Computational Linguistics: HCI, Evaluation Measures

RAFFAELLA BERNARDI

UNIVERSITÀ DEGLI STUDI DI TRENTO

E-MAIL: BERNARDI@DISI.UNITN.IT

Contents

1. Today main topics

We will see:

- HCI: NLIDB, QA, IQA, IR
- Evaluation Measures: Accuracy, Precision, Recall.

2. HCI via Natural Language

In the ’50 Machine Translation work pointed out serious problems in trying to deal with unrestricted, extended text in open domains. This led researchers in the ’60 and early ’70 to focus on question-answering dialogues in restricted domain.

Attention shifted from developing NL systems to solving individual language-related problems, e.g., to develop faster, and more efficient parsers.

Now, researchers are back to deal with unrestricted extended text and dialogues.

1. NLIDB: NL interface to DB
2. Dialogue Systems
3. QA: Question Answering
4. IQA: Interactive Question Answering
5. IR: Information Retrieval

All of them aim at assisting users to access data from some source.

3. Natural Language Interfaces to Data Bases

NLIDB refers to systems that allow the user to access information stored in a database by typing requests in some natural language. Its history (see Androutsopoulos for more details):

'60/'70 they were built having a particular DB in mind. No interest in portability issues. E.g., LUNAR

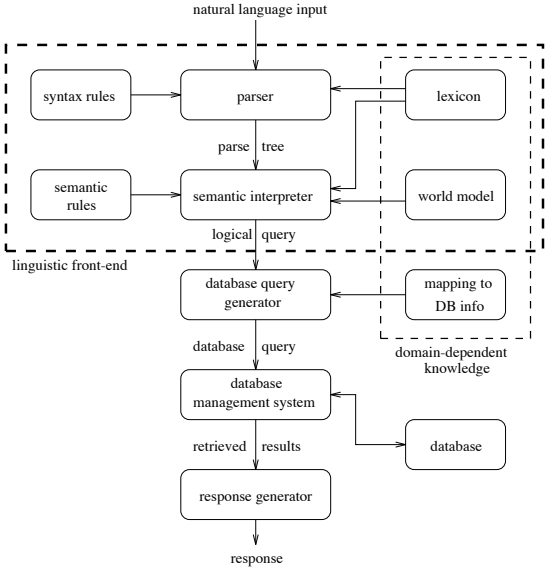
late '70 Dialogues; large DB; semantic grammars (domain dependent - no portable). E.g. LADDER

early '80 From English into Prolog evaluated against Prolog DB. Eg., CHAT-80

mid '80 popular research area. Research focused on portability issues. E.g. TEAM

'90 NLIDBs did not gain the expected commercial acceptance. Alternative solutions were successful (graphical or form-based interface). Decrease in the nubmer of papers on the topic.

3.1. Sample Architecture



3.2. Restricted NL input

Currently systems use limited subsets of NL.

Limitation user doesn't know which is this subset. Has to rephrase the question, does not know which questions could be handled.

Long term aim to broad the linguistic coverage.

Alternative approach deliberately and explicitly restrict the set of NL the user is allowed to input (controlled natural language.), e.g.: syntactic pattern, menu-base, ontology driven, complexity of NL fragments

Examples of today NLDBs:

- ACE: http://attempto.ifi.unizh.ch/site/tools/
- Geo http://www.cs.utexas.edu/users/ml/geo-demo.html
- PENG: http://www.ics.mq.edu.au/~peng/PengEditor.html
- Controlled Natural Language special gruup: http://www.sigcnl.org/

4. From the ’90 ..

Researchers got back to deal with unrestricted extended text and dialogues.

1. NLDB
2. Dialogue Systems,
3. QA
4. IQA

Slides taken from Bonnie Webber’s presentation at the BIT Seminars at FUB (December 2006).

