Relevance

Task Statement:

Build a system that retrieves documents that users are likely to find **relevant** to their queries.

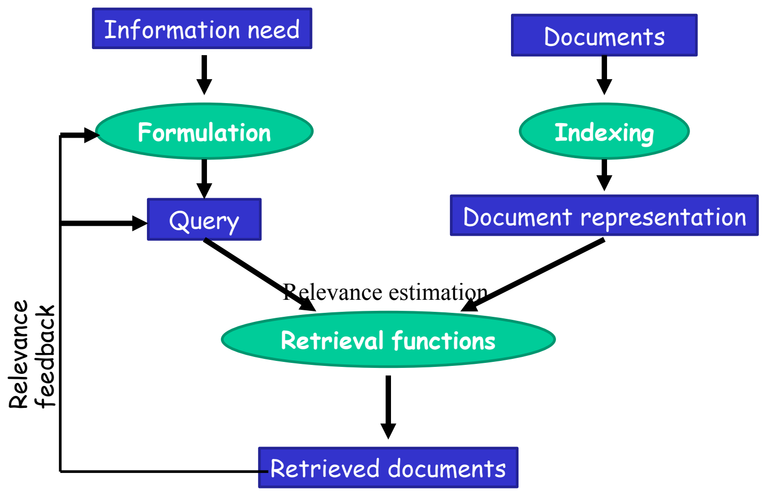
- Relevance
  - What is it?
  - Simple (and simplistic) definition: A relevant document contains the information that a person was looking for when they submitted a query to the search engine
  - Many factors influence a person's decision about what is relevant: e.g. task, context, novelty
  - **Topical relevance** (same topic) vs. **user relevance** (everything else)
  - Related to type of query

Relevance in Practice

- Retrieval models define a *view of relevance*
  - Ranking algorithms used in search engines are based on retrieval models that aim to estimate relevance
  - Typically, these models use statistical properties of text rather than linguistic (e.g. counting simple text features such as words instead of parsing and analysing sentences)
  - Most well-known/classical retrieval models focus on topical relevance
  - User relevance requires more features, different types of evidence

See IR Models lectures later

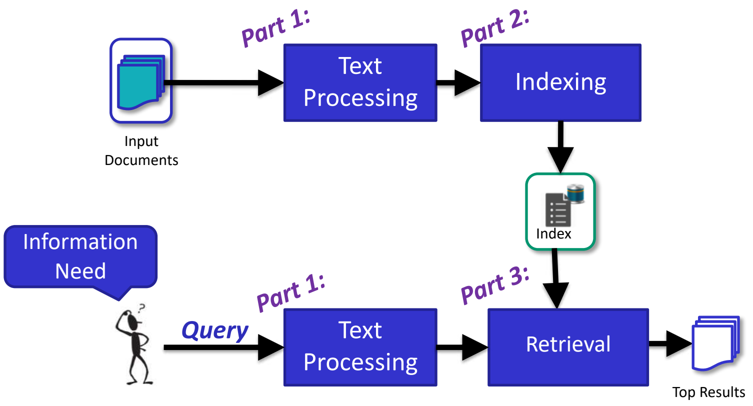
The Retrieval Process (Conceptual)



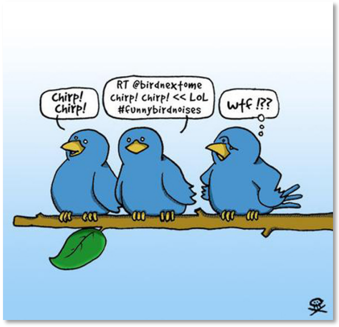
Ranked in order of ***relevance***

Indexing::: Represent each document; identify the meaning; What is the document **about**?

The Retrieval Process (In Practice)



6



Building a retrieval system

PART 1: TEXT PROCESSING

How Do We Represent Text?

- Remember: Typically, IR models use statistical properties of text rather than linguistic
- "**Bag of words**"
  - Treat all the words in a document as index terms
  - Assign a "weight" to each term based on "importance" (e.g. term frequency or, in simplest case, presence/absence of term)
  - Disregard order, structure, meaning, etc. of the words
  - **Simple, yet effective**!
- **Assumptions**
  - Term occurrence is independent
  - Document relevance is independent
  - "Words" are well-defined

Let's also assume that documents have been collected and converted into plain text

Documents

Information
vs.
Document

- Unit of retrieval
  - Web page? email; tweets;…
- Passage of free text
  - Composed of text, strings of characters from an alphabet
  - Composed of **words** of natural language
    - Newspaper article, a journal paper, a dictionary definition, email messages
  - Size of documents
    - Arbitrary
    - Email vs. newspaper article vs. journal paper

Effect on Retrieval …
More later

What's a Word?

