IR week 2. Intro to IR Models.

1. IR vs. Data Retrieval.

- Data collections: well structured collections. involves the selection of a fixed set of data based on a well-defined query.
- Information collection: semi-structured or unstructured. involves the retrieval of documents of natural language

2. IR vs. IT (filtering).

main difference: the nature of the information need; the nature of the document set

IR systems aruse 'interpret' the semantic content of the documents and rank the documents in relation to user's information need.

IT is often used to describe systems that identify relevant information for users in responses to an information need.

Similarities: similar representations
mathematical models
comparison approaches

In IR, we typically deal with one-off information needs and a relatively static collection (fixed information range)

In IT, considers the information need as being a long-standing information and document collection is viewed more as a 'stream' of information with the system making a decision regarding its relevance without access to the complete collection

3. User Role.

In traditional IR ... → the collections tended to be centralized and the user actions limited.

In more modern IR systems,

4. IR system architecture

At a high level, we can view an IR system as comprising:

- Document Set / Collection (eg. web, articles, tweets, etc.)
- Queries (representation of users information needs).
- Pre-processing components (e.g stemming)
- Comparison algorithm (determine the similarity between the query and the D).
- User Feedback module.

- In the most general case, the documents in the document collection and the information query are in the same format - namely, natural language

are then processed in some manner to produce an internal representation of both document and query.

↓

is also an format suitable for the comparison algorithm.

↘ depanding on the underlying model chosen in the IR system.

return some estimate of relevance (or similarity) for each document - query pair.

usually allow the system to rank all documents with respect to relevance to a query: ↓

The top-ranked documents can then be presented to the user.

- Feedback module.

User can offer feedback on the usefullness/relevance of the returned docs.

↓

Evidence from this feedback is then used to typically modify the users' query. ⇒ include. the incorporation of newer query terms.

, the removal of existing term and/or

the possible re-weighting of terms.

⇒ Goal: to create a better query for the user's information need leading to a better return set being returned.

5. Pre-processing: (NLP)

Applications of a set of well-known techniques to the documents and queries prior to any comparison.

- Stemming: remove common suffixes from terms occurring in the docs.
- Stop-word removal: the removal of highly frequent words/terms from docs.
  - These words add little semantic meaning to the doc. (include articles and conjunctions) The overal goal: is to reduce similar words to a common root form by identifying morphological derivations of words.
- thesaurus construction
  - ↓ try to identify synonyms within docs.

D set (underlying docs). Q set (keywords, phrases..)

same pre-processing

can add some words

paragraphs: can be represented as bags of words.

Stemming:
- disadv: over-stemming
  - skiing
  - skies
  - absorb
  - absorp
- partial string match
  - a b c d e f
  - k a b c d e f g

Stop-words
- disadv: sometimes has meaning in phrase.
  - "The the" sometimes has certain meaning

Thesaurus construction: thesis ↔ dissertation or thesis

related words: messi ↔
- soccer
- barcelona
- psg

6. IR Models.

A model can be viewed as a tuple: $[D, Q, F, R(q_i, d_j)]$.

- D is the set of logical representation of the docs
- Q is the set of the user information needs (D.Q)
- F is the framework adopted for modeling the representations & their relationships (D.Q)
- R is a ranking function which produces a ranking of docs W.R.T. estimated relevance to a qu

In most models, we have a set of index terms $t_1, t_2, \cdots, t_n$.
A weight w_{ij} is assigned to each term t_i occuring in the document.
We can also view the query as a set of terms with associated weights.

D (pseudo set).
(could be a bag of words)
(a set of terms & stems)

Q (a set of terms)

pre-processing

↓

framework (mathematical).

↓

Ranking (similarity (query, doc))

Indexing An index term is a word or expression, which may be stemmed, describing or characterizing a document $T = \{t_1, t_2, \cdots t_n\}$

	t_1	t_2	...	t_n
d_1	w_{11}	w_{21}	...	w_{n1}
d_2	w_{12}	w_{22}	...	w_{n2}
⋮				
d_n	w_{1n}	w_{2n}		w_{nn}

w_{11}: how well the term describe (related to) this document

A document is any subset of T.

↓

how to assign weights

↓

This matrix could be huge / sparse ⇒ it could be inefficient.

$\vec{d_j} = (w_1, w_2, \cdots)$. (weights can be stored in vectors, some of them could be 0).

Thus, how to find a good weight for each term?

7. Boolean Model.

based on a Boolean algebra and the set theory:

A doc contains a term or a set of terms that satisfied the query, then the document is relevant to that query.

a binary matrix

	t_1	t_2	...	t_m
d_1	0	0		1
d_2	0	1	0	0
⋮				
d_n				

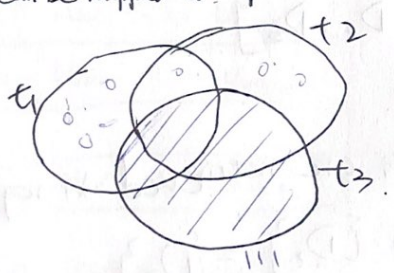
Adv: simplicity, clean formalism

Disadv: people often have difficulty formulating expressions
it suffers badly from NL features (synonymy & polysemy)
it pays no attention to frequency of terms in docs
docs are considered either relevant or non-relevant
(no potential for partial matching / no real ranking allowed)
terms in docs are considered independent of each other

e.g. $q = t_1$ AND (t_2 OR (NOT t_3))

