Information Retrieval

阮志 (Richard Sutcliffe)

Introduction
介绍

Information about 阮志

Information Retrieval

信息检索

Natural Language Processing

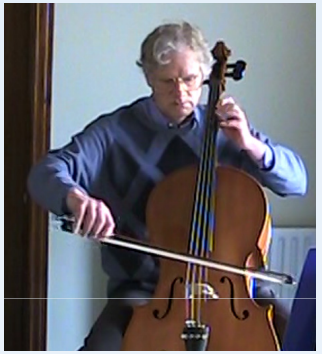
自然语言处理

Music Information Retrieval

音乐信息检索

Musicology

音乐学



I play Cello, Double Bass

演奏低音提琴

Piano

钢琴

Sail boats

驾驶帆船

More Information about the Lecturer
有关讲师的更多信息

First degree: St Andrews University, Scotland, Ph.D. University of Essex

Lectured at University of Exeter, UK, University of Limerick, Ireland, University of Essex, UK and now Northwest University China

Participant in TREC (English Question Answering), CLEF (French-English Question Answering) and NTCIR (Chinese Question Answering)

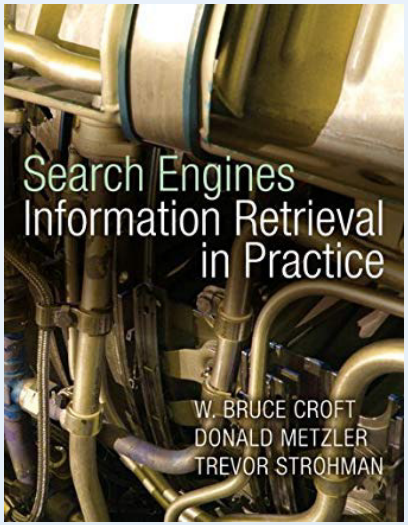
Organiser of multilingual QA evaluations at CLEF for ten years

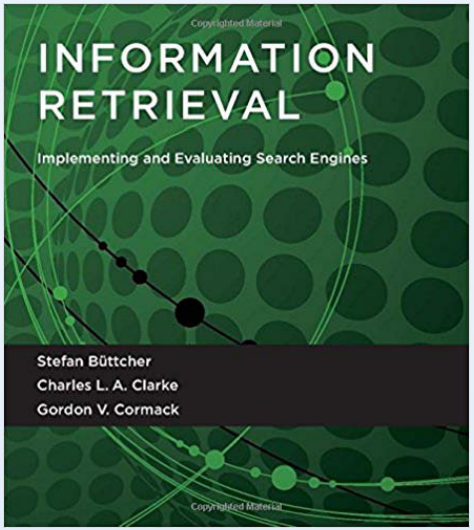
Organiser of C@merata track on Music Information Retrieval at MediaEval for four years

Books
图书

Search Engines: Information Retrieval in Practice
W. Bruce Croft, Donald Metzler, Trevor Strohman
Pearson Education
2015
http://ciir.cs.umass.edu/ downloads/
SEIRiP.pdf - on Moodle

Information Retrieval: Implementing and Evaluating Search Engines
Stefan Büttcher, Charles L. A. Clarke, Gordon V. Cormack
MIT Press
2016

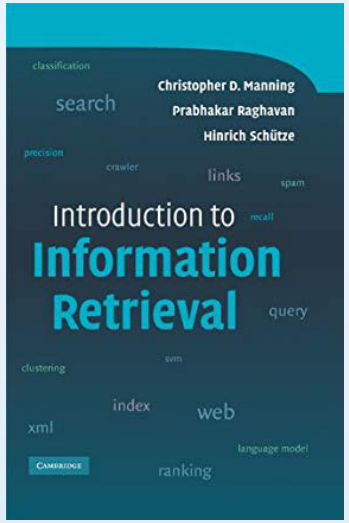


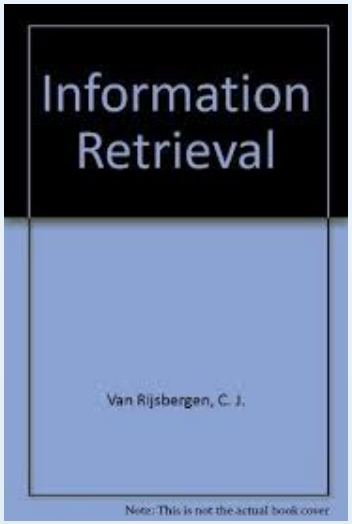


Books
图书

Introduction to Information Retrieval
C. D. Manning, P. Raghavan, H. Schütze
Cambridge University Press
2008
http:// nlp.stanford.edu/IR-book/

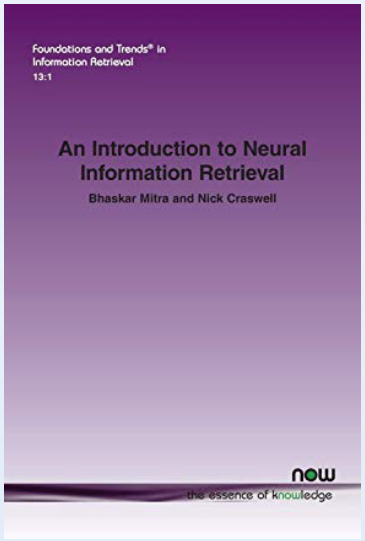
Information Retrieval
Keith van Rijsbergen
Butterworths
1979!
http://www.dcs.gla.ac.uk/Keith/Preface.html





Books
图书

An Introduction to Neural Information Retrieval
Bhaskar Mitra, Nick Craswell
Now Publishers
2018 - on Moodle



Assessment
评定

20% Lab work

20% Project

60% Final Exam

100%

Moodle site for Course
moodle网站

You can find:

- Lecture notes
- Papers
- Forum
- etc.

Introduction to Information Retrieval
信息检索简介

What is Information Retrieval?
什么是信息检索？

During the last forty years, huge amounts of information have been stored on computers around the world.

With the dawn of the WWW this information is more readily available than ever before.

People need information to solve problems.

The central question of IR: How do you find the information you want?

如何查找信息？

Traditional IR
传统的IR如何运作

Information is in the form of text stored in a number of separate files.

An Information need is specified as a short query comprising a text string.

The system responds to the query with an ordered list of files which match the query.

The user then looks through the files and hopefully finds the answer to their query.

文本文件...短文本查询...返回文件的有序列表...用户浏览文件

Why is IR Important - Brief History
简要历史 - 为什么IR很重要

In the 1950s, IR was of interest mainly to people cataloguing libraries;

In 1970s, Inverted Indexing was invented by Gerry Salton at Cornell. This allowed large collections; however, still a fringe activity;

In 1990s, Web started to become more important; Search engines created;

In 2010s, search engines are a huge business; Google, Bing, Baidu etc. are now very important organisations.