天主教教宗若望保祿二世因感冒再度住進醫院。
這是他今年第二度因同樣的病因住院。

وقال مارك ريجيف - الناطق باسم
الخارجية الإسرائيلية - إن شارون قبل
الدعوة وسيقوم للمرة الأولى بزيارة
تونس، التي كانت لفترة طويلة المقر
الرسمي لمنظمة التحرير الفلسطينية بعد خروجها من لبنان عام 1982.

Выступая в Мещанском суде Москвы экс-глава ЮКОСа заявил не совершал ничего противозаконного, в чем обвиняет его генпрокуратура России.

भारत सरकार ने आर्थिक सर्वेक्षण में वित्तीय वर्ष 2005-06 में सात फ़ीसदी विकास दर हासिल करने का आकलन किया है और कर सुधार पर ज़ोर दिया है

日米連合で台頭中国に対処…アーミテージ前副長官提言

조재영 기자= 서울시는 25일 이명박 시장이 `행정중심복합도시" 건설안에 대해 `군대라도 동원해 막고싶은 심정"이라고 말했다는 일부 언론의 보도를 부인했다.

Sample Document

McDonald's slims down spuds

Fast-food chain to reduce certain types of fat in its french fries with new cooking oil.

NEW YORK (CNN/Money) - McDonald's Corp. is cutting the amount of "bad" fat in its french fries nearly in half, the fast-food chain said Tuesday as it moves to make all its fried menu items healthier.

But does that mean the popular shoestring fries won't taste the same? The company says no. "It's a win-win for our customers because they are getting the same great french-fry taste along with an even healthier nutrition profile," said Mike Roberts, president of McDonald's USA.

But others are not so sure. McDonald's will not specifically discuss the kind of oil it plans to use, but at least one nutrition expert says playing with the formula could mean a different taste.

Shares of Oak Brook, Ill.-based McDonald's (MCD: down $0.54 to $23.22, Research, Estimates) were lower Tuesday afternoon. It was unclear Tuesday whether competitors Burger King and Wendy's International (WEN: down $0.80 to $34.91, Research, Estimates) would follow suit. Neither company could immediately be reached for comment.

…

“Bag of Words”

14 × McDonalds

12 × fat

11 × fries

8 × new

7 × french

6 × company, said, nutrition

5 × food, oil, percent, reduce, taste, Tuesday

…

Lexical Analysis (aka Tokenisation)

- The process of converting a stream of characters (the text of the documents) into a stream of words (the candidate words to be adopted as index terms)
  - Identification of the words in the text (not as easy as it sounds!)
  - Treating digits, hyphens, punctuation marks, and the case of the letters
- Cases to be considered with care :
  - Numbers (e.g. 1999 vs. 510B.C)
  - Hyphens (e.g. state-of-the-art vs. B-49)
  - Punctuation (e.g. 510B.C vs. list.id)
  - Case of letters (e.g Bank vs. bank)

Small decisions can have major impact on the effectiveness of some queries

Stopwords Removal

- Words which are too frequent among the documents in the collection are not good discriminators
  - Called **stopwords** (or ***function words***)
  - Filters out articles, prepositions, conjunctions (e.g., the, am, and) that have very low discrimination values for retrieval purposes
  - Reduces the size of the indexing structure considerably
- Strategies for stopword removal
  - List look-up (negative dictionary/stopword list)
  - Usage of frequency information from other collections
  - Frequency analysis (all terms occurring in more than 80% of documents removed)
- NB: Can be important in combinations
  - e.g., “to be or not to be”
  - Modern IR: Stopwords often not removed at indexing, but removed as part of query processing

Briefly describe three strategies for stop-word removal?