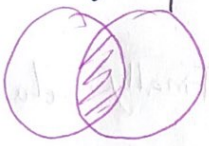
Solutions: $q =$ 000

Can be mapped to:



disjunction (OR)

t_1	t_2	t_3
1	1	1
0	1	1
0	0	1



Conjunction (AND)

- An index term is a word or expression, which may be stemmed, describing or characterizing a document, such as a keyword given for a journal article. Let $T = \{t_1, t_2, \dots, t_m\}$ be the set of all such index terms.
- A document is any subset of T, let $D = \{D_1, \dots D_n\}$
- A query is a Boolean expression Q in normal form:

$$Q = (W_1 \vee W_2 \vee \cdots) \wedge \cdots \wedge (W_i \vee W_{i+1} \vee \cdots).$$

where W_i is true for D_j when $t_i \in D_j$

We seek to find the set of documents that satisfy Q

① for each W_j in Q, find the set S_j of doc that satisfy W_j: $S_j = \{D_i \mid W_j\}$.

② Then the set of documents that satisfy Q is given by: $(S_1 \cup S_2 \cup \cdots) \cap \cdots \cap (S_i \cup S_{i+1} \cup \cdots)$

Example:

Let the set of original (real) documents be $O = \{O_1, O_2, O_3\}$.

where O_1 = "Bayes' principle - - - - - - - - - - ,"

O_2 = "Bayesian decision theory" - - - - - - - - - - ,"

O_3 = "Bayesian epistemology" - - - - - - - - - - ,"

Let the set T of terms be: $T = \{t_1 = \text{Bayes' principle}, t_2 = \text{probability}, t_3 = \text{decision-making}, t_4 = \text{Bayesian epist}\}$.

Then, the set D of documents is as follows: $D = \{D_1, D_2, D_3\}$.

where $D_1 = \{\text{probability}, \text{Bayes' principle}\}$ $D_2 = \{\text{probability}, \text{decision-making}\}$

$D_3 = \{\text{probability}, \text{Bayesian epistemology}\}$.

Let the query Q be: $Q = \text{probability} \wedge \text{decision-making}$

Then to retrieve the relevant documents:

① firstly, the following ~~set~~ S_1 and S_2 of documents D_i are obtained (retrieved): $S_1 = \{D_1, D_2, D_3\}$.

$S_2 = \{D_2\}$.

② Finally, the following documents D_i are retrieved in response to Q. $Q: \{D_1, D_2, D_3\} \cap \{D_2\} = \{D_2\}$.

This means the the original doc O_2 (corresponding to D_2) is the answer to Q.

Obviously, if there is more than one document with the same representation every such document is retrieved. Such documents are indistinguishable in the Binary IR.

8. Vector Space Model

Terms can have a non-binary weights in both queries and documents.

Here, term weights are used in the comparison between documents and queries. ~~We~~

We can sort the documents ~~and~~ based on their degree of similarity and return a ranked list to the user.

Each term in both the documents and the queries will have an associated weight.

Hence we can represent documents and queries as n-dimensional vectors:

$$\vec{d_j} = (w_{1j}, w_{2j}, \dots, w_{nj})$$

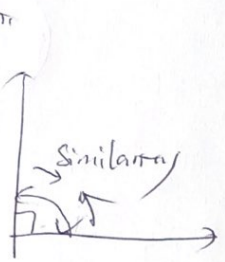
$$\vec{q} = (w_{1q}, w_{2q}, \dots w_{nq})$$

$\vec{d}$	t_1	t_2	...	t_m
d_1	w_{11}	w_{21}	...	w_{m1}
d_2	w_{12}	w_{22}	...	w_{m2}
:				
d_n	w_{1n}	w_{2n}	...	w_{mn}

$\vec{q}$ = w_{1q} w_{2q} ... w_{nq}

0.1 0.6 ... 0.2

definite: $0 \le w_{ij} \le 1$



Similarity:

$$\vec{a} \cdot \vec{b} = |\vec{a}| \cdot |\vec{b}| \cdot \cos(\vec{a}, \vec{b}) \Rightarrow \cos(\vec{a}, \vec{b}) = \frac{\vec{a} \cdot \vec{b}}{|\vec{a}|\,|\vec{b}|}$$

$$Sim(q, d_j) = \cos(\vec{q}, \vec{d_j}) = \frac{\vec{q} \cdot \vec{d_j}}{|\vec{q}|\,|\vec{d_j}|} = \frac{\sum_{i=1}^{n} w_{ij} \cdot w_{iq}}{\sqrt{\sum_{i=1}^{n} w_{ij}^2} \cdot \sqrt{\sum_{i=1}^{n} w_{iq}^2}}$$

$\sum_{t \in q \text{ and } d}$

is a function of the number of co-occuring terms between a query and a document, and of the weights of the terms in doc in the query

calculate instantly

already calculated

Return the short document which contains my query.

Normalise based on the length of query and documents.

8. Weighting Scheme.

The assignment of weights to terms is of important:

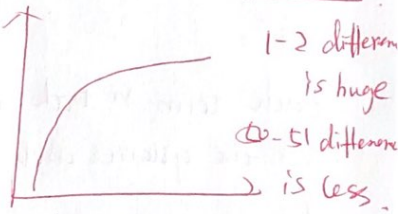
tf: term frequency in a document. (local)

idf: inverse document frequency across all documents. (global)

tf-idf weighting schemes:

$$w_{ij} = f_{ij} \times \log\left(\frac{N}{n_j}\right) = tf \times idf$$

normalisation



f_{ij} is some function of the frequency of t_i in doc d_j.

N is the number of docs in the collection.

n_i is the number of docs in the collection that contain term t_i.