20世纪50年代只是图书馆... 20世纪70年代倒装索引发明...
20世纪90年代搜索引擎... 2010年的大生意

Fields Related to IR
与IR相关的软件应用程序

Question Answering (IBM Watson)

Input is a query: Who is the wife of Tom Cruise?
Output is exact: Katie Holmes (until 2012 anyway!)

Information Extraction (e.g. template filling)

Input: document of a certain type (e.g. company takeover)

Output: Key facts - Who took over What, When, Why, for How much

问题回答...信息提取

Database Queries (e.g. SQL) vs. Information Retrieval (Manning book)
数据库与IR

	Database Retrieval	**Information Retrieval**
Matching	Exact match	Partial or best match
Inference	Deduction	Induction
Model	Deterministic	Probabilistic
Query Lang	Artificial	Natural
Query Spec	Complete	Incomplete
Items wanted	Matching	Relevant
Error response	Sensitive	Insensitive

将SQL数据库与信息检索进行比较（复杂 - 稍后我们将回到此处）

Deduction vs. Induction

Inductive reasoning is a method of reasoning in which the premises are viewed as supplying some evidence, but not full assurance, for the truth of the conclusion.
https://en.wikipedia.org/wiki/Inductive_reasoning

该

Deductive reasoning, also deductive logic, is the process of reasoning from one or more statements (premises) to reach a logical conclusion.
https://en.wikipedia.org/wiki/Deductive_reasoning

Deterministic vs. Probabilistic

Determinism is the philosophical view that all events are determined completely by previously existing causes.
https://en.wikipedia.org/wiki/Determinism

该

Determinism in IR or DBs: Given certain inputs we always get the same output. Database queries in SQL are like this.

Probabilistic reasoning is a method of representation of knowledge where the concept of probability is applied to indicate the uncertainty in knowledge.
https://study.com/academy/lesson/probabilistic-reasoning-artificial-intelligence.html

Probabilistic reasoning in IR: Keywords etc in a text are evidence that it could be relevant but this is not certain; the more keywords, the more certain.

DB vs. IR cont.
数据库与IR

DB:

- Convert data into structured form (Normal forms etc).
- Search precisely using Relational Model.
- Cannot change the overall framework.

将数据转换为结构化形式...精确搜索...无法改变框架

IR:

- Do not convert data to structured form.
- Search less precisely using keywords etc.
- Can change the overall framework - just alter search methods.

不要将数据转换为结构化形式...不太精确地搜索...可以改变框架

Hybrids of DB and IR
数据库和IR的混合组合

NoSQL Databases are one trend.

NoSQL是一种新型数据库。

MongoDB is an example:

MongoDB是一种NoSQL数据库。

- Text documents are structured
- Queries are also structured
- Large amounts of data
- Structured queries (but not SQL)
- No ‘normalisation’-style guarantees (normalised SQL database cannot be inconsistent)

Most Important Idea in IR: Inverted Indexing
IR中最重要的想法是反向索引

Naïve IR - Forward Index

浏览每个寻找关键字的文件.....慢

Input: one or more query terms (words)
Output: documents containing the query terms

We start by going through each document and making a list of the words it contains

Then, **go through each doc's list in turn** seeing whether it contains the query terms. If it does, return doc.

Called **Forward Index**: Docid → Term

Problem: Hopelessly **slow**! Time for retrieval will go up linearly with number of docs! Also goes up linearly with number of words in docs!

Unworkable!

First Idea cont.
IR中最重要的想法是反向索引

Solution - Inverted Index

倒排索引：输入关键字并获取包含关键字的所有文档的列表

Instead of Docid → Term we have Term → Docid

In other words, input a Term and in one operation we can produce a list of docs which *contain* that term.

For multiple query terms we must merge resulting lists of Docids.

Advantage: Very fast to produce lists - time does not increase (much) with size of doc collection. Merging of lists is slower but does not need to be complete (see later).

Disadvantage: Need to **create** inverted index from forward index. This is slow but only needs doing 'once'.

快速使用，创建缓慢！

SB

Second Most Important Idea in IR - Term Weighting
第二个想法，术语加权

Naive IR - Keyword Match

If a query term is in a doc, it matches. Otherwise it does not. Just return docs which contain matches and that is it.

Basis of *Boolean Retrieval*.

Advantage: Simplicity

Disadvantage: Does not take account of **how important** the word is; does not allow **ordering** of matching docs (called Ranking).

按关键字匹配，简单但忽略了单词的重要性

Advantages of Term Weighting
术语加权的优点

Dramatically improves the performance of IR in terms of the **relevance** of the result.

Makes system **no slower**.

获得更多相关文件。没有慢。

Allows ranking (i.e. ordering) of results - **most relevant** come first. Extremely important in large doc collections.

Nobody quite knows why it works!

TF*IDF
我们发音：“TF IDF”

Term Frequency: How often Term occurs in a Doc. 术语在文档中出现的频率

Hypothesis: The more often term appears, the more the doc is about the concept denoted by the term.

e.g. the more often 'horse' appears in a doc, the more the doc is about horses!

Inverted Document Frequency: *Reciprocal* of how often Term appears in entire doc collection.

将所有文档一起添加到集合中并找到术语频率f ...我们取1 / f

Hypothesis: The more often the term appears across the doc collection as a whole, the less useful it is for distinguishing **between** docs.

e.g. if 'horse' appears in all the docs, not just one or two, then 'horse' is no good for distinguishing between docs, because all docs will match!

Properties of TF and IDF
TF和IDF的特征

Term Frequency = TF
Inverted Document Frequency = IDF

文档中的术语
频率越高越好

The higher the freq of a term in a doc, the better the doc matches our query term.

On the other hand, the higher the freq of the term in docs generally, the worse it is as a search term.

So we take 1/DF (i.e. IDF) because it becomes worse for us as it becomes more frequent.

We want the best 'combination' of these factors.

随着整个文档集合的频率上升，IDF下降

Hence, multiply TF by IDF to get an overall weight for term.

所以，用IDF乘以TF

Call this **TF*IDF**

OKAPI
OKAPI，TF * IDF的替代品

Alternative to TF*IDF.

来自TF * IDF的不同算术但仍然导致术语的权重

Developed by Robertson et al. (1994).

Based on similar principles but developed from a different theoretical standpoint (a probabilistic retrieval model).

Still relies on the fundamental principles of TF and IDF.

We will look later at algorithms for computing TF, IDF and OKAPI.

Summary of Introduction to IR
摘要

IR returns ordered list of matching documents in response to keyword query.

Inverted Indexing makes fast searching possible on large collections. The basis of Google.

Term weighting is crucial to efficient retrieval.

TF*IDF is the dominant paradigm for term weighting.

IR has complementary properties to DB searching.

IR返回有序的文档列表

反向索引使其快速

术语加权很重要

TF * IDF是术语加权的主要范例

IR搜索具有与数据库搜索不同的属性

Words and Related Algorithms
单词和相关算法

Common Processing Stages
常见的加工阶段

Separate the document into words (tokenisation);

Bring all words down to lower case;

Eliminate high-frequency words;

Eliminate differences between words caused by prefixes, suffixes and infixes (stemming).