Effect of Word Variants

- We expect the retrieval system to be **robust**
  - If the query contains plural (e.g., courses) & the document contains only the singular form (course) of that word; we expect the document to still be retrieved
  - From a user perspective: No need to submit query term variants
  - From a system perspective: no need to expand the query terms with variants
- **Conflation** reduces word variants into a single form (*A linguistic process*)
  - The rationale for such a procedure is that similar words generally have similar meaning
  - **Stemming** is a specific conflation technique

Stemming

What is stemming?

- A stemming algorithm reduces all words with same root into a single root
- A stem is the portion of a word which is left after the removal of its affixes (i.e., prefixes and suffixes)
  - e.g., connect is the stem for the variants connected, connecting, and connections.
- Two words that were initially treated independently become interchangeable
  - Increases retrieval of all possibly relevant documents
- NB: Stemming can be done at indexing time or as part of query processing (like stopwords)

Context-sensitive Transformation Grammar

- Rule 2.1 (.*)SSES -> /1SS
- Rule 2.2 (.*[AEIOU].*)ED->/1
- Rule 2.3 (.*[AEIOU].*)Y->/1I

- A complete algorithm for stemming involves the specification of many such rules to match the same token
  - Iterative longest match

Briefly describe how the stemming process is done?

Porter, M.F. (1980): An algorithm for suffix stripping, in *Program - automated library and information systems*, 14(3): 130-137

Effect of Stemming

- Compression
  - May reduce the index size 10-50%
- Stems are thought to be useful for improving retrieval performance
  - 5-10% improvement for English, up to 50% in Arabic
  - However, many Web search engines do not adopt stemming in its **strict form**
  - They try to consider stemming on a **query-by-query** basis, for instance detection if the word is important (e.g. a named entity or noun), and stemming if not.
- Problem
  - GRAVITY has two meanings
  - GRAVITATION -> GRAVITY
  - Prevent interpretation of word meanings

Discuss the pros and cons of stemming?

Text Processing Example

- Original Text

Twinkle, twinkle, little bat.
How I wonder what you're at!
Up above the world you fly.
Like a tea-tray in the sky.

- Tokenisation

twinkle twinkle little bat how i wonder what you re at up above the world you fly like a tea tray in the sky

- Stopword removal

twinkle twinkle little bat wonder world like tea tray sky

- Stemming

twinkl twinkl littl bat wonder world like tea trai sky

Thus far …

Documents ➔ {Words}

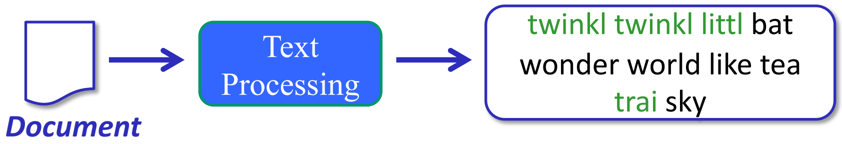
- Chopped the text into words (token)
  - ***Tokenisation (Lexical Analysis)***
- Removed Functional words
  - ***Stopword removal***
- Compressed word variations
  - ***Stemming***
- Result is
  - ***{docID, term, frequency} triplets***

Building a retrieval system

PART 2: INDEXING

Indexing

- The text processing step allows us to create concise bag-of-word representations for each document



- However, how do we find these documents quickly when a user enters a query?
  - It would be far too slow to match each document in turn against the query; there may be billions, or trillions of documents to consider!

Indexing is a process of storing the document representations created by the text processing step in a fast look-up structure

Inverted Index

What are the benefits of using an inverted index?
What are the costs?

- The primary data structure generated by the indexing process is the inverted index (aka inverted file)
- Main idea: Index Index $^{-1}$
  - Invert 'Document ➔ Term' index(es) to a single 'Term ➔ Document' index
  - The speed of retrieval is maximised by considering only those terms that have been specified in the query
- The speed is achieved only at the cost of *very substantial* storage and processing overheads
- Basic steps:
  - Make a "dictionary" of all the tokens (words) in the collection
  - For each token, list all the docs it occurs in
  - Do a few things to reduce redundancy in the data structure

Inverted Index (Conceptually)

An Inverted File is a document-term matrix representation "inverted" so that rows become columns and columns become rows

docs	*t1*	*t2*	*t3*
D1	1	0	1
D2	1	0	0
D3	0	1	1
D4	1	0	0
D5	1	1	1
D6	1	1	0
D7	0	1	0
D8	0	1	0
D9	0	0	1
D10	0	1	1

Terms	D1	D2	D3	D4	D5	D6	D7	…
t1	1	1	0	1	1	1	0	
t2	0	0	1	0	1	1	1	
t3	1	0	1	0	1	0	0	

Index $^{-1}$: $\{kw_j\} \rightarrow doc_i$

Index: $doc_i \rightarrow \{kw_j\}$

How Are Inverted Files Created

- Documents are parsed to extract tokens. These are saved with the Document ID.