5. Question Answering

- START: http://start.csail.mit.edu/
- QUALIM http://demos.inf.ed.ac.uk:8080/qualim/ (multilingual)
- QUETAL http://experimental-quetal.dfki.de/ (also images)
- QUADRA http://www.laancor.com/technology/quadra/ (in your PC)

New edition of SLP has a chapter on QA: https://web.stanford.edu/~jurafsky/slp3/28.pdf

5.1. History of ideas

QA has received a great deal of attention in recent years, but the area is not new:

- Simmons (1965) reviewed 15 implemented and working QA systems.
- Philosophical discussions of QA date back millenia
- Psychological issues in QA have been investigated since the 1930s.
- Best QA technology aimed to serve as a "front-end" to a database system:
  - Natural language questions rather than formal DB queries
  - User doesn't need to know how/where the data are stored.
  - QA systems maps questions to database queries (Possible interaction with user if ambiguity is detected.)
  - Problems of answer extraction left to the DB system.
  - QA system could use the DB data model and update model to answer or respond to certain types of questions.

5.2. Re-emergence of QA

Research in QA went quiet from the late 1980s until the late 1990s.

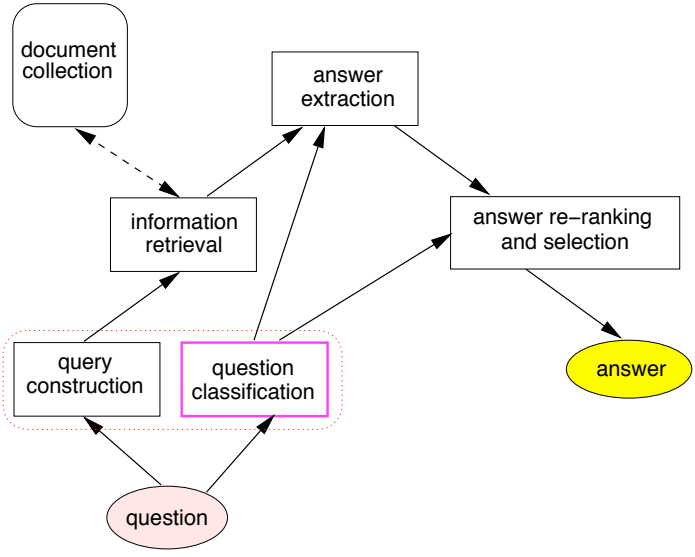
Re-emerged with the rise of search engines, shifting to QA over unstructured data (i.e., free text) from the web, or (for purposes of evaluation and/or comparison) from closed, quality-controlled corpora, e.g., newspaper and encyclopedia articles.

Formal, large-scale evaluated activity began with the NIST-run TREC QA track:

- 20 groups participated in 1999, 33 in 2005
- facilitated rapid dissemination of results and formation of a community
- dramatically increased speed at which new techniques have been adopted.

Database QA has also re-emerged (even more recently) focusing on robust domain independent techniques and on providing better guarantees that a user's question can be answered.

5.3. Sample of QA architecture



5.3.1. Question Classification: Why and How? Strategies for extracting answer candidates depends in part of the type of question: factoid, list, definition/other, event etc.

Within factoid QA, classifying questions (e.g., Person, Location, Date, Quantity) can provide semantics constraints on answer candidates.

Classification can be obtained by means of manually constructed rules (if a question starts with Who or Whom: type Person).

Manually constructed rule sets (can) have large coverage and reasonable accuracy, but they are not sufficient to support fine-grained classification.

Current approaches either just consider coarse classes, or they organize QA types in a hierarchy, picking up answer requirements from the most specific class that the classifier is able to reliably identify.

5.3.2. Question Classification: Problems Questions can be ambiguous: different senses may fit into different classes. E.g., "What is an SME"?

- A Small-to-Medium Enterprise (abbreviation)
- A company with fewer than 100 employees (definition)

Classes are not really disjoint, so even unambiguous question can fit into more than one class. Eg., "What is a bipolar disorder?"

Variability of expressing the same question, e.g.,

- What tourist attraction are there in Reims
- Where do tourist go in Reims
- What can I see in Reims
- What is worth seeing in Reims

5.3.3. Document Retrieval in QA

- QA over free text arose as an off-shoot of document retrieval
- In document retrieval, queries are interpreted as requests for documents about a topic
- QA requires answers not the documents containing them
- The answers to factoid questions are very locally in a document. Documents containing such local information are easily not tagged as relevant by standard document retrieval models.
- So document ranking by relevance may not correlate with a document's likelihood of containing an answer to a particular question
- If document retrieval fails to return documents containing answers, all subsequent processing is useless.

