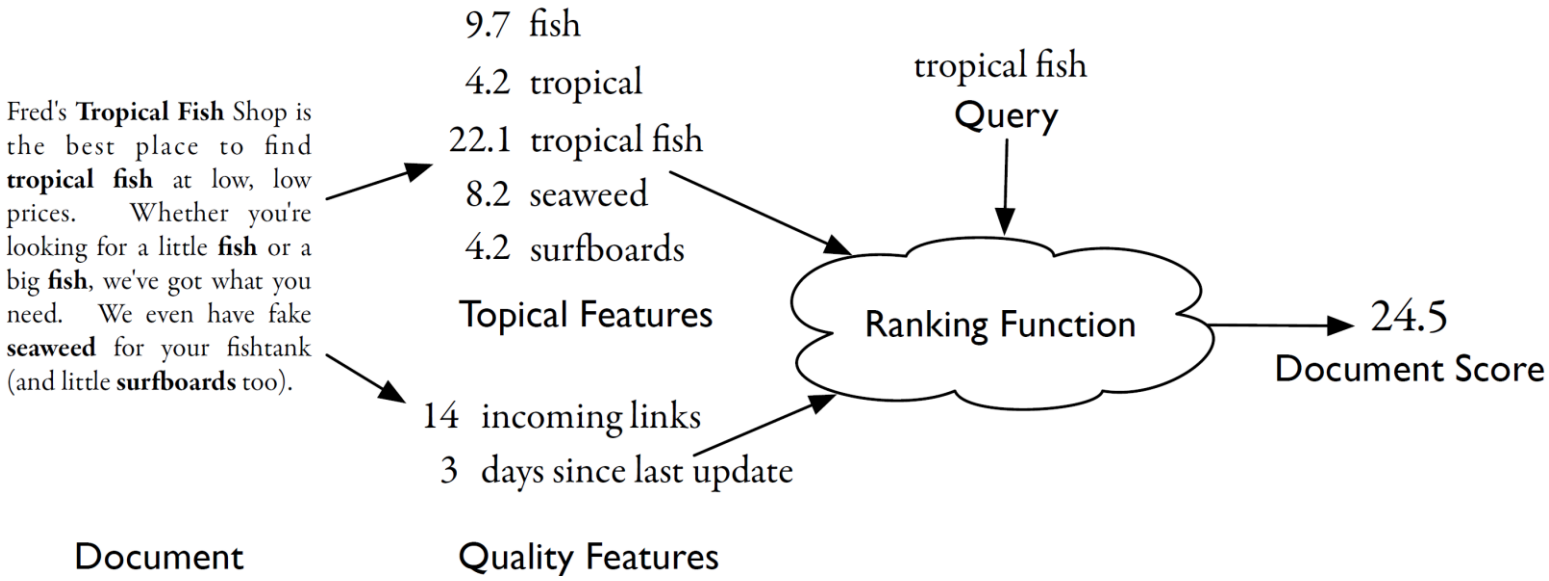
Web Services and Web Data

XJCO3011

Session 12 – Indexing

Abstract Model of Ranking

- Documents are written in natural human languages, which are difficult for computers to analyse directly. So, the text is transformed into *index terms* or *document features*.
- A document feature is some attribute of the document we can express numerically. In the figure, we show two kinds of features:
  - Topical features estimate the degree to which the document is about a particular subject.
  - Document quality features, e.g. the number of web pages that link to this document, or the number of days since this page was last updated.
- Quality features don't address whether the document is a good topical match for a query, but address its quality; a page with no incoming links that hasn't been edited in years is probably a poor match for any query