Doc 1

Now is the time for all good men to come to the aid of their country

Doc 2

It was a dark and stormy night in the country manor. The time was past midnight

Term	Doc #
now	1
is	1
the	1
time	1
for	1
all	1
good	1
men	1
to	1
come	1
to	1
the	1
aid	1
of	1
their	1
country	1
it	2
was	2
a	2
dark	2
and	2
stormy	2
night	2
in	2
the	2
country	2
manor	2
the	2
time	2
was	2
past	2
midnight	2

How Inverted Files are Created

- After all documents have been parsed the inverted file is sorted alphabetically.

Term	Doc #
now	1
is	1
the	1
time	1
for	1
all	1
good	1
men	1
to	1
come	1
to	1
the	1
aid	1
of	1
their	1
country	1
it	2
was	2
a	2
dark	2
and	2
stormy	2
night	2
in	2
the	2
country	2
manor	2
the	2
time	2
was	2
past	2
midnight	2

→

Term	Doc #
a	2
aid	1
all	1
and	2
come	1
country	1
country	2
dark	2
for	1
good	1
in	2
is	1
it	2
manor	2
men	1
midnight	2
night	2
now	1
of	1
past	2
stormy	2
the	1
the	1
the	2
the	2
their	1
time	1
time	2
to	1
to	1
was	2
was	2

25

How Inverted Files are Created

- Multiple term entries for a single document are merged.
- Within-document term frequency information is compiled.

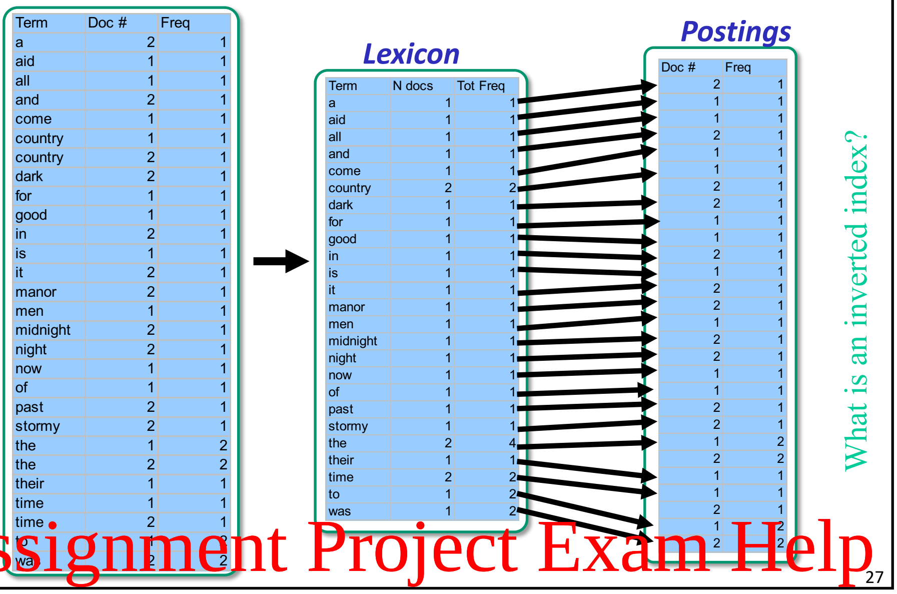
Term	Doc #
a	2
aid	1
all	1
and	2
come	1
country	1
country	2
dark	2
for	1
good	1
in	2
is	1
it	2
manor	2
men	1
midnight	2
night	2
now	1
of	1
past	2
stormy	2
the	1
the	1
the	2
the	2
their	1
time	1
time	2
to	1
to	1
was	2
was	2

→

Term	Doc #	Freq
a	2	1
aid	1	1
all	1	1
and	2	1
come	1	1
country	1	1
country	2	1
dark	2	1
for	1	1
good	1	1
in	2	1
is	1	1
it	2	1
manor	2	1
men	1	1
midnight	2	1
night	2	1
now	1	1
of	1	1
past	2	1
stormy	2	1
the	1	2
the	2	2
their	1	1
time	1	1
time	2	1
to	1	2
was	2	2