5.3.4. Document Retrieval in QA: Approaches Use standard IR engine (eg., Lucene) in a way more appropriate for QA:

- Break documents into passages and perform passage retrieval
- Use more complex strategies to index the documents/passage collection (stem, PoS, dependencies)

Apply additional filters.

5.3.5. Candidate Answer Extraction Goal: locate candidate answers within the retrieved texts (documents or passages).

Evidence used for candidate answers:

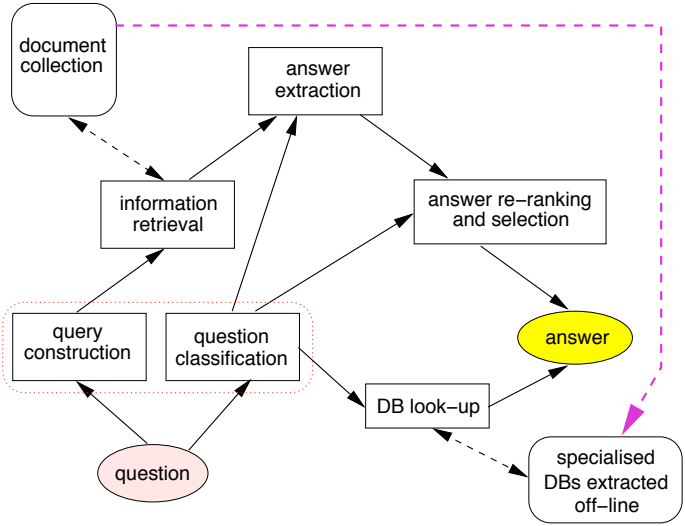
- string patterns
  - specifying string context in which answer string may be found.
  - specifying properties of the answer string
- presence of answer type linked to the question type
  - from prior indexing of the different types of named entities
  - on-the-fly named entity recognition
- patterns in light-weight logical form (e.g., arguments to particular predicates

5.3.6. Answer Re-ranking and Selection

Answer selection could reflect:

- confidence in query's ability to return a correct answer
- document ranking and query/document match
- frequency with which N-gram appears in set of returned snippets
- conformance with the answer type
- weight contributed by overlapping N-grams.

5.4. Enriched QA Architecture



5.4.1. Fact Caching

- Extracting particular kinds of facts off-line and caching them can speed up QA for simple factoid questions.
- Create patterns for fact extraction in whatever kind of terms have been used in indexing.
- Extracted facts stored in flat tables for later table look up
- Still need to recognize when a question can be answered via a cached fact.

5.4.2. Other QA issues

- Locating answer candidates and/or material relevant to the answer
- Improving extraction of answer candidates through better linguistic analysis
- Improved techniques for answer re-ranking and selection
- Answering list Qs, definition/other Qs, event Qs, Why Qs, How Qs, What's the difference Qs, Qs of opinion, etc.
- Fusing information when more than one piece of information is relevant to an answer
- Providing extended answers (summarising opinions, laying out what's known about an issue, providing justification for an answer.)
- Interactive Question Answering (different types of user and system-initiated interactions)

5.5. Understanding in QA?

In 2000 (TREC-9) semantically equivalent questions received some attention. For 54 of the questions additional variants were generated by paraphrasing the original question. All systems were sensible to the changes of the surface structure. This showed the need of further research aimed at deeper understanding of questions.

Since 2001, TREC also includes questions with no answers in the documents and questions with “list” answers. Both aspects requires some more level of understanding: systems to measure their certainty about an answer, on the on hand, and on the other they may need to synthesize the answer from multiple documents.

5.6. Challenges

This trend will be pushed further by the answer evaluation criteria the required presentation modes:

- answer must be correct, succinct, coherent and justified.
- Systems will need to handle cases when:
  - multiple answers are found,
  - when no answer is found, or
  - when contradictory answers are found.
- System will have to go beyond extraction of snippets of text to provide answer synthesis across sentences and across documents.

Providing appropriate coherent answers will be a major research area and will depend heavily on progress in text summarization.

5.7. Today research

TREC 2017: http://trec.nist.gov/pubs/call2017.html **Live QA Track** In this track systems generate answers to real questions originating from real users via a live question stream, in real time.