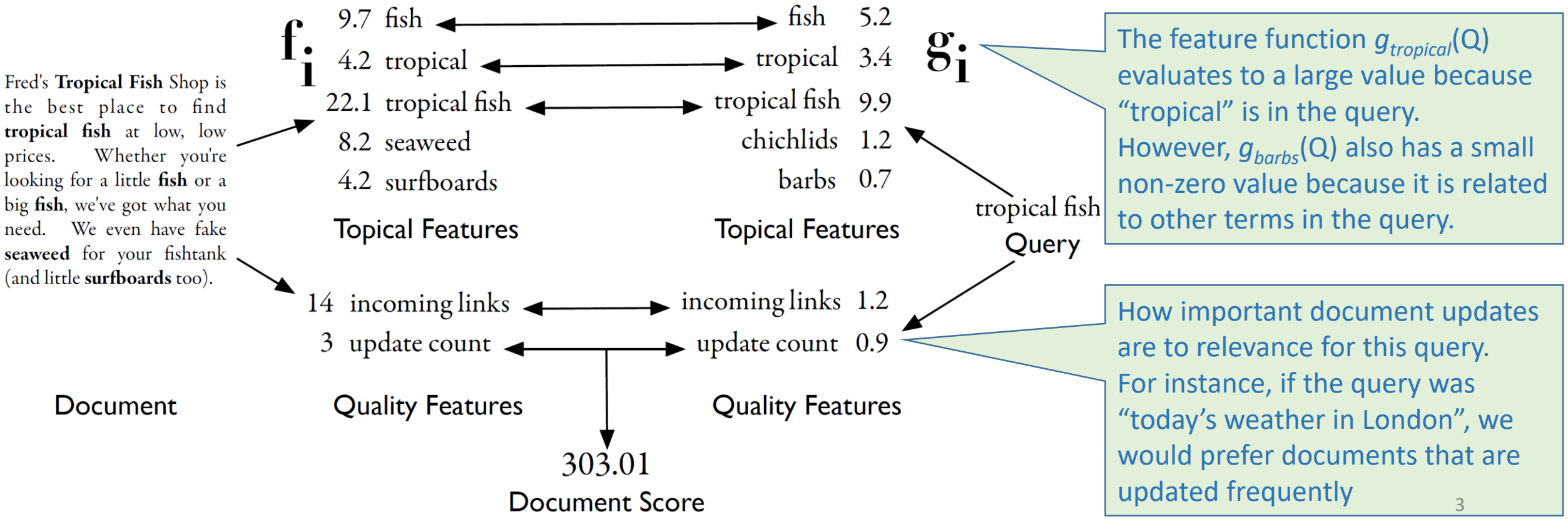


A more concrete ranking model

- We assume that the ranking function R takes the following form:

$$R(Q, D) = \sum_i g_i(Q) f_i(D)$$

- Here, f_i is some feature function that extracts a number from the document text, and g_i is a similar feature function that extracts a value from the query



The feature function $g_{tropical}$(Q) evaluates to a large value because "tropical" is in the query. However, g_{barbs}(Q) also has a small non-zero value because it is related to other terms in the query.

How important document updates are to relevance for this query. For instance, if the query was "today's weather in London", we would prefer documents that are updated frequently

Inverted Indices

- All modern search engine indices are based on *inverted indices*
- An inverted index is the computational equivalent of the index found in the back of all textbooks.
- The index is inverted because usually we think of words being part of documents, but if we invert this idea, documents are associated with words.

Index

Book Index

Index structure

- Index terms are often alphabetized like a traditional book index, but they need not be, since they are often found directly using a hash table.
- Each index term has its own inverted list that holds the relevant data for that term. For a book index, the relevant data is a list of page numbers. In a search engine, the data might be a list of documents.
- Each list entry is called a *posting*, and the part of the posting that refers to a specific document or location is often called a pointer.
- Each document in the collection is given a unique number to make it efficient for storing document pointers.
- Indices in books store more than just location information. For important words, often one of the page numbers is marked in boldface, indicating that this page contains a definition or extended discussion about the term.
- Inverted files can also have extended information, where postings can contain a range of information other than just locations. By storing the right information along with each posting, the feature functions can be computed efficiently.
- In the next few slides, we will look at some different kinds of inverted files.

Documents

- The simplest form of an inverted list stores just the documents that contain each word, and no additional information
- Notice that this index does not record the number of times each word appears; it only records the documents in which each word appears.
- Inverted lists become more interesting when we consider their intersection; for example find the sentence that contains the words "coloration" and "freshwater".