26

How Inverted Files are Created



Term	Doc #	Freq
a	2	1
aid	1	1
all	1	1
and	2	1
come	1	1
country	1	1
country	2	1
dark	2	1
for	1	1
good	1	1
in	2	1
is	1	1
it	2	1
manor	2	1
men	1	1
midnight	2	1
night	2	1
now	1	1
of	1	1
past	2	1
stormy	2	1
the	1	2
the	2	2
their	1	1
time	1	1
time	2	1
to	1	2
was	2	2

Lexicon

Term	N docs	Tot Freq
a	1	1
aid	1	1
all	1	1
and	1	1
come	1	1
country	2	2
dark	1	1
for	1	1
good	1	1
in	1	1
is	1	1
it	1	1
manor	1	1
men	1	1
midnight	1	1
night	1	1
now	1	1
of	1	1
past	1	1
stormy	1	1
the	2	4
their	1	1
time	2	2
to	1	2
was	1	2

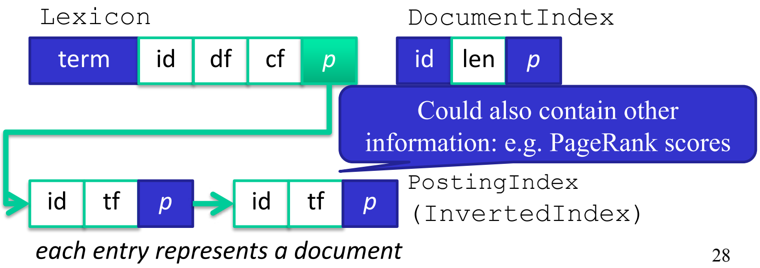
Postings

Doc #	Freq
2	1
1	1
1	1
2	1
1	1
1	1
2	1
2	1
1	1
1	1
2	1
1	1
2	1
2	1
1	1
2	1
2	1
1	1
1	1
2	1
2	1
1	2
2	2
1	1
1	1
2	1
1	2
2	2

What is an inverted index?

Indexing Summary

- Indexing produces a fast document search structure, containing:
  - **Term Dictionary (Lexicon):** Records the list of all unique terms and their statistics
  - **Inverted Index**: Records the mapping between terms and documents
- A **third data structure** often also exists:
  - **Document Index:** Records the list of all documents and their statistics



Lexicon

term | id | df | cf | *p*

DocumentIndex

id | len | *p*

Could also contain other information: e.g. PageRank scores

id | tf | *p* | id | tf | *p*

PostingIndex (InvertedIndex)