分开的话
小写
没有高频词
词干

Tokenisation for European Languages
欧洲语言的标记化

Tokenisation is the process of extracting all of the individual words from a document.

It is not easy to perform accurately.

从文档中提取所有单词

It is language specific.

Tokenisation for European Languages cont.
欧洲语言的标记化

Can use finite state methods for this, based on regular expressions.

Mostly, can separate words by spaces.

另请参阅

Hard to make it completely accurate:

- Full stop ('.') can end sentence
- Full stop ('.') can occur in a word ('Section 3.1')
- Dash ('-') can occur in middle of sentence
- Dash ('-') can occur in a word ('Section 3-1')

我们可以匹配的各种字母

Need to make tokenisation as efficient as possible.

Tokenisation for Chinese
中文的标记化

Of course, Chinese is very different from English.

A Chinese word is usually made up of 1-5 hànzì.

Many common words are 2 hànzì.

一个字有多长？许多常用词是两个汉字

More specific or technical terms tend to be 3 or more hànzì.

The precise breakdown depends on the type of text.

Possible tools are:

结巴 (Jiēbā) (means to stammer)
https://github.com/fxsjy/jieba

THULAC
https://github.com/thunlp/THULAC

Tokenisation for Chinese cont.
中文的标记化

To index a Chinese word (e.g. 兔子 Tùzi rabbit), we can use

individual hànzì only	兔　子
words only, i.e. hànzì pairs etc.	兔子
hànzì + words	兔　子　兔子

单独的汉字索引，单词，汉字和单词

Stemming
与中国人相比，英语中的形态变化

In English, the form of a word changes:

a **dog**	一只狗 (Yī zhǐ gǒu)	中文 same
three **dogs**	三只狗 (Sān zhī gǒu)	中文 same
I **eat** cake	我吃蛋糕 (Wǒ chī dàngāo)	中文 same
She **eats** cake	她吃蛋糕 (Tā chī dàngāo)	中文 same
I **ate** cake	我吃了蛋糕 (Wǒ chīle dàngāo)	了= finish
I am **eating** cake	我在吃蛋糕 (Wǒ zài chī dàngāo)	在= in progress
I was **eating** cake	我在吃蛋糕 (Wǒ zài chī dàngāo)	在= in progress
I have **eaten** cake	我吃过蛋糕 (Wǒ chīguò dàngāo)	过= over

Stemming cont.
与中国人相比，英语中的形态变化

So in English, same word, different endings

for nouns:
- singular ending
- plural ending

在英语中有许多不同的单词结尾。

所以我们需要将每个单词转换为标准形式。

for verbs:
- infinitive form (eat)
- third person singular form (eats)
- simple past form (ate)
- past participle (eaten)

Note on Syntactic Case
句法案例

In some languages, word changes depending on whether it is subject, object etc

In English, this only affects pronouns:

✓ I gave her a book
× I gave she a book
✓ She gave me a book
× She gave I a book

句法案例：动词之前与动词之后的单词拼写。

In English:
green = right, good
red = wrong, bad

In Chinese:
red = right, good
green = wrong, bad

Stemming in IR (for English)
词干

As we saw, terms with a common stem usually have similar meanings. e.g.

connect, connected, connecting, connection, connections

同一个词，不同的拼写
我们想纠正这个

In IR, performance is usually improved if groups of words having a common stem are treated as instances of the same search word. Example:

document contains ‘connecting’
contains ‘connect’
→ we want a match

我们希望这些话能够匹配

但是，它们拼写不同

The process of reducing different forms of a word to a stem is known as **stemming**.

The best known algorithm is due to M. F. Porter (1980). 着名的方法

Martin F. Porter 1944-present

Martin F. Porter is the inventor of the Porter Stemmer, one of the most common algorithms for stemming English, and the Snowball programming framework.

His 1980 paper "An algorithm for suffix stripping", proposing the stemming algorithm, has been cited over 8,000 times.

Wikipedia

The Porter Stemming Algorithm
Porter词干算法

Remarkably clever and not surpassed for English to this day!

Based on two principles:

1. Take suffixes off a string using patterns, but 从一个词开始起飞

2. Do not take off too much (using Measure concept). 但是不要起飞太多！

'Measure' in Porter Stemming
Measure('措施')的概念

A vowel is one of the letters a, e, i, o and u.

All other letters excluding y are consonants.

在英语中，如何查找字母是元音还是辅音

The letter y is a *consonant* if the preceding letter is a, e, i, o, u, or if it starts the word; Otherwise, y is a *vowel*.

字母'y'既是元音又是辅音!!

Examples of Vowel and Consonant
元音和辅音的例子

toy contains the consonants t and y. o is a vowel.

syzygy contains the consonants s, z and g. Here, each occurrence of y is a vowel.

yym contains the consonant y, the vowel y and the consonant m, in that order.

对于字母'y'，我们遵循上一张幻灯片中的规则。这取决于具体情况。

对于所有其他字母，它们要么是元音要么是辅音字母。它一直都是一样的。

c, C, v, V in Porter
Porter算法中的c，C，v，V

A single consonant is denoted by c and a single vowel by v.
A list ccc... of length greater than 0 is denoted by C.

A list vvv... of length greater than 0 is denoted by V.

Any word has one of four forms:

CVCV...C
CVCV...V
VCVC...C
VCVC...V

每个字都适合这种模式

Měi gè zì dōu shìhé zhè zhǒng móshì

c是一个辅音
v是一个元音
几个v被称为V.
几个c被称为C.

c shì yīgè fǔyīn
v shì yīgè yuán yīn
jǐ gè v bèi chēng wèi V
Jǐ gè c bèi chēng wèi C

c, C, v, V in Porter cont.
Porter算法中的c，C，v，V

CVCV...C
CVCV...V
VCVC...C
VCVC...V

can be denoted

[C] VCVC... [V]

where [C] means an optional C and [V] means an optional V.

can be re-written as:
[C] $(VC)^m$ [V]

m is called the **measure**

'm'的值是单词的'度量'(measure)

[C]表示可选C
[V]表示可选V
[C] biǎoshì kě xuǎn C

$(VC)^m$ means VC repeated m times

VC重复'm'次
VC chóngfù'm'cì

What is Measure?
“措施”是什么意思？

Measure is a number

这是一个数字

Indicates how long a word is

表示一个单词有多长

If a word is 'too short' we do not take off an ending

如果单词太短，我们不会从它的末尾删掉字母

See later for more

Examples of Measure = 0
Measure = 0的字符串示例

tr	C	C $(VC)^0$ []	measure=0
ee	V	[] $(VC)^0$ V	measure=0
tr-ee	CV	C $(VC)^0$ V	measure=0
y	C	C $(VC)^0$ []	measure=0
b-y	CV	C $(VC)^0$ V	measure=0

Match this pattern to string to find measure:

[C] $(VC)^m$ [V]

将此模式与字符串匹配。

然后，查看m的所需值。

其次，将C，V的序列与公式匹配，找到'm'。
First, convert the string to a sequence of C and V.