SemEval 2016: Community QA: http://alt.qcri.org/semeval2016/task3/

6. Interactive Question Answering

Simple “one-shot” QA pairs too simple to be considered IQA.

But IQA has been applied to many thing:

- single round of user-system interaction through relevance feedback questionaires in TREC’s 2006
- Extended negotiation dialogues (the system draws the user into a conversation, understands what the user is looking for, what the user has done and what the users knows.)

6.1. Characteristic of IQA

- Extended interaction –hence the need to model the context of the interaction and established by the interaction.

  No IQA without context modelling.

- System-initiated contributions and interactions:
  - System-initiated contributions to its answer turn
  - System-initiated interactions outside its answer turn: Context modelling does not itself constitute interaction.
- User-initiated interactions, including but not limited to:
  - asking domain-related questions
  - answering clarification questions posed by the system

    Users need to do more to achieve knowledge-related goals.

6.2. Context in IQA

Context is need to resolve context-dependent linguistic phenomena that occur in any extended text or dialogue (pronoun, anaphora, ellipsis, fragments or e.g., “other countries”)

- Q1: What museum in Florence was damaged by a major bomb explosion?
- A: Uffizi Gallery
- Q2: What kind of explosives were used?

- Q1: In what country did the game of croquet originate?
- A1: Ireland
- Q2: How about bowls?

6.3. Context in Open Domain IQA

At every turn, a system should check whether the topic of the interaction has:

- stayed the same?
- switched to an item from the previous context?
- switched to something new?

6.4. System-initiated Interactions

In Open Domain QA issues related to this problem have not been addressed, in DBQA/NLDB, most of the work dates back to the 80's and early 90's. E.g,

- What information is needed to take a particular type of initiative?
- When does the system have such information?
- What leads the system to take the initiative?

6.4.1. Type of system initiatives

- Question clarification:
  - Q1: What is the largest country in Southeast Asia?
  - S: By "largest" do you mean in land area of population?

  But, how much clarification is it wise to ask for? E.g., By "country in South Asia" do you mean countries south of China and east of India?

- Question Verification:
  - User: Restaurants on Page Mill Road?
  - Systems: I found 258 restaurants on Page Mill Road. Would you like to try searching by type of cuisine?

  N.B. CO-OP (Kaplan, 1983) used a similar strategies.

6.4.2. System-initiated contributions

- Justification
  - User: What's the largest country in Southeast Asia?
  - User: Largest by land mass:
    1. Indonesia: 1,919,440 Km^2 (CIA World Factbook)
    2. ...
- Facilitate users exploration of additional hypothesis.
- Contribute to user understanding of domain by correcting misconceptions and by answering questions related to one that has failed. E.g
  - User: Does China have the largest population of Muslims in Southeast Asia?
  - System: China is not considered part of Southeast Asia. Estimates of the Muslim population of China range from 20 million to 100 million. Indonesia has the largest population of Muslims in Southeast Asia. As of 2006, approximately 88 % of its 222 million people are Muslims.

6.5. System initiative in DB IQA

System initiative facilitated through update model: Justification

- User: Can I borrow Jurafsky & Martin?
- System: No. Until it is removed from the Reference Section, it is not available for borrowing.

6.6. Ontology in IQA

Annika Flycht-Eriksson investigates the use of ontologies in IQA.

The role of the interaction is to correctly interpret user requests, hence an important role of the ontology is to:

- organize the semantics of natural language expressions and
- organize the knowledge of the world as perceived by the user.
- provide an interface to external information sources from which the system collects answers to user requests.

6.7. Example

Example of a information exchange where ontological knowledge is used to guide clarification requests and database access:

- U: What do waterfowl look like?
- S: Choose one of the following Waterfowl: Barncale Goose, Tufted Duck, Bean Goose, Whooper Swan, Gadwall
- U: Gadwall
- S: You can receive information about the size and plumage of a Gadwall. Please, specify what you are interested in.
- U: size
- S: A Gadwall has a length of 56 cm from beak tip to tail. Information about wingspan is missing for Gadwall.

Similarly, the interaction could be used to try to map the vague proprieties to ones suitable for database access.

