Search Engines--Information Retrieval in Practice

tanshuqiu

Email: tsq@cqut.edu.cn

1

Processing Text

First : From Words to Terms

Second : Text Statistics

Third : Document Parsing

Fourth : Document Structure and Markup

2

Processing Text

Fifth : Link Analysis

Sixth : Information Extraction

3

From Words to Terms

After gathering the text we want to search, the next step is to decide whether it should be modified or restructured in some way to simplify searching. The types of changes that are made at this stage are called text transformation or, more often, text processing.

The goal of text processing is to convert the many forms in which words can occur into more consistent index terms. Index terms are the representation of the content of a document that are used for searching.

Text Statistics

One of the most obvious features of text from a statistical point of view is that the distribution of word frequencies is very skewed. There are a few words that have very high frequencies and many words that have low frequencies.

Text Statistics

In fact, the two most frequent words in English (“the” and “of ”) account for about 10% of all word occurrences. The most frequent six words account for 20% of occurrences, and the most frequent 50 words are about 40% of all text! On the other hand, given a large sample of text, typically about one half of all the unique words in that sample occur only once.

Text Statistics

This distribution is described by Zipf ’s law, which states that the frequency of the rth most common word is inversely proportional to r or, alternatively, the rank of a word times its frequency (f) is approximately a constant (k):

