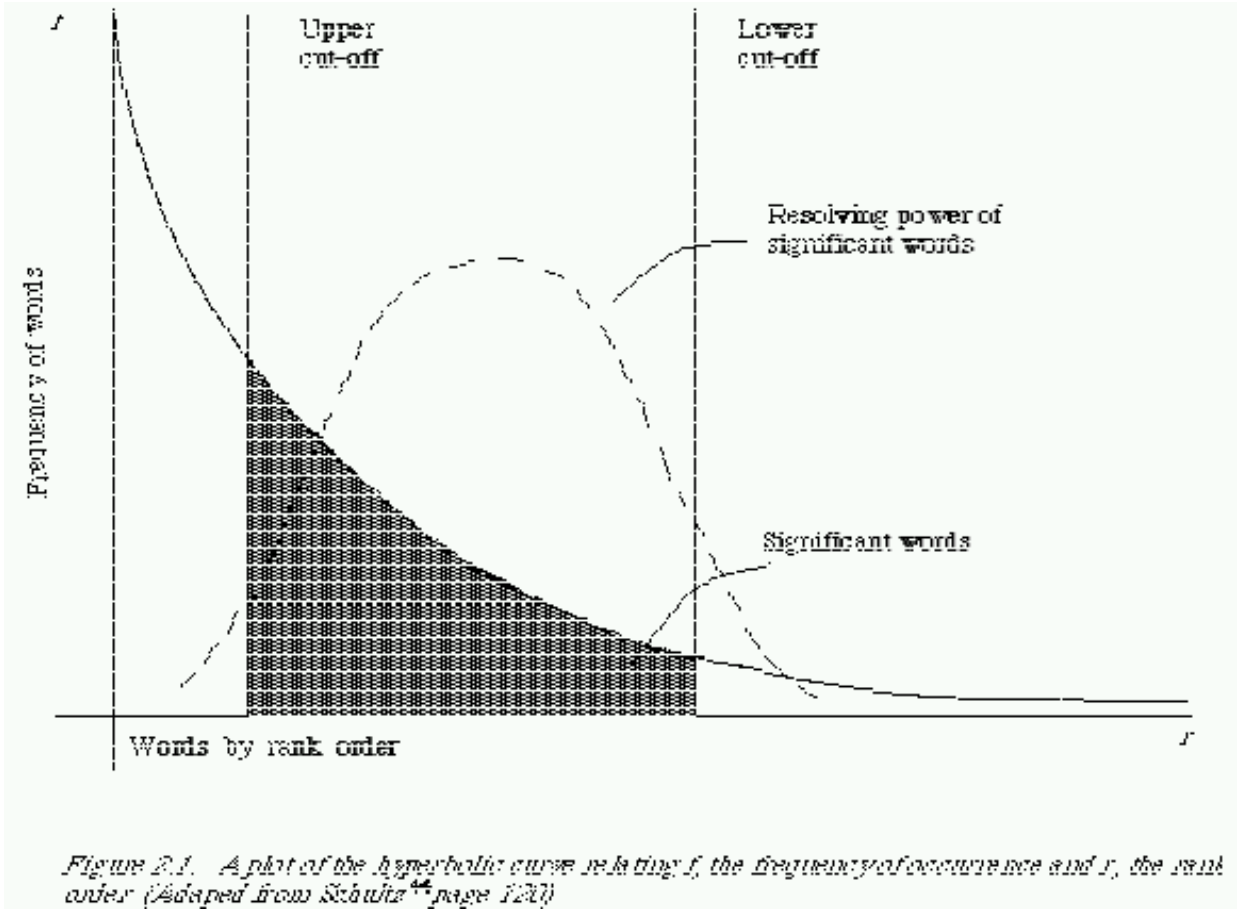


Figure 2.1. A plot of the hyperbolic curve relating f, the frequency of occurrence and r, the rank order (Adapted from Schultz[44] page 120)

Elimination of all high-frequency words might produce losses in recall; Elimination of all low-frequency words might produce losses in precision (and recall).