首先，将字符串转换为C和V的序列。
Second, match the sequence of C and V with the formula to find 'm'.

Examples of Measure = 1
Measure = 1的字符串示例

tr-ou-bl-e	C (VC) V	C $(VC)^1$ V	measure=1
oa-ts	(VC)	[] $(VC)^1$ []	measure=1
tr-ee-s	C (VC)	C $(VC)^1$ []	measure=1
i-v-y	(VC) V	[] $(VC)^1$ V	measure=1

Match this pattern to string to find measure:

$[C]\ (VC)^m\ [V]$

将此模式与字符串匹配。

然后，查看m的所需值。

Examples of Measure = 2
Measure = 2的字符串示例

tr-ou-bl-e-s	C (VCVC)	C $(VC)^2$	measure=2
pr-i-v-a-t-e	C (VCVC) V	C $(VC)^2$ V	measure=2
oa-t-e-n	(VCVC)	$(VC)^2$	measure=2
o-rr-e-r-y	(VCVC) V	$(VC)^2$ V	measure=2

Match this pattern to string to find measure:

[C] $(VC)^m$ [V]

将此模式与字符串匹配。

然后，查看m的所需值。

Form of Rules
规则的句法格式

(*condition*) S1 → S2

'（condition）'是一个布尔表达式
'(condition)' is a boolean expression

if word ends in S1

如果单词以字符串S1结尾

and word before S1 matches condition

并且S1之前的字母与条件匹配

then replace S1 with S2

然后用S2替换S1

Mostly, *condition* is something like *m > 0*

Note: S2 can be null string (written ε for Greek *epsilon*)

If S2 is ε, S1 is removed.

Examples of Rules - 1
规则的例子

Rule: s → ε

Word: cats

Method:

1. Apply rule to end of word: cats → cat

2. Check condition. No condition.

3. So rule **applies**. 'cats' becomes 'cat'.

单词末尾的字母's'映射到epsilon。
The letter 's' at the end of the word maps to epsilon.

规则开头没有（条件）。
There is no (condition) at the beginning of the rule.

Examples of Rules - 2
规则的例子

Rule: (m>0) eed → ee

Word: feed

Method:

1. Apply rule: feed → fee

2. Check *measure of resulting stem* fee

f-ee C()V m=0

3. Check condition: m>0 **not satisfied**

4. So rule does not apply. 'feed' stays as 'feed'

feed - give someone food

fee - something you pay

so feed should not become fee!

Use this:

[C] $(VC)^m$ [V]

to find measure

（条件）不满意

规则不适用。

Examples of Rules - 3
规则的例子

Rule: (m>0) eed → ee

Word: agreed

Method:

1. Apply rule: agreed → agree

2. Check measure of resulting stem agree

a-gr-ee (VC)V m=1

3. Check condition: m>0 **satisfied**

4. So rule applies. 'agreed' becomes 'agree'

'agreed' and 'agree' are different forms of same word

so, change 'agreed' to 'agree'

Use this:
[C] $(VC)^m$ [V]
to find measure

（条件）满意。

该规则适用。

Examples of Rules - 4
规则的例子

Rule: (m>1) ance → ε

Word: allowance

Method:

1. Apply rule: allowance → allow

2. Check measure of resulting stem allow

a-ll-o-w (VCVC) m=2

3. Check condition: m>1 **satisfied** （条件）满意。

4. So rule applies. 'allowance' becomes 'allow' 该规则适用。

Use this:
[C] $(VC)^m$ [V]
to find measure

Examples of Rules - 5
规则的例子

Rule: (m>1) ance → ε

Word: trance

Method:

1. Apply rule: trance → tr !!! tr is not a word!!

2. Check measure of resulting stem tr

tr C () m=0

3. Check condition: m>1 **not satisfied** （条件）不满意

4. So rule does not apply. 'trance' stays as 'trance' 规则不适用。

Use this:
$[C]\ (VC)^m\ [V]$
to find measure

Examples of Rules - 6
规则的例子

Rule: (m>1) ance → ε

Word: entrance

Method:

1. Apply rule: entrance → entr !!!

2. Check measure of resulting stem entr

e-ntr (VC) m=1

3. Check condition: m>1 **not satisfied** （条件）不满意

4. So rule does not apply. 'entrance' stays as it is 规则不适用。

Use this:
[C] $(VC)^m$ [V]
to find measure

Expressions used in Porter Rule Conditions
“Porter”规则条件中使用的字符串表达式

*S	The stem ends in s. Similarly for other letters.	茎以's'结尾。
v	The stem contains a vowel.	茎包含元音。
*d	The stem ends in a double consonant, e.g. tt, ss.	茎以双辅音结束。
*o	The stem ends cvc where the second c is not w, x, or y. (e.g. wil, hop).	茎端'cvc'。

The condition may also contain boolean expressions using *and*, *or* and *not*.

（condition）也可以包含使用AND，OR，NOT的布尔表达式

Example Conditions
（条件）的例子

(m=1)	The measure of the stem before S1 is 1. S1之前的词干“度量”为1。
(m>1 and (*S or *T))	The measure of the stem before S1 is greater than 1 and the stem ends in s or t 在S1之前的茎的量度大于1并且茎以's'或't'结束。
(*d and not (*L or *S or *Z))	The stem before S1 ends in a double consonant other than l, s or z. S1之前的词干以“l”，“s”或“z”之外的双辅音结束。

Application of Rules
规则的应用

The rules are written in several groups.

规则分为几组。

In a set of rules written as a group, if several rules match, the rule with the longest matching S1 is used.

如果组中的多个规则匹配，则使用具有最长匹配S1的规则。

Example Rules:

SSES → SS
IES → I
SS → SS
S → ε

Application of Rules:

caresses maps to caress since sses is the longest match for S1. Thus the last rule is not used even though it matches.

caress maps to caress (S1=ss) even though the last rule also matches.

cares maps to care (S1=s)

Porter Step 1a
“Porter”规则的第1a步

SSES	→	SS	caresses → caress
IES	→	I	ponies → poni
			ties → ti
SS	→	SS	caress → caress
S	→	ε	cats → cat

Porter stems are not always proper words

e.g. poni vs. pony
e.g. ti vs. tie

Porter Step 1b Part I
“Porter”规则的第1b I步

(m>0)	EED	→	EE	feed → feed agreed → agree
(*v*)	ED	→	ε	plastered → plaster bled → bled
(*v*)	ING	→	ε	motoring → motor sing → sing

Stem ‘plaster’ contains a vowel
Stem ‘bl’ does not contain a vowel

Porter Step 1b Part II
“Porter”规则的第1b II步

If the second or third rules in Step 1b is successful, the following rules are applied:

AT	→	ATE	conflat(ed) → conflate
BL	→	BLE	troubl(ing) → trouble
IZ	→	IZE	siz(ed) → size
(*d and not (*L or *S or *Z))	→	single letter	hopp(ing)→hop tann(ed)→tan fall(ing)→fall hiss(ing)→hiss fizz(ed) →fizz
(m=1 and *o)	→	E	fail(ing) → fail fil(ing) → file

如果步骤1b中的第二或第三条规则成功，则应用以下规则：

在某些情况下替换字母'e'。

删除像'pp'这样的双字母。替换为'p'。

Notes on Porter Step 1
关于'Porter'步骤1的工作原理的说明

Steps 1a and 1b reduce plurals to singular, past participles to present infinitives.