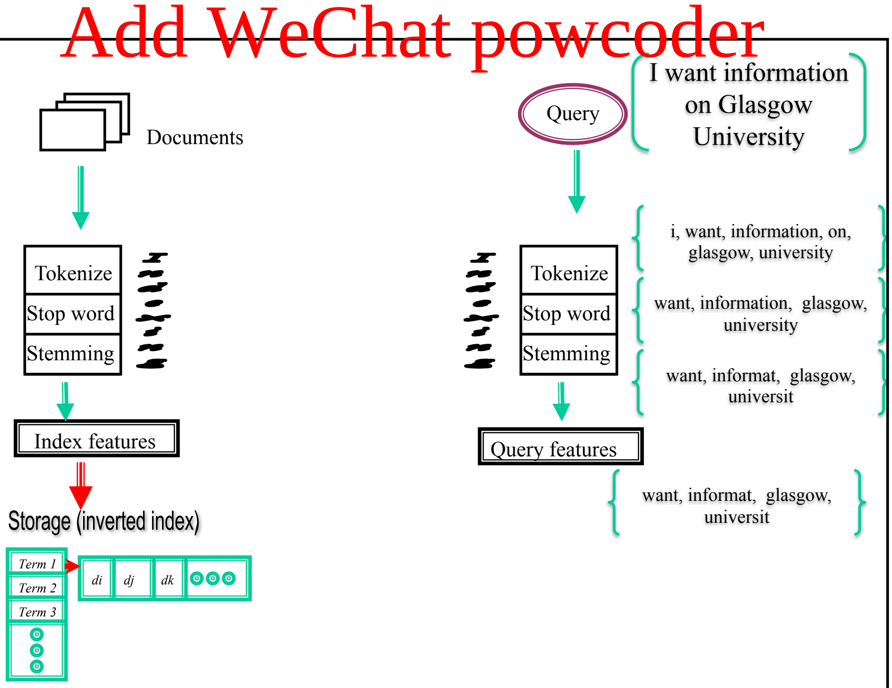
each entry represents a document

Inverted Index: Summary Steps

- Identify each document and note its ***id***
- Extract terms from the document
- Order them alphabetically and collect frequency
- Remove stopwords
- Stem the remaining words and update the frequency – add **document id** as well
- Repeat the above steps for all documents & then build the inverted index

Discuss the procedures for building an inverted index

This is often an offline process performed only once



Documents

Tokenize

Stop word

Stemming

Index features

Storage (inverted index)

Term 1

Term 2

Term 3

di dj dk

Query

I want information on Glasgow University

i, want, information, on, glasgow, university

want, information, glasgow, university

want, informat, glasgow, universit

Tokenize

Stop word

Stemming

Query features

want, informat, glasgow, universit

Building a retrieval system

PART 3: RETRIEVAL

Retrieval

- So far we have discussed how:

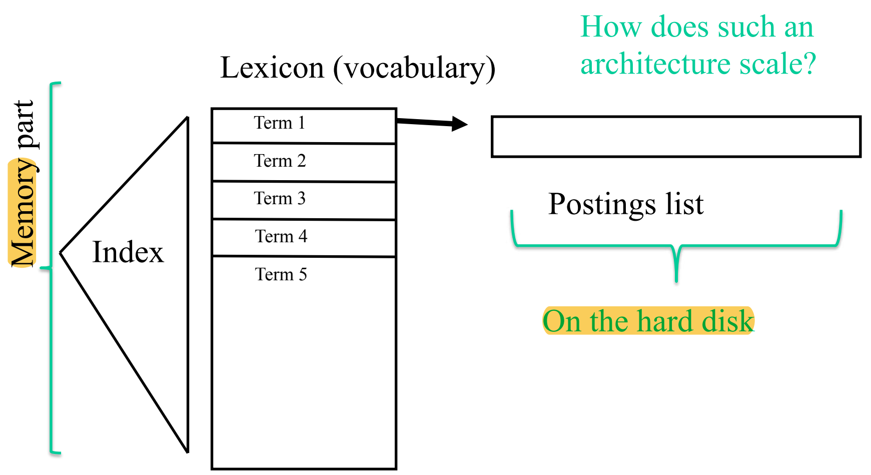
Text Processing — *Can be used to generate a concise bag-of-words representation of each document*

Indexing — *Can store these document representations in a fast searchable structure*

- But how do we find the documents that are actually relevant to the user's query?

Retrieval is the process of using the index to rank documents for the user query

Searching an Inverted File



(Simple) Search Algorithm

- Lexicon search
- Fetching of occurrences
- Manipulation of occurrences

Relevance Estimation

- The process in which we compute the **relevance** of a document for a query
- For example, relevance can be estimated using a similarity measure
- A similarity measure comprises:
  - Term weighting scheme which allocates numerical values to each of the index terms in *a query or document* reflecting their relative **importance**
  - Similarity coefficient - uses the term weights to compute the overall *degree of similarity* between a query and a document

What are the components of a similarity measure?

See IR Models lectures later

Basic Retrieval

Let's start with a basic example of how the inverted index is used

- Query for 'time' **and** 'dark'

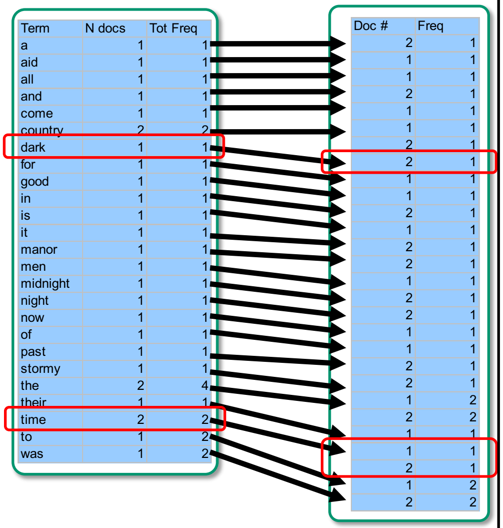
There are 2 docs with "time" in dictionary

- **IDs 1** and **2** from posting file

There is 1 doc with "dark" in dictionary

- **ID 2 from posting file**

Therefore, only doc ID 2 satisfy the query.