3.2 Term的自动抽取及其加权

Information content (self information) of a term:

$$\text{INFORMATION} = -\log_2 p$$

$$INFORMATION = -\log_2(0.0001) = -(-13.278) = 13.278$$

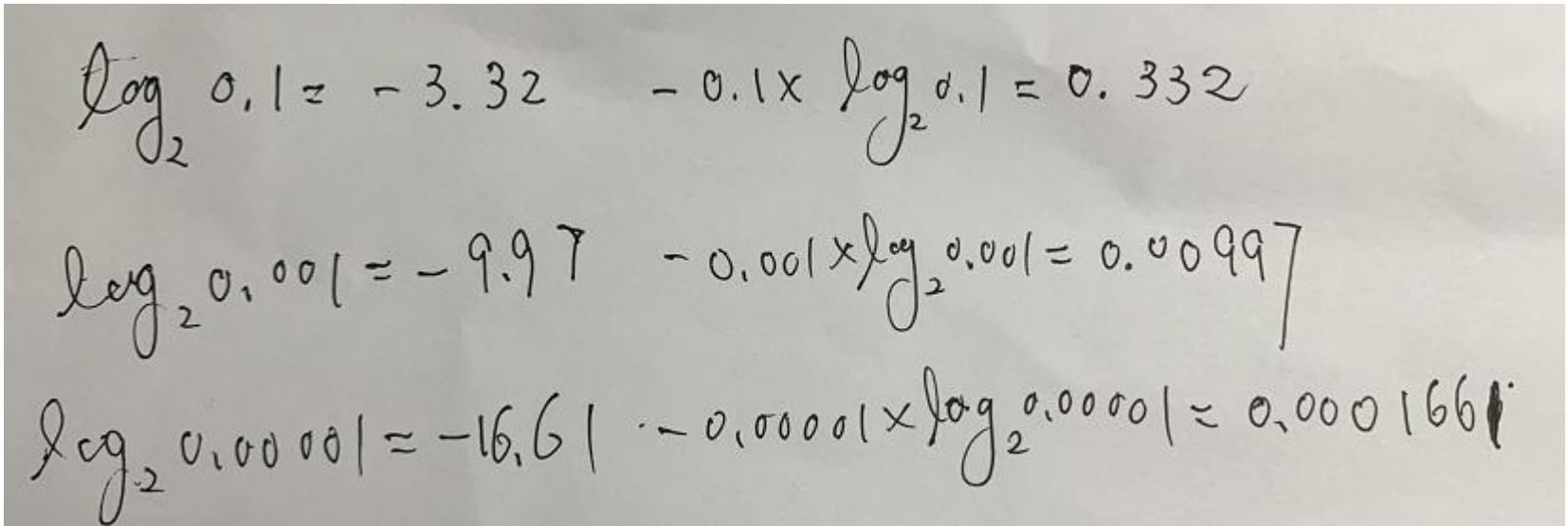
$$INFORMATION = -\log_2(0.1) = -(-3.223) = 3.223$$

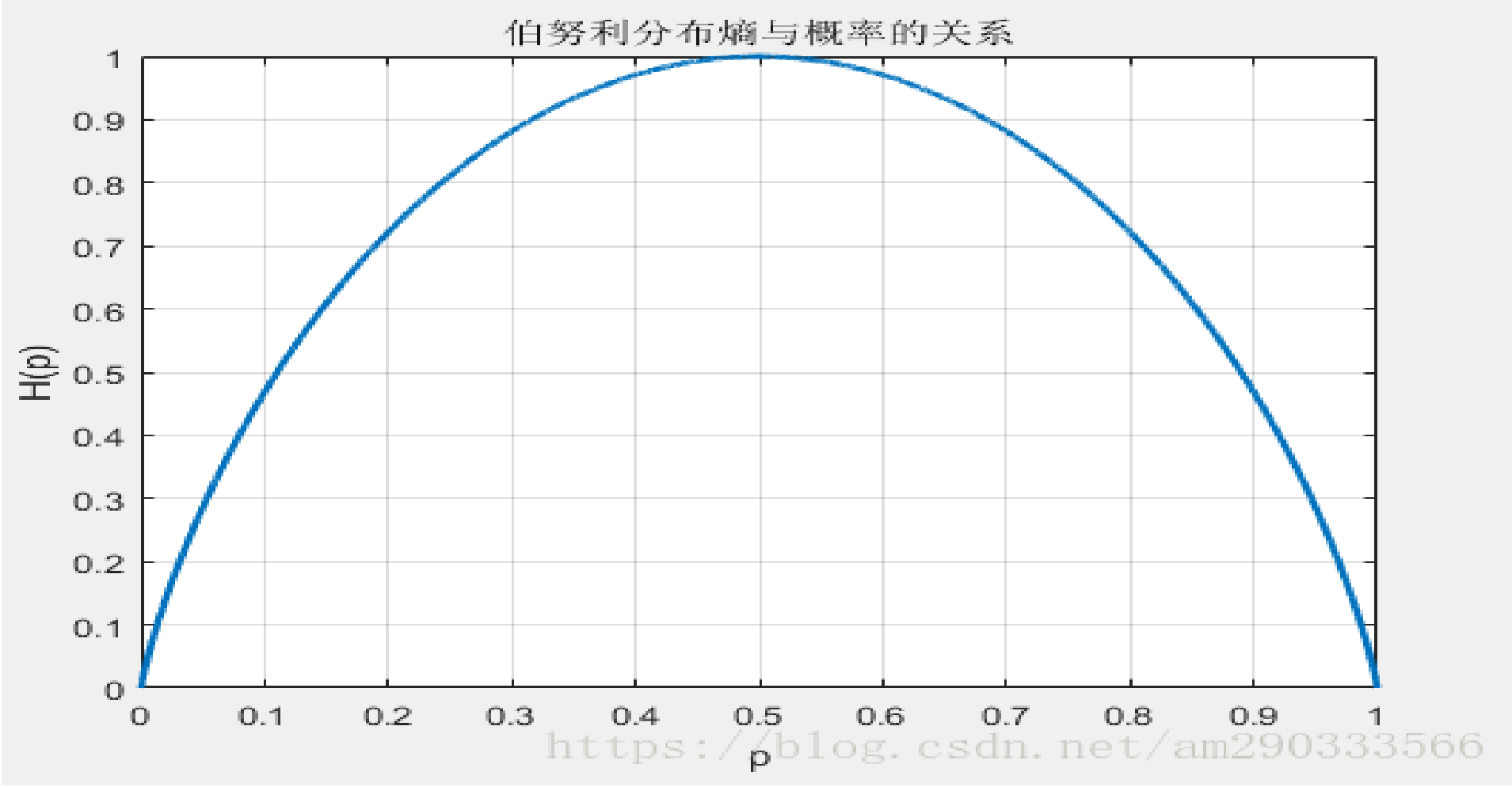
Reduced uncertainty: when terms are assigned as content identifiers to the documents of a collection.
The smaller the probability of a term, the higher is the reduction in uncertainty.