步骤1a和1b将复数减少为单数，过去分词以呈现不定式

In Step 1b Part II, the ‘e’ which was taken off when the stems -ed or -ing were removed is put on again where necessary.

在必要时将'e'放回到单词上。

In addition, certain double letters are reduced to single letters.

将像'pp'这样的双字母缩小为单个字母，如'p'。

In Step 1c, the ‘y’ at the end of some words is turned into and ‘i’. This makes e.g. ‘happy’ and ‘happiness’ both finally end up as ‘happi’, and therefore match.

有些'y'变成'i'。所以“happy”变成“happi”，“happiness”变成“happi”。因此他们匹配。

Porter Step 2
“Porter”规则的第2步

(m>0)	ATIONA	→	ATE	relational → relat
(m>0)	TIONAL	→	TION	conditional → condition
				rational → rational
(m>0)	ENCI	→	ENCE	valenci → valence
(m>0)	ANCI	→	ANCE	hesitanci → hesitance
(m>0)	IZER	→	IZE	digitizer → digitize
(m>0)	ABLI	→	ABLE	conformabli → conformable
(m>0)	ALLI	→	AL	radically → radical
(m>0)	ENTLI	→	ENT	differently → different
(m>0)	ELI	→	E	vileli → vile
(m>0)	OUSLI	→	OUS	analogously → analogous
(m>0)	IZATION	→	IZE	vietnamization → vietnamize
(m>0)	ATION	→	ATE	predication → predicate

Porter Step 2 cont.
“Porter”规则的第2步

(m>0)	ATOR	→	ATE	operator → operate
(m>0)	ALISM	→	AL	feudalism → feudal
(m>0)	IVENESS	→	IVE	decisiveness → decisive
(m>0)	FULNESS	→	FUL	hopefulness → hopeful
(m>0)	OUSNESS	→	OUS	callousness → callous
(m>0)	ALITI	→	AL	formality → formal
(m>0)	IVITI	→	IVE	sensitivity → sensitive
(m>0)	BILITI	→	BLE	sensibility → sensible

Porter Step 3
“Porter”规则的第3步

(m>0)	ICATE	→	IC	triplicate → triplic
(m>0)	ATIVE	→	ε	formative → form
(m>0)	ALIZE	→	AL	formalize → formal
(m>0)	ICITI	→	IC	electricity → electric
(m>0)	ICAL	→	IC	electrical → electric
(m>0)	FUL	→	ε	hopeful → hope
(m>0)	NESS	→	ε	goodness → good

请记住，epsilon（ε）表示空字符串。

Porter Step 4
“Porter”规则的第4步

(m>1)	AL	→	ε	revival → reviv
(m>1)	ANCE	→	ε	allowance → allow
(m>1)	ENCE	→	ε	inference → infer
(m>1)	ER	→	ε	airliner → airlin
(m>1)	IC	→	ε	gyroscopic → gyroscop
(m>1)	ABLE	→	ε	adjustable → adjust
(m>1)	IBLE	→	ε	defensible → defens
(m>1)	ANT	→	ε	irritant → irrit
(m>1)	EMENT	→	ε	replacement → replac
(m>1)	MENT	→	ε	dependent → depend
(m>1)	and (*S or*T)) ION	→	ε	adoption → adopt
(m>1)	OU	→	ε	homologous → homolog
(m>1)	ISM	→	ε	communism → commun
(m>1)	ATE	→	ε	activate → activ

Porter Step 4 cont.
“Porter”规则的第4步

(m>1)	ITI	→	ε	angularity → angular
(m>1)	OUS	→	ε	homologous → homolog
(m>1)	IVE	→	ε	effective → effect
(m>1)	IZE	→	ε	bowdlerize → bowdler

Porter Step 5a and Step 5b
“Porter”规则的第5a,5b步

Step 5a

(m>1) E	→	ε	probate → probat
(m=1 and not *o) E	→	ε	cease → ceas

Step 5b

(m>1 and *d and *L)	→	single letter	control → control roll → roll

Porter Algorithm Applied to Example Words
Porter算法应用于实例词

'X→2 Y'表示通过应用'Porter'规则2，字符串X变为字符串Y.

h-e-lpf-u-ln-e-ss →2 h-e-lpf-u-l →3 h-e-lp
h-e-lpf-u-l →3 h-e-lp
h-e-lp
dr-a-m-a-t-i-c-i-z-a-t-io-n →2 dr-a-m-a-t-i-c-i-z-e →4 dr-a-m-a-t-i-c
dr-a-m-a-t-i-c-i-s-a-t-io-n →2 dr-a-m-a-t-i-c-i-s-a-t-e →4 dr-a-m-a-t-i-c-i-s
dr-a-m-a-t-i-c-i-z-e →4 dr-a-m-a-t-i-c
dr-a-m-a-t-i-c
dr-a-m-a
dr-a-m
n-e-c-e-ss-i-t-a-t-e →4 n-e-c-e-ss-i-t
n-e-c-e-ss-i-t-a-t-e-d →1bl n-e-c-e-ss-i-t-a-t →1bll n-e-c-e-ss-i-t-a-t-e →4 n-e-c-e-ss-i-t
n-e-c-e-ss-a-r-y →1c n-e-c-e-ss-a-r-i
str-e-ss-e-s →1a str-e-ss
str-e-ss-e-d →1bl str-e-ss
r-i-gg-i-ng →1bl r-i-gg →1bll r-i-g

Porter - Number of Words Reduced at Each Step
每一步Porter算法减少的字数

Words Reduced Step 1	3597
Words Reduced Step 2	766
Words Reduced Step 3	327
Words Reduced Step 4	2424
Words Reduced Step 5	1373
Words not Reduced	3650

These figures are for an example text.

They indicate how often the different rules are used.

The Stoplist
信息检索停止列表

A stoplist is a list of words which are not used for indexing. 不用于索引的单词

A stoplist may contain:

- Words which are very frequent and therefore likely to occur within all documents being indexed;
- Function words such a determiners (the, a), prepositions (in, on, above), quantifiers (many few), auxiliary verbs (have, am);
- Numbers;
- Punctuation;
- Non-alphanumeric characters.

高频词
功能词
数字
标点
非字母数字字符

The design of the stoplist depends on the application domain.