6.8. Characteristics for the Ontology

Design decision	Dialogue	IE
Concepts and Taxonomy		
Internal structure of concepts	-	-
Metaclasses	x	x
Subclass_Of	x	x
Multiple inheritance	x	?
Part-whole treatment	x	x
Properties and Relations		
Domain restrictions	x	x
Range restriction	x	x
Local	x	x
Instance	x	x
Class	-	U
Polymorph	x	x
Binary relations	x	x
Arbitrary n-ary relations	U	U
Default values	U	-
Cardinality restrictions on values	x	x
Procedural attachments for values	U	U
Instances		
Instances of categories	x	U
Multiple instantiation	x	x
Facts (instances of relations)	x	U
Claims (assertion of fact by instance)	?	?

6.9. Today research on Dialogues

SIGdial http://www.sigdial.org/workshops/conference18/

7. Conclusions

QA lesson:

- Never forget to look at what has been done in the past!
- Work on modules
- Flow of ideas across fields
- what was not successful in the past could be successful now

Keywords in NLP:

- systems
- hybrid approaches
- evaluation:

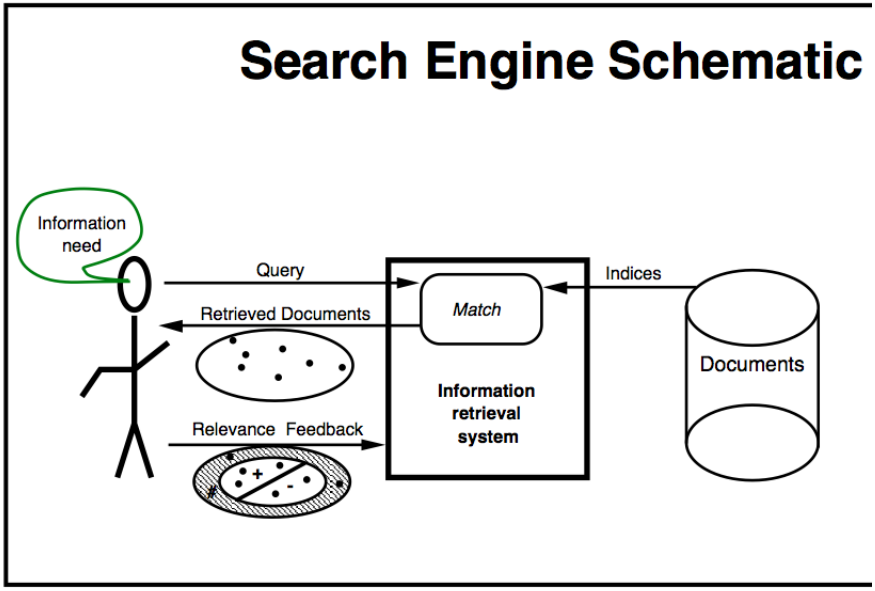
7.1. New trend

- New trendy task: Visual Question Answering (we will speak about this.).
- New approach: Neural Networks/Deep learning/Memory Network. NO MODULES. ONE Model/Network.

7.2. Evaluation campaigns

- TREC http://trec.nist.gov/
- Textual Entailment: http://pascallin.ecs.soton.ac.uk/Challenges/RTE/
- Sens eval: http://www.senseval.org/
- CLEF http://www.clef-campaign.org/
- TAC http://apl.jhu.edu/~paulmac/kbp.html

8. Information Retrieval



Finding Out About

8.1. History of IR

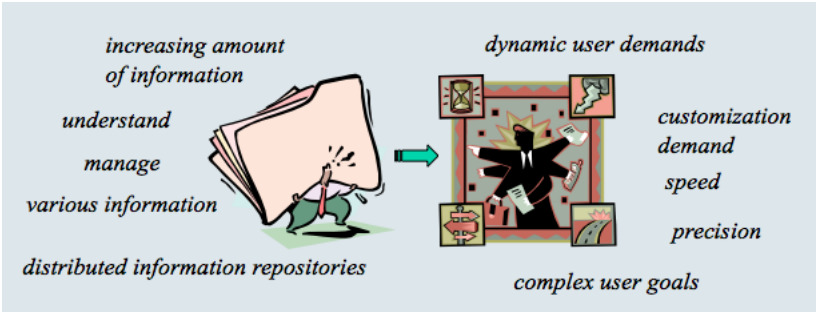
1960-70's *Small* text retrieval systems; basic boolean and vector-space retrieval models.