3.2 Term的自动抽取及其加权

The average, or expected information of a term (the average reduction in uncertainty) over a dictionary W:

$$AVERAGE-INFORMATION(ENTROPY)=-\sum_{k=1}^{t} p_k \log_2 p_k$$





作为一个具体的例子，当随机变量只取两个值，例如 1，0 时，即 X 的分布为：

$$P(X=1)=p, \quad P(X=0)=1-p, \quad 0 \le p \le 1$$

熵为：

$$H(p) = -p\log_2 p - (1-p)\log_2(1-p)$$

这时，熵 $H(P)$ 随概率 p 变化的曲线　　（单位为比特）

3.2 Term的自动抽取及其加权

- The inverse document frequency weight

Use of relative frequency: the document frequency of term k DF_k

The inverse document frequency:

$$IDF_k = \log_2 \frac{n}{DF_k} + 1 = \log_2(n) - \log_2(DF_k) + 1$$

where n is the number of documents in the collection.

e.g. n=1000; A:100; B:500; C:900

A:4.322; B:2.000; C:1.132

IDF example, suppose N = 1 million

term	DF_k	IDF_k
calpurnia	1	
animal	100	
sunday	1,000	
fly	10,000	
under	100,000	
the	1,000,000	

$$\mathrm{IDF}_k = \log_{10}(N/\mathrm{DF}_k)$$

There is one IDFf value for each term k in a collection.

Collection vs. Document frequency

Word	Collection frequency	Document frequency
insurance	10440	3997
try	10522	8760

- Which word is a better search term (and should get a higher weight)?

TF*IDF: $WEIGHT_{ik} = TF_{ik} \times IDF_k$

Binary → count → weight matrix

	Antony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth
Antony	5.25	3.18	0	0	0	0.35
Brutus	1.21	6.1	0	1	0	0
Caesar	8.59	2.54	0	1.51	0.25	0
Calpurnia	0	1.54	0	0	0	0
Cleopatra	2.85	0	0	0	0	0
mercy	1.51	0	1.9	0.12	5.25	0.88
worser	1.37	0	0.11	4.15	0.25	1.95

Effect of IDF on ranking

- Does idf have an effect on ranking for one-term queries, like
  - iPhone
- idf has no effect on ranking one term queries
  - idf affects the ranking of documents for queries with at least two terms
  - For the query capricious person, idf weighting makes occurrences of capricious count for much more in the final document ranking than occurrences of person.