Term	N docs	Tot Freq
a	1	1
aid	1	1
all	1	1
and	1	1
come	1	1
country	2	2
dark	1	1
for	1	1
good	1	1
in	1	1
is	1	1
it	1	1
manor	1	1
men	1	1
midnight	1	1
night	1	1
now	1	1
of	1	1
past	1	1
stormy	1	1
the	2	4
their	1	1
time	2	2
to	1	2
was	1	2

Doc #	Freq
2	1
1	1
1	1
2	1
1	1
1	1
2	1
2	1
1	1
1	1
2	1
1	1
2	1
2	1
1	1
2	1
2	1
1	1
1	1
2	1
2	1
1	2
2	2
1	1
1	1
2	1
1	2
2	2

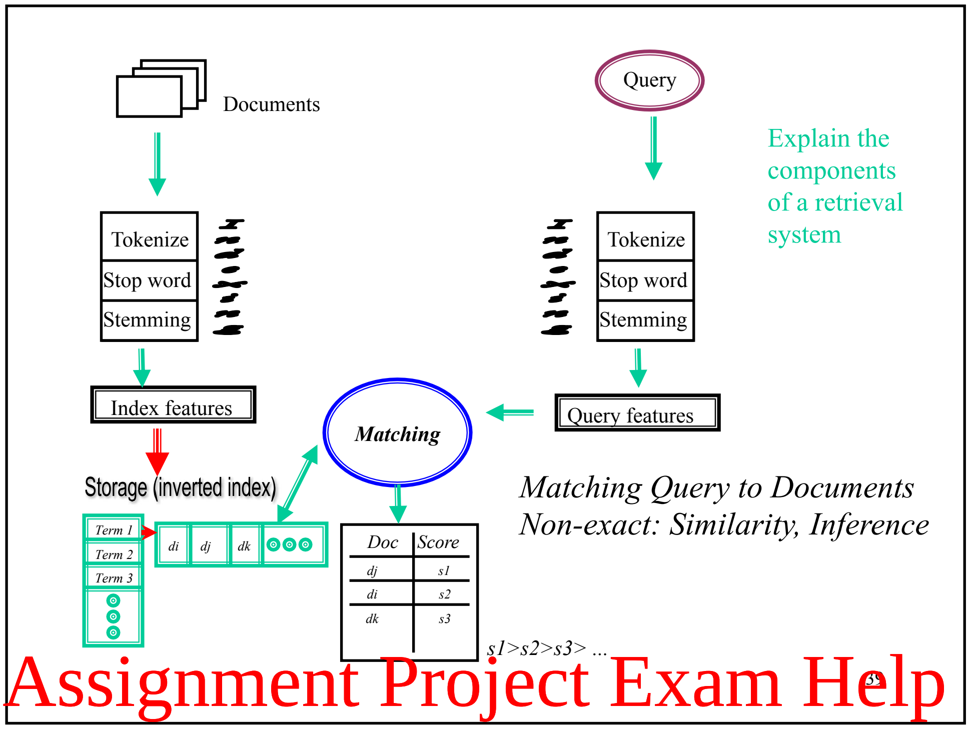
Best-Match Retrieval Algorithm

- The previous example illustrates **binary** AND search
  - **All** of the query terms need to be matched
  - **All** terms are considered equal
- Instead, we could use a basic Best-Match ranking

```
For each document I, Score(I) = 0;
For each query term
    Search the lexicon list
    Pull out the postings list
    for each document J in the list,
        Score(J) = Score(J) + 1;
```

Retrieval based on Keywords (Best-Match Retrieval)

- Compare the terms in a document and query
- Compute similarity between each document in the collection and the query based on the terms that they have in common
- Sorting the documents in order of decreasing similarity with the query
- The output is a ranked list and displayed to the user – the top docs are more relevant as estimated by the system



Doc	Score
dj	s1
di	s2
dk	s3

Explain the components of a retrieval system

Matching Query to Documents
Non-exact: Similarity, Inference