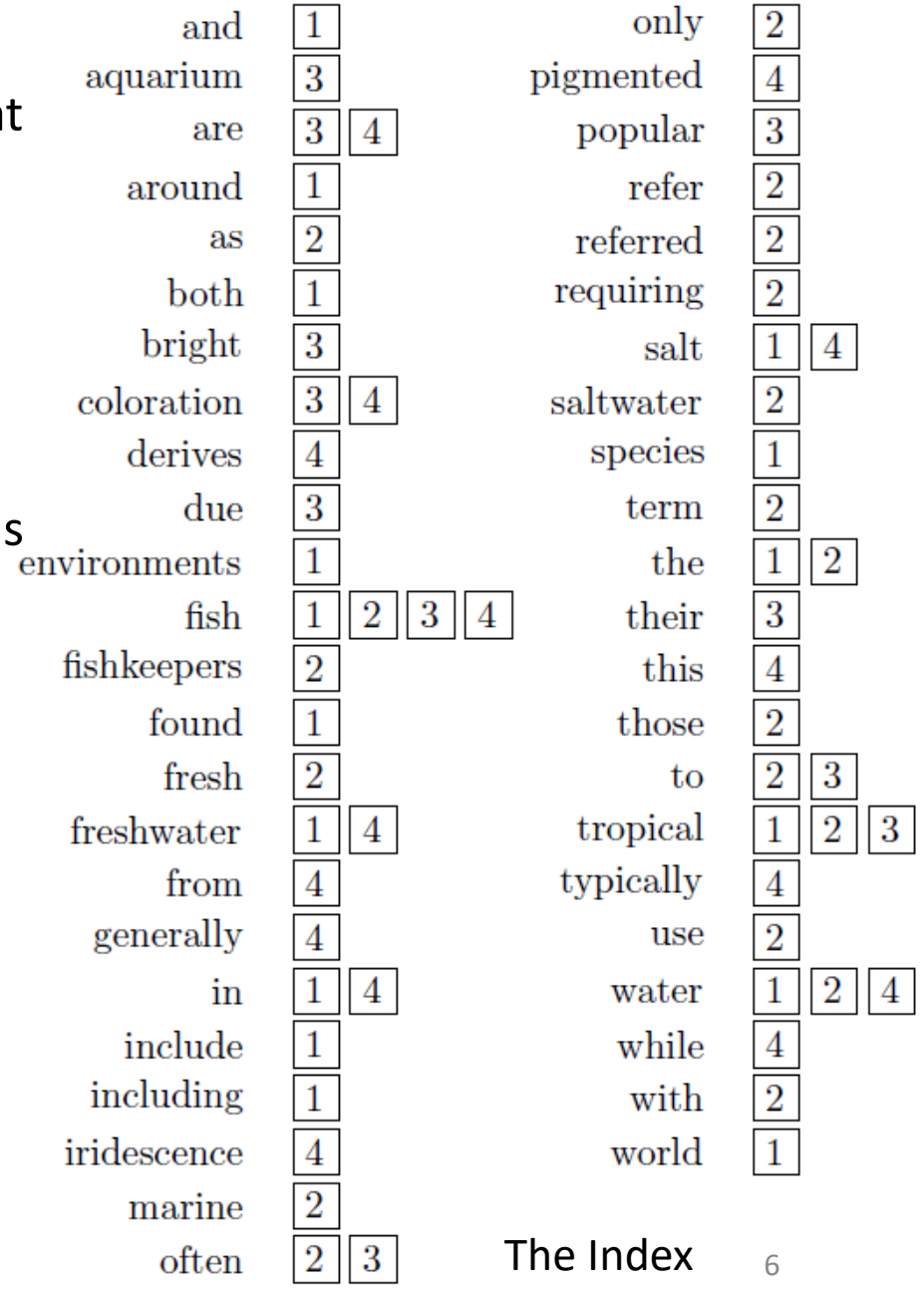
S_1 Tropical fish include fish found in tropical environments around the world, including both freshwater and salt water species.

S_2 Fishkeepers often use the term tropical fish to refer only those requiring fresh water, with saltwater tropical fish referred to as marine fish.

S_3 Tropical fish are popular aquarium fish, due to their often bright coloration.

S_4 In freshwater fish, this coloration typically derives from iridescence, while salt water fish are generally pigmented.