3.2 Term的自动抽取及其加权

- The signal-noise ratio

The noise of an index term k for a collection of n documents:

$$NOISE_k = -\sum_{i=1}^{n} \frac{TF_{ik}}{TTF_k} \log_2 \frac{TF_{ik}}{TTF_k}$$

“concentration” of a term

perfectly even distribution(the noise is maximized):

$$NOISE_k = -\sum_{i=1}^{n} \frac{1}{n} \log_2 \frac{1}{n} = \log_2 n$$

perfectly concentrated distribution(the noise is zero):

$$NOISE_k = -\frac{TTF_k}{TTF_k} \log_2 \frac{TFF_k}{TTF_k} = 0$$

3.2 Term的自动抽取及其加权

The signal of an term k for a collection of n documents:

$$SIGNAL_k = \log_2(TTF_k) - NOISE_k \geq 0$$

Minimized: 0

Maximized: $SIGNAL_k = \log_2(TTF_k)$

$$WEIGHT_{ik} = TF_{ik} \times SIGNAL_k$$

3.2 Term的自动抽取及其加权

- The term discrimination value

$$AVERAGE\ -SIMILARITY\ = CONSTANT \sum_{i=1, i\neq j}^{n} \sum_{j=1}^{n} SIMILAR\ (D_i, D_j)$$

Density of the document space = AVERAGE-SIMILARITY

When all SIMILAR(D_i,D_j)=1, maximized:

$$AVERAGE - SIMILARITY = CONSTANT \sum_{i=1, i\neq j}^{n} \sum_{j=1}^{n} 1 = CONSTANT \times n(n-1)$$

(Let CONSTANT=1/n(n-1),
then AVERAGE-SIMILARITY=1)

3.2 Term的自动抽取及其加权

For high-frequency term: fairly even frequency distribution
  removal: reduce AVGSIM (addition: increase AVGSIM)

For more concentrated distribution:
  removal: increase AVGSIM (addition: reduce AVGSIM)

3.2 Term的自动抽取及其加权

Let $AVGSIM_k$ be the space density with term k removed from all the documents, define

the discrimination value $DISCVALUE_k$ for each term k:

$$DISCVALUE_k = AVGSIM_k - AVGSIM$$

$DISCVALUE_k > 0$, good discriminators
$DISCVALUE_k \approx 0$, indifferent discriminators
$DISCVALUE_k < 0$, poor discriminators

$$WEIGHT_{ik} = TF_{ik} \times DISCVALUE_k$$

3.2 Term的自动抽取及其加权

Density of the document space by the centroid:

$$AVERAGE - TF_k = \frac{1}{n}\sum_{i=1}^{n} TF_{ik}$$

$$AVGSIM = CONSTANT \sum_{i=1}^{n} SIMILAR(\overline{D}, D_i)$$

Table 3-3 Best and Worst Discriminators for Three Collections

(Cranfield: 424 Documents in Aerodynamics; MED: 450 Documents in Medicine; *Time:* 425 Documents in World Affairs)

Cranfield 424	MED 450	*Time* 425
a Best discriminators		
1. Panel	1. Marrow	1. Buddhist
2. Flutter	2. Amyloidosis	2. Diem
3. Jet	3. Lymphostasis	3. Lao
4. Cone	4. Hepatitis	4. Arab
5. Separate	5. Hela	5. Viet
6. Shell	6. Antigen	6. Kurd
7. Yaw	7. Chromosome	7. Wilson
8. Nozzle	8. Irradiate	8. Baath
9. Transit	9. Tumor	9. Park
10. Degree	10. Virus	10. Nenni
b Worst discriminators		
2642. Equate	4717. Clinic	7560. Work
2643. Theo	4718. Children	7561. Lead
2644. Bound	4719. Act	7562. Red
2645. Effect	4720. High	7563. Minister
2646. Solution	4721. Develop	7564. Nation
2647. Method	4722. Treat	7565. Party
2468. Press	4723. Increase	7566. Commune
2649. Result	4724. Result	7567. U.S.
2650. Number	4725. Cell	7568. Govern
2651. Flow	4726. Patient	7569. New

Adapted from reference 19.