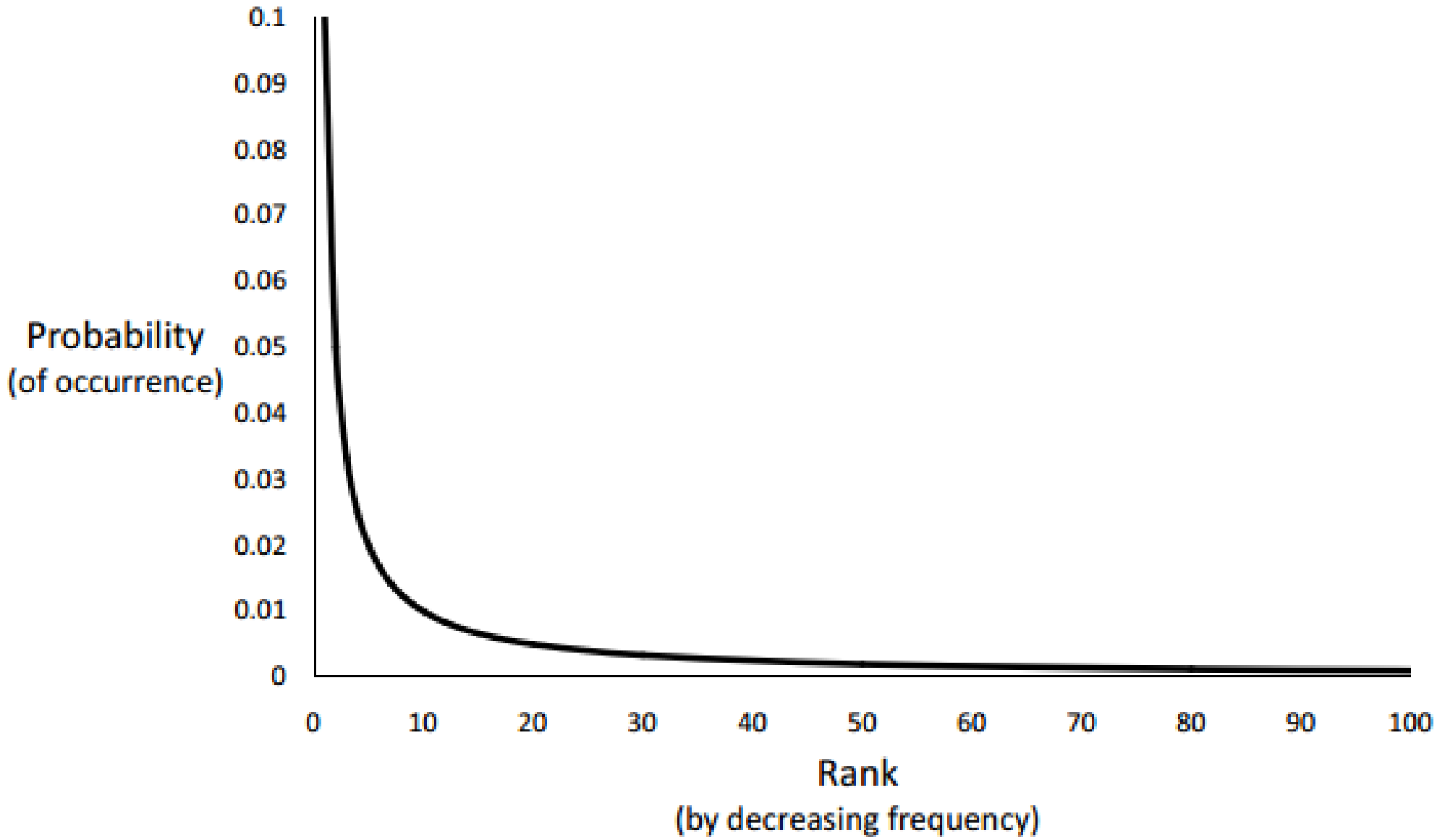
$$r \cdot f = k$$

Text Statistics

We often want to talk about the probability of occurrence of a word, which is just the frequency of the word divided by the total number of word occurrences in the text. In this case, Zipf ’s law is:

$$r \cdot P_r = c$$

where Pr is the probability of occurrence for the rth ranked word, and c is a constant.


0.1
0.09
0.08
0.07
0.06
0.05
0.04
0.03
0.02
0.01
0
Probability
(of occurrence)
0
10
20
30
40
50
60
70
80
90
100
Rank
(by decreasing frequency)

Total documents	84,678
Total word occurrences	39,749,179
Vocabulary size	198,763
Words occurring > 1000 times	4,169
Words occurring once	70,064

Word	*Freq.*	*r*	P_r(%)	$r.P_r$	*Word*	*Freq*	*r*	P_r(%)	$r.P_r$
the	2,420,778	1	6.49	0.065	has	136,007	26	0.37	0.095
of	1,045,733	2	2.80	0.056	are	130,322	27	0.35	0.094
to	968,882	3	2.60	0.078	not	127,493	28	0.34	0.096
a	892,429	4	2.39	0.096	who	116,364	29	0.31	0.090
and	865,644	5	2.32	0.120	they	111,024	30	0.30	0.089
in	847,825	6	2.27	0.140	its	111,021	31	0.30	0.092
said	504,593	7	1.35	0.095	had	103,943	32	0.28	0.089
for	363,865	8	0.98	0.078	will	102,949	33	0.28	0.091
that	347,072	9	0.93	0.084	would	99,503	34	0.27	0.091
was	293,027	10	0.79	0.079	about	92,983	35	0.25	0.087
on	291,947	11	0.78	0.086	i	92,005	36	0.25	0.089
he	250,919	12	0.67	0.081	been	88,786	37	0.24	0.088
is	245,843	13	0.65	0.086	this	87,286	38	0.23	0.089
with	223,846	14	0.60	0.084	their	84,638	39	0.23	0.089
at	210,064	15	0.56	0.085	new	83,449	40	0.22	0.090
by	209,586	16	0.56	0.090	or	81,796	41	0.22	0.090
it	195,621	17	0.52	0.089	which	80,385	42	0.22	0.091
from	189,451	18	0.51	0.091	we	80,245	43	0.22	0.093
as	181,714	19	0.49	0.093	more	76,388	44	0.21	0.090
be	157,300	20	0.42	0.084	after	75,165	45	0.20	0.091
were	153,913	21	0.41	0.087	us	72,045	46	0.19	0.089
an	152,576	22	0.41	0.090	percent	71,956	47	0.19	0.091
have	149,749	23	0.40	0.092	up	71,082	48	0.19	0.092
his	142,285	24	0.38	0.092	one	70,266	49	0.19	0.092
but	140,880	25	0.38	0.094	people	68,988	50	0.19	0.093

Word	*Freq.*	r	P_r(%)	$r.P_r$
assistant	5,095	1,021	.013	0.13
sewers	100	17,110	.000256	0.04
toothbrush	10	51,555	.000025	0.01
hazmat	1	166,945	.000002	0.04

Text Statistics

Another useful prediction related to word occurrence is vocabulary growth. As the size of the corpus grows, new words occur. Based on the assumption of a Zipf distribution for words, we would expect that the number of new words that occur in a given amount of new text would decrease as the size of the corpus increases.