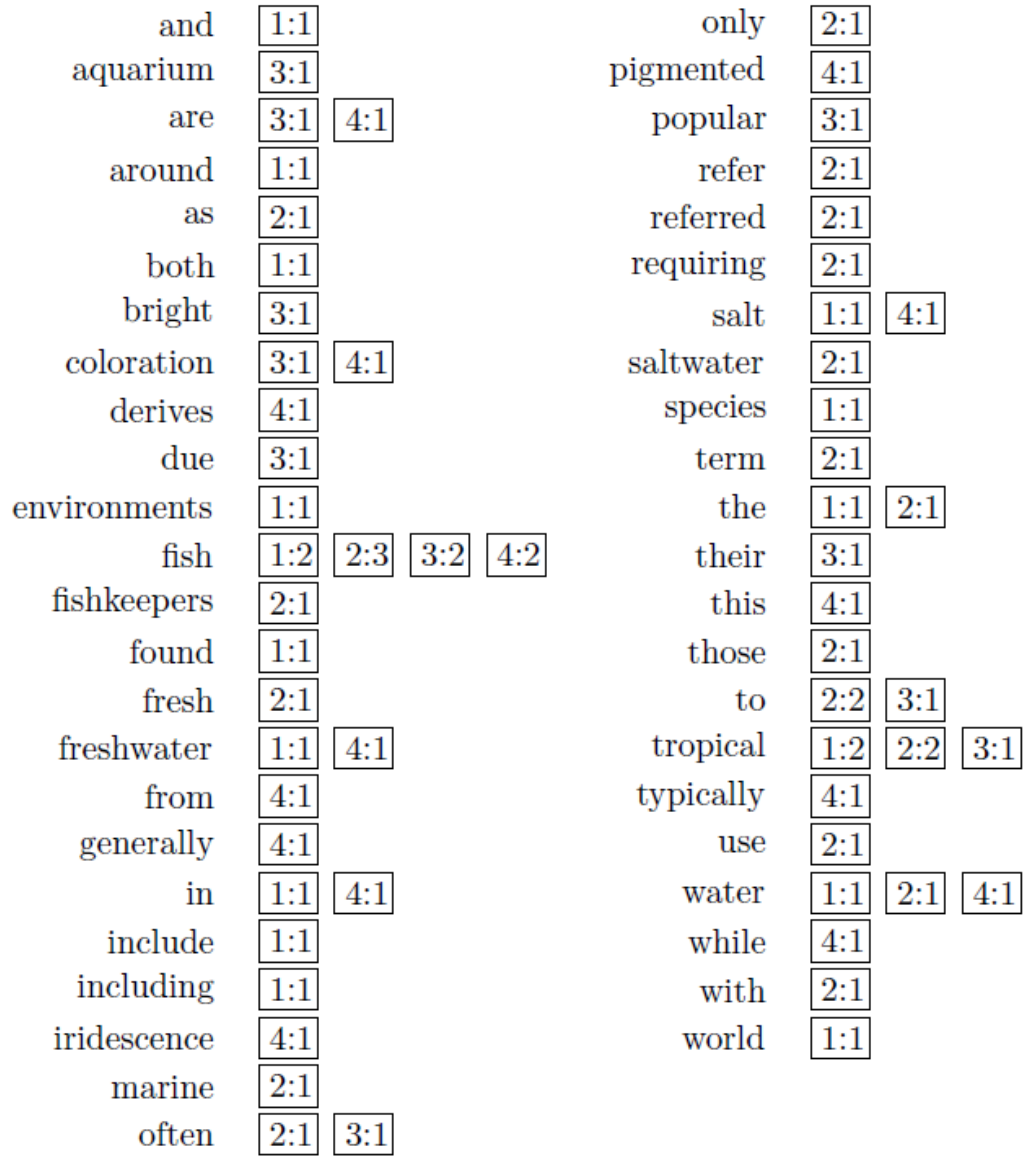
The "documents"



The Index

Counts

- In this method, each posting now has a second number which is the number of times the word appears in the document
- Example: find the most relevant document for the query “tropical fish”
- We prefer S2 over S1 and S3 for the query “tropical fish”, since S2 contains “tropical” twice and “fish” three times.
- In general, word counts can be a powerful predictor of document relevance. In particular, word counts can help distinguish documents that are about a particular subject from those that discuss that subject in passing



An inverted index, with word counts

Positions

and	1,15																		
aquarium	3,5																		
are	3,3	4,14																	
around	1,9																		
as	2,21																		
both	1,13																		
bright	3,11																		
coloration	3,12	4,5																	
derives	4,7																		
due	3,7																		
environments	1,8																		
fish	1,2	1,4	2,7	2,18	2,23														
			3,2	3,6	4,3														
			4,13																
fishkeepers	2,1																		
found	1,5																		
fresh	2,13																		
freshwater	1,14	4,2																	
from	4,8																		
generally	4,15																		
in	1,6	4,1																	
include	1,3																		
including	1,12																		
iridescence	4,9																		

marine	2,22																		
often	2,2	3,10																	
only	2,10																		
pigmented	4,16																		
popular	3,4																		
refer	2,9																		
referred	2,19																		
requiring	2,12																		
salt	1,16	4,11																	
saltwater	2,16																		
species	1,18																		
term	2,5																		
the	1,10	2,4																	
their	3,9																		
this	4,4																		
those	2,11																		
to	2,8	2,20	3,8																
tropical	1,1	1,7	2,6	2,17	3,1														
typically	4,6																		
use	2,3																		
water	1,17	2,14	4,12																
while	4,10																		
with	2,15																		
world	1,11																		

An inverted index, with word positions,

tropical	1,1		1,7	2,6	2,17		3,1												
fish	1,2	1,4		2,7	2,18	2,23	3,2	3,6	4,3	4,13									

Aligning posting lists for "tropical" and "fish" to find the phrase "tropical fish"

- When looking for matches for a query like "tropical fish", the location of the words in the document is an important predictor of relevance
- Imagine a document about food that included a section on tropical fruits followed by a section on saltwater fish. So far, none of the indices we have considered contain enough information to tell us that this document is not relevant
- We want to know if the document contains the exact phrase "tropical fish".
- To determine this, we can add position information to our index
- Each posting contains two numbers: a document number first, followed by a word position
- By aligning the posting lists, we can find the best match for a phrase. The phrase matches are easy to see in the figure; they happen at the points where the postings are lined up in columns.