Keith van Rijsbergen
F测量的发明者

Pioneer of Information Retrieval

Wrote a famous book on IR　他写了一本关于IR的着名书。

Invented Error (E) which combined
P and R
他将P和R结合起来制作F Measure

1-E became F measure!

Book:
http://www.dcs.gla.ac.uk/Keith/Preface.html

van Rijsbergen Stoplist
van Rijsbergen停止列表

A	CANNOT	INTO	OUR	THUS
ABOUT	CO	IS	OURS	TO
ABOVE	COULD	IT	OURSELVES	TOGETHER
ACROSS	DOWN	ITS	OUT	TOO
AFTER	DURING	ITSELF	OVER	TOWARD
AFTERWARDS	EACH	LAST	OWN	TOWARDS
AGAIN	EG	LATTER	PER	UNDER
AGAINST	EITHER	LATTERLY	PERHAPS	UNTIL
ALL	ELSE	LEAST	RATHER	UP
ALMOST	ELSEWHERE	LESS	SAME	UPON
ALONE	ENOUGH	LTD	SEEM	US
ALONG	ETC	MANY	SEEMED	VERY
ALREADY	EVEN	MAY	SEEMING	VIA
ALSO	EVER	ME	SEEMS	WAS
ALTHOUGH	EVERY	MEANWHILE	SEVERAL	WE
ALWAYS	EVERYONE	MIGHT	SHE	WELL

van Rijsbergen Stoplist
van Rijsbergen停止列表

AMONG	EVERYTHING	MORE	SHOULD	WERE
AMONGST	EVERYWHERE	MOREOVER	SINCE	WHAT
AN	EXCEPT	MOST	SO	WHATEVER
AND	FEW	MOSTLY	SOME	WHEN
ANOTHER	FIRST	MUCH	SOMEHOW	WHENCE
ANY	FOR	MUST	SOMEONE	WHENEVER
ANYHOW	FORMER	MY	SOMETHING	WHERE
ANYONE	FORMERLY	MYSELF	SOMETIME	WHEREAFTER
ANYTHING	FROM	NAMELY	SOMETIMES	WHEREAS
ANYWHERE	FURTHER	NEITHER	SOMEWHERE	WHEREBY
ARE	HAD	NEVER	STILL	WHEREIN
AROUND	HAS	NEVERTHELESS	SUCH	WHEREUPON
AS	HAVE	NEXT	THAN	WHEREVER
AT	HE	NO	THAT	WHETHER
BE	HENCE	NOBODY	THE	WHITHER
BECAME	HER	NONE	THEIR	WHICH

van Rijsbergen Stoplist
van Rijsbergen停止列表

BECAUSE	HERE	NOONE	THEM	WHILE
BECOME	HEREAFTER	NOR	THEMSELVES	WHO
BECOMES	HEREBY	NOT	THEN	WHOEVER
BECOMING	HEREIN	NOTHING	THENCE	WHOLE
BEEN	HEREUPON	NOW	THERE	WHOM
BEFORE	HERS	NOWHERE	THEREAFTER	WHOSE
BEFOREHAND	HERSELF	OF	THEREBY	WHY
BEHIND	HIM	OFF	THEREFORE	WILL
BEING	HIMSELF	OFTEN	THEREIN	WITH
BELOW	HIS	ON	THEREUPON	WITHIN
BESIDE	HOW	ONCE	THESE	WITHOUT
BESIDES	HOWEVER	ONE	THEY	WOULD
BETWEEN	I	ONLY	THIS	YET
BEYOND	IE	ONTO	THOSE	YOU
BOTH	IF	OR	THOUGH	YOUR
BUT	IN	OTHER	THROUGH	YOURS

van Rijsbergen Stoplist
van Rijsbergen停止列表

BY	INC	OTHERS	THROUGHOUT	YOURSELF
CAN	INDEED	OTHERWISE	THRU	YOURSELVES

Kučera and Francis

Henry Kučera (1925-2010)
Brown University

W. Nelson Francis (1910-2002)
Brown University

Created the Brown Corpus of Standard American English

Conducted very important computational frequency studies on the corpus

他们创建了“布朗语料库”并对词频进行了第一次计算机分析。

Word Frequency and Stoplists
单词频率和停止列表

No. Words	Examples	% of Words
2	the, of	10
6	the, of, and, to, a, in	20
18		30
250		50

(Kučera and Francis, 1967)

What Does this Show?
词频表显示什么？

Not all words are equally frequent.

A small collection of words (~250) accounts for 50% of a text.

There is thus a frequency distribution of words.

Hans Peter Luhn at IBM was also interested in this.

Hans Peter Luhn (1896-1964)

Worked at IBM TJ Watson Research Centre
Yorktown Heights, NY, USA

A Computer Science researcher:

Luhn Algorithm (Checksum) for
credit card numbers

KWIC (Key Words In Context) indexing

Selective dissemination of information ("SDI")