Table 3-5 Highest-Ranking Discriminator and Nondiscriminator Terms

(852 Abstracts in Ophthalmology)

Nondiscriminators				Discriminators			
Term	Document frequency	Total frequency	Average frequency	Term	Document frequency	Total frequency	Average frequency
8672. Patient	201	408	2.03	1. Rubella	10	47	4.70
8671. At	194	292	1.51	2. Capillary	19	54	2.84
8670. Use	179	247	1.38	3. Laser	11	32	2.91
8669. Have	194	257	1.32	4. Collagen	12	40	3.33
8668. Retinal	134	275	2.05	5. Cyst	17	42	2.47
8667. Present	184	219	1.19	6. Cholinesterase	6	26	4.33
8666. Has	171	231	1.35	7. Fiber	16	50	3.13
8665. Effect	150	259	1.73	8. Cyclodialysis	4	12	3.00
8664. Result	179	234	1.31	9. Implant	18	36	2.00
8663. Found	174	228	1.31	10. Uveitis	21	45	2.14
8662. Report	141	172	1.22	11. Vessel	36	82	2.28
8661. Occular	125	194	1.55	12. Spray	2	25	12.50

Comparison 1 (425 documents, 24 test queries, Times)

Recall	Binary weights BIN_{ik}	Term frequency weights $FREQ_{ik}$	Binary with IDF weights $BIN_{ik}/DOCFREQ_k$	Term frequency with IDF $FREQ_{ik}/DOCFREQ_k$
0.1	0.8257	0.7496	0.8085	0.8536
0.2	0.7555	0.7071	0.7741	0.7901
0.3	0.6754	0.6710	0.7114	0.7568
0.4	0.6224	0.6452	0.6328	0.7503
0.5	0.5708	0.6351	0.6218	0.6783
0.6	0.5299	0.5866	0.5673	0.6243
0.7	0.4618	0.5413	0.5124	0.5823
0.8	0.4087	0.5004	0.4384	0.5643
0.9	0.2959	0.3865	0.3374	0.4426
1.0	0.2854	0.3721	0.3188	0.4170

Table 3-15 Comparison of binary term weighting with inverse document frequency (IDF) weights (time collection, 425 documents, 24 search requests). (a) Comparison of binary and term frequency weighting with and without inverse document frequency normalization.

(1) BIN wins in low recall end, FREQ wins in high recall end;
(2) FREQ/DOCFREQ (significantly) better than BIN/DOCFREQ

Comparison 2 (425 documents, 24 test queries, Times)

Table 3-16 Recall-Precision Results for Term Deletion
(*Time* 425 Collection, 24 Queries)

Recall	Standard BIN_{ik} weights	Standard $FREQ_{ik}$ weights	$FREQ_{ik}$ weights	
			IDF CUT	DISC CUT
0.1	0.8257	0.7496	0.8601	0.7911
0.2	0.7555	0.7071	0.8268	0.7485
0.3	0.6754	0.6710	0.7503	0.7362
0.4	0.6224	0.6452	0.7144	0.7000
0.5	0.5708	0.6351	0.6872	0.6777
0.6	0.5299	0.5866	0.6168	0.6350
0.7	0.4618	0.5413	0.5645	0.5907
0.8	0.4087	0.5004	0.5017	0.5510
0.9	0.2959	0.3865	0.4071	0.4177
1.0	0.2854	0.3721	0.3906	0.4019

IDF = inverse document frequency
DISC = discrimination value

IDF CUT: DF >=104 removed; DISC CUT: negative removed

(1) Removal better than composite IDF at low and medium recall;

(2) IDF CUT and DISC CUT significantly better than FREQ

Comparison 3 (425 documents, 24 test queries, Times)

Composite weighting functions

$FREQ_{ik}$ weights with IDF $FREQ_{ik}/DOCFREQ_k$	$FREQ_{ik}$ weights with $DISCVALUE_k$ $FREQ_{ik} \cdot DISCVALUE_k$	$FREQ_{ik}$ weights with $SIGNAL_k$ $FREQ_{ik} \cdot SIGNAL_k$	$FREQ_{ik}/DOCFREQ_k$ with IDF CUT	$FREQ_{ik} \cdot DISCVALUE_k$ with DISC CUT
0.8536	0.8406	0.7212	0.8975	0.8028
0.7901	0.7881	0.7006	0.8315	0.7480
0.7568	0.7197	0.6471	0.7800	0.7286
0.7305	0.6901	0.6229	0.7574	0.6938
0.6783	0.6704	0.6105	0.7372	0.6737
0.6243	0.6176	0.5587	0.6529	0.6349
0.5823	0.5727	0.5263	0.5912	0.5847
0.5643	0.5169	0.4612	0.5481	0.5475
0.4426	0.4208	0.3830	0.4318	0.4259
0.4170	0.4053	0.3593	0.4118	0.4085

(1) FREQ/DOCFREQ with IDF CUT significantly better at low and medium recall;

(2) SIGNAL is not as good as expected