Fields and Extents

- Real documents are not just lists of words. They have sentences and paragraphs that separate concepts into logical units
- Some documents have titles and headings that provide short summaries of the rest of the content.
- This is where extent lists come in. An extent is a contiguous region of a document. We can represent these extents using word positions
- For example, if the title of a book started on the fifth word and ended just before the ninth word, we could encode that as (5,9).
- E.g., aligning posting lists for “fish” and title to find matches of the word “fish” in the title field of a document

fish	1,2	1,4		2,7	2,18	2,23	3,2	3,6	4,3	4,13
title	1:(1,3)		2:(1,5)							4:(9,15)

Scores

- If the inverted lists are going to be used to generate feature function values, why not just store the value of the feature function? This approach makes it possible to store feature function values that would be too computationally intensive to compute during the query processing phase
- We could make a list for “fish” that has postings like [(1:3.6), (3:2.2)], meaning that the total feature value for “fish” in document 1 is 3.6, and in document 3 it is 2.2
- The number 3.6 came from taking into account how many times “fish” appeared in the title, in the headings, in large fonts, in bold, and in links to the document. Maybe the document doesn’t contain the word “fish” at all, but instead many names of fish, such as “carp” or “trout”. The value 3.6 is then some indicator of how much this document is about fish.
- Storing scores like this both increases and decreases the system’s flexibility. It increases flexibility because computationally expensive scoring becomes possible, since much of the hard work of scoring documents is moved into the index.
- However, flexibility is lost, since we can no longer change the scoring mechanism once the index is built.
- More importantly, information about word proximity is gone in this model, meaning that we can’t include phrase information in scoring unless we build inverted lists for phrases, too.
- These precomputed phrase lists require considerable additional space.

Ordering

- So far, we have assumed that the postings of each inverted list would be ordered by document number.
- Although this is the most popular option, this is not the only way to order an inverted list.
- An inverted list can also be ordered by score, so that the highest-scoring documents come first.
- This makes sense only when the lists already store the score, or when only one kind of score is likely to be computed from the inverted list.
- By storing scores instead of documents, the query processing engine can focus only on the top part of each inverted list, where the highest-scoring documents are recorded

Auxiliary Structures

- An inverted file is just a collection of inverted lists.
- To search the index, some kind of data structure is necessary to find the inverted list for a particular term.
- The simplest way to solve this problem is to store each inverted list as a separate file, where each file is named after the corresponding search term. To find the inverted list for “dog,” the system can simply open the file named dog and read the contents.
- However, document collections can have millions of unique words, and most of these words will occur only once or twice in the collection. This means that an index, if stored in files, would consist of millions of files, most of which are very small
- Unfortunately, modern file systems are not optimized for this kind of storage.
- A file system typically will reserve a few kilobytes of space for each file, even though most files will contain just a few bytes of data. The result is a huge amount of wasted space.
- Imagine for example, a collection of 70,000 words which only occur once. These inverted lists would require about 20 bytes each, for a total of about 2MB of space. However, if the file system requires 1KB for each file, the result is 70MB of space used to store 2MB of data.
- In addition, many file systems still store directory information in unsorted arrays, meaning that file lookups can be very slow for large file directories.

Inverted Files

- To fix these problems, inverted lists are usually stored together in a single file, which explains the name inverted file.
- An additional directory structure, called the vocabulary or lexicon, contains a lookup table from index terms to the byte offset of the inverted list in the inverted file.
- In many cases, this vocabulary lookup table will be small enough to fit into memory. In this case, the vocabulary data can be stored in any reasonable way on disk and loaded into a hash table at search engine startup.
- If the search engine needs to handle larger vocabularies, some kind of tree-based data structure should be used to minimize disk accesses during the search process.

That's it Folks

Further Reading

Chapter 5: Search Engines Information Retrieval in Practice by W. Bruce Croft, Donald Metzler, and Trevor Strohman