1980's *Large* document database system, many run by companies

1990's Searching FTPable documents on the *Internet*; Searching the World Wide Web (Yahoo, Altavista)

2000's Link analysis for web search (Google); QA (TREC QA track); Multimedia IR; Cross-Language IR; Document Summarization; Recommender systems, automated text categorization and clustering (iTunes, Amazon, Flick)

8.2. IR challenges



8.3. Retrieval: Example

Take a book of one million words, for instance “Shakespeare’s Collected Works”. You want to know which plays contain the words “Brutus” and “Caesar” and do not contain “Calpurnia”.

An easy way of achieving this task is “to read” through the text. There is a simple unix command for this called “grep”. But what about if:

- the text collection is much *bigger*? Billions or trillions of words.
- you want to search for the word “Romans” when occur somewhere *near to* “countrymen”
- You want to have a *ranked* retrieval of the document to know which are the “best” answers.

Instead of grepping the text, IR systems *index* it.

8.4. Index: example

The matrix below represents whether a certain word occurs (1) or does not occur (0) in a given document.

	d1	d2	d3	d4	d5	d6	...
Antony	1	1	0	0	0	1	
Brutus	1	1	0	1	0	0	
Caesar	1	1	0	1	1	1	
Calpurnia	0	1	0	0	0	0	
...							

Hence, the documents that contain “Brutus” and “Caesar” but do not contain “Calpurnia” are:

$$110100 \text{ and } 110111 \text{ and } 101111 = 100100$$

in words, d1, d4.

8.5. Problems with index

Usually IR is done from a very large document collection (or "corpus"). For instance, assume we have:

- 1 million documents,
- each document is about 1,000 words (2-3 book pages),
- each word is about 6 bytes.
- Then, the document collection is about 6 gigabytes (GB) size.
- With around 500,000 distinct terms

The term-document matrix would be too big: $500K \times 1M$ has half-a-trillion 0's and 1's. They would not fit in a computer's memory.

The 0's could be many (sparsa data). It might be better to record only the things that do occur, that is, the 1's. This is the idea behind "inverted index".

8.6. Inverted Index

Dictionary	*Postings*							
Brutus	1	2	4	11	31	45	173	174
Caesar	1	2	4	5	6	16	57	132
Calpurnia	2	31	54	101				
...								

Terminology Note The dictionary is also called "vocabulary" or "lexicon". Each item in the list of the documents in which the word occur is called "posting". The list is called "postings list" (or "inverted list").

8.7. Indexing and Inverted Index

Formally, *indexing* is the process of associating one or more keywords with each document they are about. The vocabulary used can either be controlled or uncontrolled (closed or open.)

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

The *inverse mapping* captures, for each keyword, the documents it describes:

$$\texttt{Index}^{-1} : kw_i \rightarrow \{doc_j\}$$

Keywords are linguistic atoms – typically words, pieces of words, or phrases – used to characterize the content of a document. They must bridge the gap between the users' characterization of information need (i.e. their queries) and the characterization of the documents'topical focus against which these will be matched.