Text Statistics

The relationship between the size of the corpus and the size of the vocabulary was found empirically by Heaps (1978) to be:

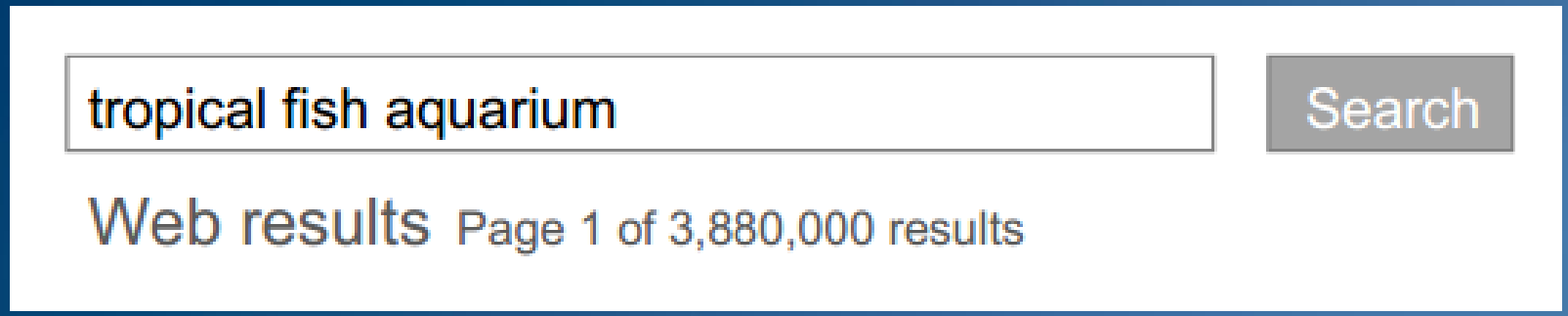
$$v = k \cdot n^{\beta}$$

where v is the vocabulary size for a corpus of size n words, and k and β are parameters that vary for each collection.

Text Statistics

Estimating Collection and Result Set Sizes

Word occurrence statistics can also be used to estimate the size of the results from a web search. All web search engines have some version of the query interface shown in the following figure, where immediately after the query and before the ranked list of results, an estimate of the total number of results is given. This is typically a very large number, and descriptions of these systems always point out that it is just an estimate. Nevertheless, it is always included.



Result size estimate for web search

Text Statistics

Estimating Collection and Result Set Sizes

If we assume that words occur independently of each other, then the probability of a document containing all the words in the query is simply the product of the probabilities of the individual words occurring in a document.

Text Statistics

Estimating Collection and Result Set Sizes

For example, if there are three query words a, b, and c, then:

$$P(a \cap b \cap c) = P(a) \cdot P(b) \cdot P(c)$$

where P (a ∩ b ∩ c) is the joint probability, or the probability that all three words occur in a document, and P (a), P (b), and P (c) are the probabilities of each word occurring in a document.

Text Statistics

Estimating Collection and Result Set Sizes

A search engine will always have access to the number of documents that a word occurs in (f_a, f_b, and f_c), and the number of documents in the collection (N), so these probabilities can easily be estimated as P (a) =f_a/N, P (b) = f_b/N, and P (c) = f_c/N. This gives us

$$f_{abc} = N \cdot f_a/N \cdot f_b/N \cdot f_c/N = (f_a \cdot f_b \cdot f_c)/N^2$$

where f_{abc} is the estimated size of the result set.