词频分析的另一个先驱

Luhn's Ideas about Word Frequency / Luhn关于词频的想法

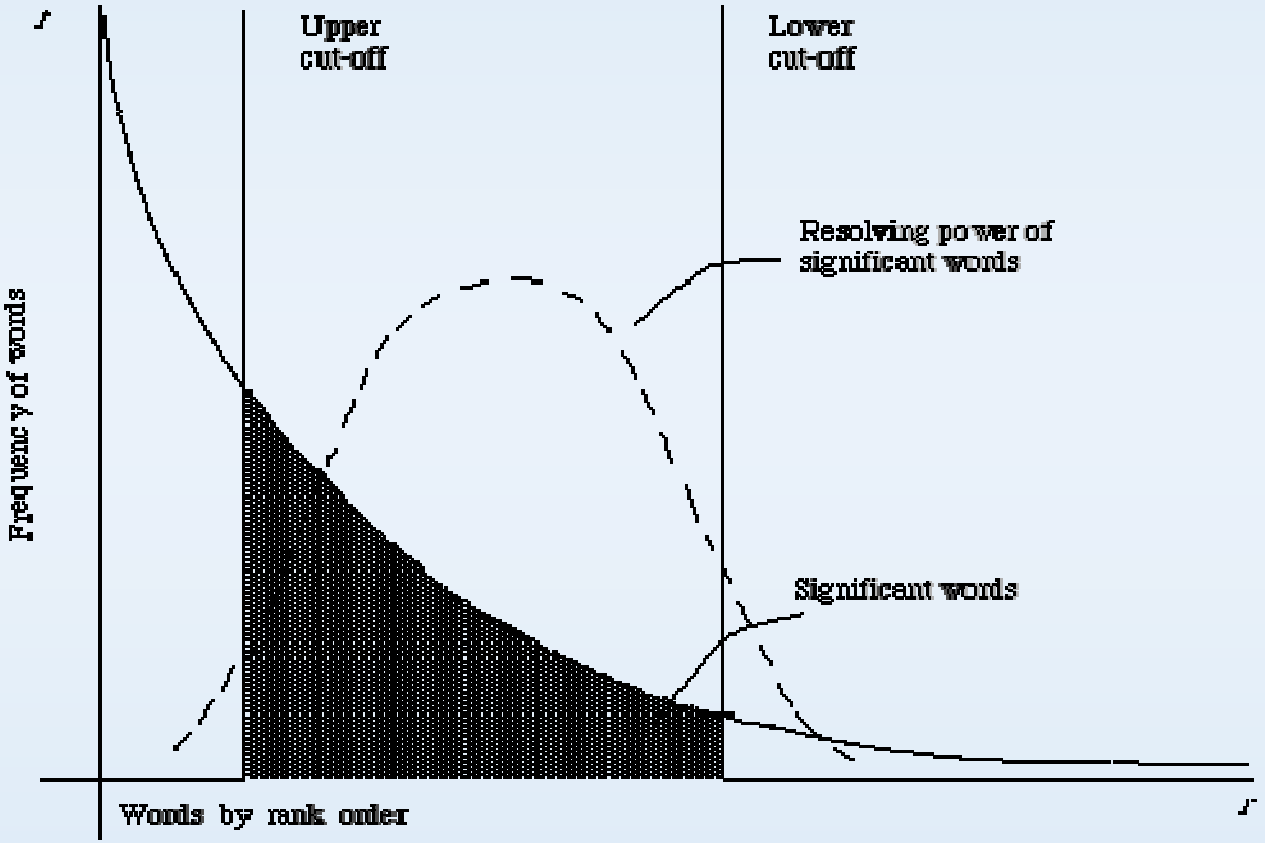


Figure 2.1. A plot of the hyperbolic curve relating f, the frequency of occurrence and r, the rank order (Adapted from Schultz[44] page 120)

Notes on Luhn's Graph
关于Luhn图的注记

Words are arranged on the X-Axis in decreasing order of frequency.

字在X轴上按频率的降序排列。

Frequency of a word is shown on the Y-Axis.

字的频率显示在Y轴上。

So, freq of words falls off asymptotically - rapidly for first few words, then less and less rapidly as we move through the entire vocabulary.

单词的频率首先迅速下降，然后不那么快。

This ties in with Kučera and Francis

- 2 words = 10% text
- 18 words = 30% text etc

这与Kučera和Francis一致

在右边，我们有许多非常罕见的词。

At the asymptote (right-hand side) we have a lot of words which are infrequent.

There are a lot of words in English (and other languages too of course) which you might hear spoken only once every few years!

有些词你几年就会听到一次！

All this has ramifications for text processing and indexing.

所有这些要点对于文本处理都非常重要。

Key Points about Luhn's Graph
关于Luhn图的要点

Shaded area of graph shows ‘interesting’ words - neither too rare nor too frequent.

Such words fall between upper and lower cutoff points.

Dotted bell curve shows (hypothesises) the resolving power of words, especially ‘interesting’ words.

For bell curve, Y-Axis is assumed to be different (Resolving Power not Frequency).

So, resolving power is highest in middle of shaded region between upper and lower cutoff.

‘Resolving Power’ of a word w is assumed to be ‘ability to find a relevant document given w as input’.

图表的阴影区域显示有趣的单词 - 既不太罕见也不太频繁。

这些词落在上下截止点之间。

虚线钟形曲线显示了单词的分辨能力。

对于钟形曲线，假设Y轴是分辨功率而不是频率。

在上下截止点之间的阴影区域中间，分辨率最高。

假设词w的分辨能力是在给定w作为输入的情况下找到相关文档的能力。

George Kingsley Zipf (1902–1950)

Received all his degrees (BS, MS, PhD) from Harvard

Expert on Chinese

Chairman of the German Department

Studied distributions in linguistics and other fields

Popularised Zipf's Law

Zipf's Law

$$P_n \sim 1/n^a$$

where

P_n is the frequency of a word ranked n^{th}

exponent a is almost 1

The most frequent word will occur approximately: twice as often as the second most frequent word, three times as often as the third most frequent word etc.

The frequency of any word is inversely proportional to its rank in the frequency table.

Advantages of Stoplists in IR
IR中的禁用词列表的优点

The use of a fairly small stoplist can considerably reduce the number of terms to be indexed. This follows from Kučera and Francis (1967).

一个相当小的停止列表可以大大减少要编入索引的术语数量。

This can make the inverted index small and can make the IR system run faster.

这可以使倒排索引变小并且可以使IR系统运行得更快。

Words which occur frequently are likely to be present in all documents. Thus they have minimal power to discriminate between documents during retrieval.

经常出现的词语很可能出现在所有文件中。因此，它们在检索期间具有最小的区分文档的能力。

However...

Disadvantages of Stoplists in IR
IR中的停止列表的缺点

What about phrase searches? These are common on the Web:

- Names like “King of Denmark”
- Song titles “Let it be” (Beatles)
- Quotations “To be or not to be” (Hamlet)
- Queries with internal structure “flights to London”

These all need stopwords – and possibly punctuation – to be indexed.

但是，短语搜索在Web上很常见：
名称
歌名
语录
结构化查询

这些都需要索引停用词和标点字符。

Efficiency and Stoplists
效率和停用词列表

Use of a stoplist will save time and space.

However, modern IR techniques can offset these advantages:

- efficient indexing
- use of compression

For these reasons, most modern Web search engines do not use a stoplist.

使用停止列表将节省时间和空间。

但是，现代信息检索技术可以抵消这些优势：

- 高效的索引
- 使用压缩

由于这些原因，大多数现代Web搜索引擎不使用停止列表。

Capitalisation, Fullstops and Hyphens
用大写，句号和连字符

Most systems

- Bring words down to lower case (or at least match lower to upper) 将单词转换为小写

- Eliminate fullstops in abbreviations 从缩写中删除句号

e.g.
USA -> usa
U.S.A. -> USA -> usa

- eliminate hyphens (or at least match terms with hyphens to those without) 删除连字符

e.g.
anti-social -> antisocial

Truecasing - Finding the Proper Capitalisation of Words
“Truecasing” - 将单词转换为正确的大小写

The first word in a sentence starts with a capital: ‘Fishing is a popular hobby.’ The word is ‘fishing’.	句子中的第一个单词
Names are in capitals: 'Richard gave a lecture in Xi'an'. Words are 'Richard', 'Xi'an', not 'richard', 'xi'an'.	恰当的名字
A word can be emphased in an email or tweet: ‘VERY interesting’, ‘I DO NOT agree’. The words are ‘very’, ‘do’ and ‘not’.	在'推文'等中强调了单词
Titles: ‘Teaching Information Retrieval: A New Survey’. The words are ‘teaching’, ‘information’, ‘retrieval’, ‘new’, ‘survey’.	标题
Acronyms: ‘The District Organiser (DO) is…’. The word is ‘DO’ not ‘do’.	缩略语

Size of Term Vocabulary on Web
Web上术语词汇量的大小

Firstly there are words in many different languages.

