**Workshop 2: Principles of IR**

Examples of widely used IR systems

* Google
* Bing

Example of specialist domain IR:

* Scientific Retrieval
* Legal databases
* Social media platforms

Not only text is returned from IR but also speech and music. e.g. Siri, Google Assist, Bixby.

Definition of IR: **Finding** material that is seen as **unstructured** by a computer that satisfies **information need** from within **large collections.**

Unstructured Material:

This means that the computer does not see an obvious structure that is always the same. The opposite to these are rigidly structured databases. Usually there is document structure for the human user e.g. titles and paragraphs. There can also be visualisation structure such as HTML, XML, Latex etc.

Information Need:

This is the information the user wants to know more about. Implemented using a query.

There are 3 types of information needs:

* Informational: Information is needed to learn more about something.
* Navigational: Requiring the information to go to some page.
* Transactional: Need to do something using the web.

Characteristics of the 3 types:

Navigational:

* Intention is to find a particular website or webpage.
* Company/business/org name
* Domain suffix
* Length of query (<3)
* Google classifies this type of query as a ‘go query’

e.g. Entering YouTube into the search bar of a browser.

Transactional:

* Queries containing interact terms (buy, chat etc.)
* Queries containing obtaining terms (download, get etc.)
* Terms relating to movies, songs, lyrics, recipes, images, humour etc.

e.g. download movies

Informational

* Use of question words
* Query length > 2
* “Queries that cover a broad topic for which there may be thousands of relevant results” Wikipedia
* User most likely not looking for specific information or website.
* Neither navigational nor transactional

e.g. What does suffix mean?

Why is classification of important need important?

\*\*\*---\*\*\* Add classic search model image

User behaviour

* An average user makes 3 searches per day.
* 78% of queries not modified?
* Searches are ambiguous, Ill-defined and imprecise
* Search have grammatical errors and spelling mistakes
* A lot of queries are very short

Relevance

* How can this be measured?
* Quantitatively or Qualitatively?
* Scale? Could be difficult
* How long the user stays on the page.
* If navigated to another page from the clicked-on page.
* Implementing relevance is not trivial.

Implementing IR systems

* Need to find similarities between query terms and documents
* Index the documents, this means reducing the documents to terms which tell us what the document is about.
* Queries can then be mapped to index terms
* Index construction is not trivial

Indexing to support search

* Acquire documents through crawling, feed etc.
* Generate index for each document
* Store the index in a easy to search form

Search process

* Reduce the query to indices
* Map the query against the index
* Retrieve and rank the results
* Help users browse the results and re-formulate the query (e.g. do you mean \_\_\_?)

Representation of documents of index

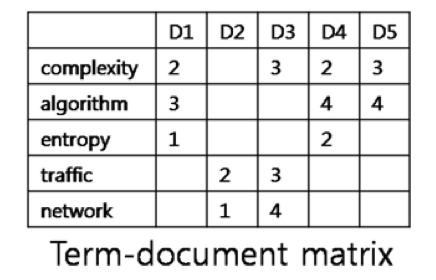
* Which words to use as indices. All or some?
* How do you deal with singular and plurals?
* Does knowledge of the words in the document help? E.g. knowing Manchester is AKA MCR
* How can you determine what the document is mainly about?

“bag of words” model

* Use all words in the document as indices
* Rank the words from ascending to descending occurrence
* How does this model deal with negations? E.g. this paper is not about amphibians live.
* Would it work for all languages?
* Is the meaning lost without the order?
* Is the meaning lost without context?

Term-document matrix

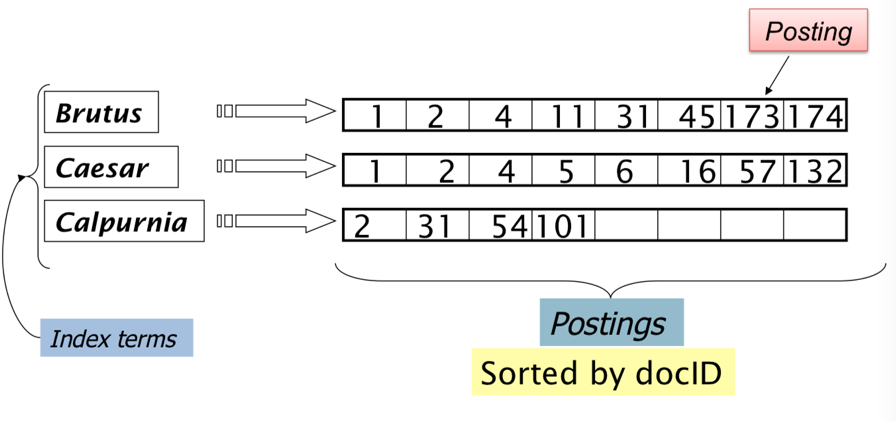
* Represents a word against documents and the intersection gives the number of times the word appears in the document.



* A term-document incidence matrix contains only a 0 or 1 at the intersection i.e. Either the document contains the word or it does not.
* A small collection that has something like 1 million documents can result in a massive matrix
* So if there is around 1 million documents and 500000 unique words in the collection then the matrix would have around 500,000,000,000 (half a trillion) possible entries.
* And the matrix could be sparse: No more than 1 billion non-zeros.
* Solution would be to only store the 1s

Inverted Index

* For each term to we store a list of all documents that contain t.



* We need variable size postings lists
* On disk, a continuous run of posting is used
* In memory, can use linked lists or variable length arrays.
* Better then matrix as only space which is needed is used.
* Sort by terms then document ID.
* Multiple term entries in a single document are merged
* Split into dictionary and postings