Word(s)	*Document Frequency*	*Estimated Frequency*
tropical	120,990	
fish	1,131,855	
aquarium	26,480	
breeding	81,885	
tropical fish	18,472	5,433
tropical aquarium	1,921	127
tropical breeding	5,510	393
fish aquarium	9,722	1,189
fish breeding	36,427	3,677
aquarium breeding	1,848	86
tropical fish aquarium	1,529	6
tropical fish breeding	3,629	18

Text Statistics

Estimating Collection and Result Set Sizes

Better estimates are possible if word co-occurrence information is also available from the search engine. Obviously, this would give exact answers for two-word queries. For longer queries, we can improve the estimate by not assuming independence. In general, for three words

$$P(a \cap b \cap c) = P(a \cap b) \cdot P(c|(a \cap b))$$

Text Statistics

Estimating Collection and Result Set Sizes

$$P(a \cap b \cap c) = P(a \cap b) \cdot P(c|(a \cap b))$$

where P (a ∩b) is the probability that the words a and b co-occur in a document, and P (c|(a∩b)) is the probability that the word co-occurs in a document given that the words a and b occur in the document. If we have co-occurrence information, we can approximate this probability using either P (c|a) or P (c|b) , whichever is the largest.

Text Statistics

Estimating Collection and Result Set Sizes

For example query “tropical fish aquarium” in Table, this means we estimate the result set size by multiplying the number of documents containing both “tropical” and “aquarium” by the probability that a document contains “fish” given that it contains “aquarium”, or:

$$f_{tropical \cap fish \cap aquarium} = f_{tropical \cap aquarium} \cdot f_{fish \cap aquarium} / f_{aquarium}$$
$$= 1921 \cdot 9722/26480 = 705$$

Text Statistics

Estimating Collection and Result Set Sizes

Similarly, for the query "tropical fish breeding":

$$f_{tropical \cap fish \cap breeding} = f_{tropical \cap breeding} \cdot f_{fish \cap breeeding} / f_{breeding}$$
$$= 5510 \cdot 36427/81885 = 2451$$

Document Parsing

Tokenizing

Document parsing involves the recognition of the content and structure of text documents.

Tokenizing is the process of forming words from the sequence of characters in a document.

Document Parsing

Tokenizing

For example, that the text:

Bigcorp's 2007 bi-annual report showed profits rose 10%.

would produce the following sequence of tokens:

bigcorp 2007 annual report showed profits rose

Document Parsing

Tokenizing

Although this simple tokenizing process was adequate for experiments with small test collections, it does not seem appropriate for most search applications, because too much information is discarded.

Document Parsing

Tokenizing

Some examples of issues involving tokenizing that can have significant impact on the effectiveness of search are:

Small words (one or two characters) can be important in some queries, usually in combinations with other words. For example, xp, ma, pm.

Both hyphenated and non-hyphenated forms of many words are common. In some cases the hyphen is not needed. At other times, hyphens should be considered either as part of the word or a word separator

Document Parsing

Tokenizing

Special characters are an important part of the tags, URLs, code, and other important parts of documents that must be correctly tokenized.

Capitalized words can have different meaning from lowercase words. For example, “Bush” and “Apple”.

Document Parsing

Tokenizing

The fact that these examples come from queries also emphasizes that the text processing for queries must be the same as that used for documents. If different tokenizing processes are used for queries and documents, many of the index terms used for documents will simply not match the corresponding terms from queries. Mistakes in tokenizing become obvious very quickly through retrieval failures.

Document Parsing

Stopping

The most popular— “the,” “a,” “an,” “that,” and “those” —are determiners. Prepositions, such as “over,” “under,” “above,” and “below,” represent relative position between two nouns

Document Parsing

Stopping

Two properties of these function words cause us to want to treat them in a special way in text processing.

First, these function words are extremely common.

Second, both because of their commonness and their function, these words rarely indicate anything about document relevance on their own.

In information retrieval, these function words have a second name: **stopwords**.

Document Parsing

Stopping

We call them stopwords because text processing stops when one is seen, and they are thrown out. Throwing out these words decreases index size, increases retrieval efficiency, and generally improves retrieval effectiveness.

Constructing a stopword list must be done with caution. Removing too many words will hurt retrieval effectiveness in particularly frustrating ways for the user.