- Perhaps 500,000 words in English.

Secondly there are acronyms, trademarks, email addresses and proper names e.g. R2-D2, C-3PO, IBM, B-52 and so on.

Thirdly there are typographical errors and mis-spellings.

The total vocabulary for all languages is large, perhaps hundreds of millions i.e. 100,000,000 distinct terms.

For English-only, there could be about 1,000,000 terms including acronyms etc.

Term Weighting Algorithms
术语加权算法

Karen Spärck Jones 1935-2007

Pioneer of Information Retrieval

Developed fundamental principles, along with Gerard Salton, Keith van Rijsbergen and Cyril Cleverdon

Advanced term weighting algorithms

Invented Inverted Document Frequency (IDF)

Term Weighting with TF*IDF
TF * IDF术语加权

As we saw in the introduction, two principles are widely used:

文档中的频率

Term Frequency (within document)

所有文件的总体频率。

Inverted Document Frequency (across documents)

INVERTED意味着我们按文档频率划分一个。

Recall that these are normally multiplied together to give an overall weight for each term.

乘以一起

We call this TF*IDF.

Term Frequency (Within a Document)
期限频率（文件内）

This is a measure based on the frequency of the term within the document.

它基于文档中的术语频率

It needs to be normalised in some way for two reasons:

测量需要标准化，即放在标准值范围内，例如：[0,1]。

- A term occurring twenty times is not twenty times as significant as one occurring once;

发生二十次的术语不是一次发生的术语的二十倍

- Terms are likely to occur more frequently in longer documents than in shorter ones.

在较长的文档中，术语可能比在较短的文档中更频繁地出现。

What Does Normalisation of TF Do?
TF的归一化有什么作用？

1. Causes values to lie on a required range, e.g. 0.0-1.0.

值将位于正确的范围内。

2. Flattens values so that very large raw values of TF result in slightly larger normalised values.

'变平'价值：更大的原始价值变成稍大的标准化价值。

To achieve (1) we can divide the value by the maximum value it has in the entire collection (See Croft (1983) below).

要做（1）我们除以整个集合中的最大值。

To achieve (2) we can take the log of the value (see Harman (1986) below).

要做（2）我们取值的对数。

Formulae for Term Frequency
期限频率公式

K=0.5

1: $cfreq = K + (1 - K)\ (\ freq_{ij} / maxfreq_j\ )$

(Croft 1983)

(cf. Manning & Schütze 6.4.2 Eqn 6.15 - identical)

2: $hfreq = log_2\ (\ freq_{ij} + 1\ ) / log_2(\ length_j\ )$

(Harman 1986)

(cf. Manning & Schütze 6.4.1 Eqn 6.13 - not identical)

where

$freq_{ij}$	=	freq of term i in document j	文件j中的术语i的频率
$maxfreq_j$	=	max freq of any term in document j	文件j中任何术语的最大频率
$length_j$	=	number of unique terms in doc j	文件j中的唯一术语数量
K	=	tuning constant	调整常数

Notes on TF Formulae
期限频率公式

In **1,** $freq_{ij}$ / $maxfreq_j$ varies between 0 and 1.

该值在[0,1]范围内。

Use of *maxfreq* effectively normalises the effect of $freq_{ij}$.

freqij的效果可以使用maxfreq进行标准化。

Tuning constant is between 0 and 1:

调谐常数在[0,1]范围内。

- It controls the influence of *cfreq.*
- As K gets larger, the effect of the coefficient gets smaller.
- Thus, differences in frequency have less effect on the normalised frequency.

它控制着cfreq的影响。
随着K变大，系数的影响变小。
因此，频率差对归一化频率的影响较小。

with a K of zero the effect of differences in frequency is largest.

零K值意味着频率差异影响最大。

Notes on TF formulae cont.
期限频率公式

In **2,** we use the number of terms in the document (word count minus repetition) rather than the max frequency of a term.

我们使用文档中的术语数，而不是最大频率。

Recall, the longer the document, the higher the likely freq of any particular term.

文档越长，可能存在的术语频率越高。

We also 'flatten' the effect using the log:

我们用日志功能展平

$\log_2 1 = 0$
$\log_2 10 = 3.3$
$\log_2 100 = 6.6$
$\log_2 1000 = 10.0$

As you see, $\log_2$ x rises quite slowly as x rises rapidly.

当x迅速上升时，log2x缓慢上升。

$\log_2$ is commonly used for this flattening effect in IR functions.

log2x通常用作展平函数。

Notes on TF formulae cont.
期限频率公式

In 2, if $freq_{ij}$ is zero, log(0) could be computed, resulting in $-\infty$.

如果freqij为零，则可以计算log（0），从而得到-∞。

To avoid this, we add 1 on the numerator.

为避免这种情况，我们在分子上加1。

The denominator is a measure of document length.

分母是文档长度的度量。

Inverted Document Frequency
逆文档频率

1 $IDF_i = log_2 (N / n_i + 1)$

(Spärck Jones 1972)

(cf. Manning & Schütze 6.2.1 Eqn 6.7 - not identical)

2 $IDF_i = log_2 (maxn / n_i + 1)$

(Spärck Jones 1979)

where

N	= number of documents in collection	收集中的文件数量
n_i	= total occurrences of term *i* in collection	集合中术语i的总出现次数
maxn	= max frequency of any term in collection	收集中任何术语的最大频率

Inverted Document Frequency cont.
逆文档频率

1 $IDF_i = log_2 (N / n_i + 1)$

2 $IDF_i = log_2 (maxn / n_i + 1)$

log <1	=	small negative	小负面
log 1	=	0	零
log large	=	small positive	小积极的

1: If N >> n_i, IDF is small positive integer 小正整数
if N << n_i, IDF is near zero 接近于零

2: if *maxn* >> n_i, IDF is small positive 小积极的
if *maxn* << n_i, IDF is near zero 接近于零

Inverted Document Frequency Key Point
关于逆文档频率的重点

Key point:

If a term is rare in the document collection, IDF is large.

如果特定单词在整个文档集合中的频率较低，则IDF很大。

If a term is common in the document collection, IDF is small.

如果文档集中的单词频率很高，则IDF很小。

A rare term allows us to distinguish between documents.

一个罕见的术语允许我们区分文件。

A common term does not.

一个常见的术语不是。

Combining TF with IDF
将TF与IDF结合起来

A term weight can be obtained by multiplying TF ($cfreq_{ij}$) by the inverted document frequency:

可以通过将TF（cfreqij）乘以反转文档频率来获得术语权重：

$$w_{ij} = cfreq_{ij} * IDF_i$$

with a suggested K in $cfreq_{ij}$ of 0.5

(Salton and Buckley 1988)

freqij中K的建议值为0.5。

TF*IDF is fundamental to modern search.

TF * IDF是现代搜索的基础。

However, other factors are important too and we will discuss them later.

但是，其他因素也很重要，我们稍后会讨论。