Document Parsing

Stopping

If storage space requirements allow, it is best to at least index all words in the documents. If stopping is required, the stopwords can always be removed from queries. By keeping the stopwords in the index, there will be a number of possible ways to execute a query with stopwords in it.

Document Parsing

Stemming

Stemming, also called conflation, is a component of text processing that captures the relationships between different variations of a word. More precisely, stemming reduces the different forms of a word that occur because of inflection or derivation to a common stem.

Document Parsing

Stemming

There are two basic types of stemmers: algorithmic and dictionary-based. An algorithmic stemmer uses a small program to decide whether two words are related, usually based on knowledge of word suffixes for a particular language. By contrast, a dictionary-based stemmer has no logic of its own, but instead relies on pre-created dictionaries of related terms to store term relationships.

Document Parsing

Phrases and N-grams

Phrases are clearly important in information retrieval. Many of the two- and three-word queries submitted to search engines are phrases, and finding documents that contain those phrases will be part of any effective ranking algorithm.

Document Parsing

Phrases and N-grams

The definition that has been used most frequently in information retrieval research is that a phrase is equivalent to a simple noun phrase. This is often restricted even further to include just sequences of nouns, or adjectives followed by nouns. Phrases defined by these criteria can be identified using a part-of-speech (POS) tagger.

Document Structure and Markup

In database applications, the fields or attributes of database records are a critical part of searching. Queries are specified in terms of the required values of these fields. In some text applications, such as email or literature search, fields such as author and date will have similar importance and will be part of the query specification.

Document Structure and Markup

Some parts of the structure of web pages, indicated by HTML markup, are very significant features used by the ranking algorithm. The document parser must recognize this structure and make it available for indexing.

Link Analysis

Links connecting pages are a key component of the Web. Links are a powerful navigational aid for people browsing the Web, but they also help search engines understand the relationships between the pages. These detected relationships help search engines rank web pages more effectively.

Link Analysis

A link in a web page is encoded in HTML with a statement such as:

```
For more information on this topic, please go to <a
href="http://www.somewhere.com">the somewhere page</a>.
```

In this link, "the somewhere page" is called the anchor text, and http://www.somewhere.com is the destination. Both components are useful in the ranking process.

Link Analysis

Anchor Text

Anchor text has two properties that make it particularly useful for ranking web pages. First, it tends to be very short, perhaps two or three words, and those words often succinctly describe the topic of the linked page.

Link Analysis

Anchor Text

Anchor text is usually written by people who are not the authors of the destination page. This means that the anchor text can describe a destination page from a different perspective, or emphasize the most important aspect of the page from a community viewpoint. The fact that the link exists at all is a vote of importance for the destination page.

Link Analysis

PageRank

Every web page on the Internet has a PageRank, and it is uniquely determined by the link structure of web pages. As this example shows, PageRank has the ability to distinguish between popular pages (those with many incoming links, or those that have links from popular pages) and unpopular ones.

The PageRank value can help search engines if through the millions of pages that contain the word "eBay" to find the one that is most popular (www.ebay.com).

Information Extraction

Information extraction is a language technology that focuses on extracting structure from text. For search applications, the primary use of information extraction is to identify features that can be used by the search engine to improve ranking.

Information Extraction

Most of the recent research in information extraction has been concerned with features that have specific semantic content, such as named entities, relationships, and events. Although all of these features contain important information, named entity recognition has been used most often in search applications.

Information Extraction

A named entity is a word or sequence of words that is used to refer to something of interest in a particular application. The most common examples are people’ s names, company or organization names, locations, time and date expressions, quantities, and monetary values.

Information Extraction

Two main approaches have been used to build named entity recognizers: rule based and statistical.

A rule-based recognizer uses one or more lexicons (lists of words and phrases) that categorize names. Some example categories would be locations (e.g., towns, cities,), people’ s names, and organizations If these lists are sufficiently comprehensive, much of the extraction can be done simply by lookup.

Information Extraction

A statistical entity recognizer uses a probabilistic model of the words in and around an entity. A number of different approaches have been used to build these models