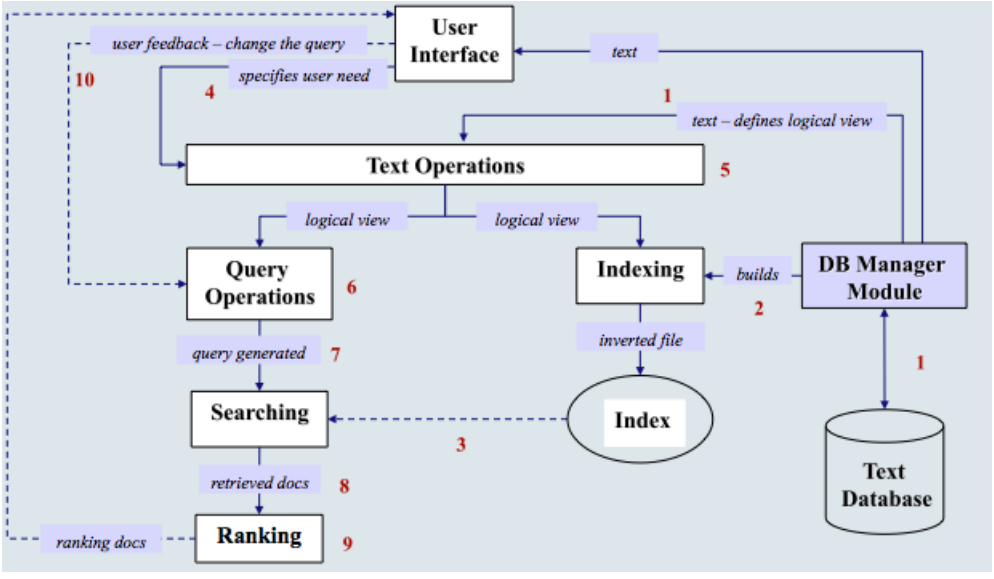
8.8. Index size and space

A document level index needs a value for each ⟨ term, document ⟩ pair. A word-level index needs a value for each word in the collection. We could end up having an inverted index that takes as much space as the text itself.

The elimination of *stop words* helps reducing the space.

Several compression methods have been proposed.

8.9. Retrieval Process



9. Query Languages

The vocabulary used can either be controlled or uncontrolled (closed or open vocabularies.)

The execution (transformation into a formal language) of a query depend on the underling "data model":

- Structured data (data in tables.)
- Semi-structured data (document's structure)
- Unstructured data (free text.)

It also depends on whether we want an exact match between the query terms and the documents (boolean query) or a "full-text retrieval" (non boolean query). Ranking of the result is important in both cases.

9.1. Query processing

Boolean Query mechanics Find X: return all documents containing the term X (X=single words or phrases); complex queries built with boolean operators ('and", "or" and also "but_not").

It's precise: document matches the query or does not match it.

Pattern Matching a set of syntactic features that occur in a text segment; segments that fulfil the pattern specifications –pattern match. Retrieve pieces of text that have some property. Eg. by means of *regular expressions*: general pattern build up by simple strings and operators, e.g. "pro(blem|tein)(s|ε)(0|1|2)*" will match "problem02" and "proteins"

9.2. Query processing: Boolean Model

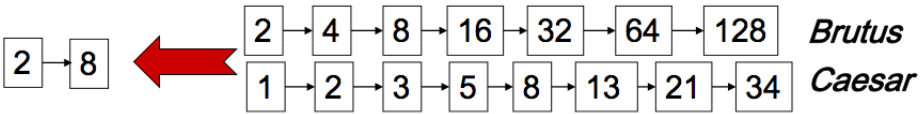
- Consider processing the query:

  Brutus *AND* ***Caesar***

  – Locate ***Brutus*** in the Dictionary;
    - Retrieve its postings.

  – Locate *Caesar* in the Dictionary;
    - Retrieve its postings.

  – "Merge" the two postings:



10. Problem with Boolean search

- Boolean queries often result in either too few (=0) or too many (1000s) results. In Boolean retrieval, it takes a lot of skill to come up with a query that produces a manageable number of hits.
- We wish to be able to *rank* documents, and rank higher the relevant ones.
- Hence, we need to assign a *score to each query-document pair*, say in $[0, 1]$. This score measures how well document and query "match".

11. Ranking

- In Boolean queries a document either matches or does not match the query.
- The order of the returned documents is not specified (often reflects the internal organization of the index.).
- In large collections, the number of (unordered) returned documents is far too big for "human consumption"
- Need to rank the matching documents according to the (estimated) relevance of a document to a query (assigning a score to a (document, query) pair.)
- In actual Digital Libraries, ranking is essential for both: Boolean queries and Full-text (non-boolean) queries.

11.1. Ranking: boolean query

A simple scoring method is to use linear combination of boolean values, by assigning a weight to each field. E.g. the query contains “sorting”:

Score = 0.6*<***sorting*** in Title> +
0.3*<***sorting*** in Abstract> +
0.05*<***sorting*** in Body> +
0.05*<***sorting*** in Boldface>

the document score is obtained by *adding* up the wighted contributions of the fields. The *weights* are evaluated with *Machine Learning* methods, based on a test corpus, a suite of test queries and a set of relevance judgments.

11.2. Full text queries (non boolean)

The query is a sequence of query terms, and it's not practical to consider them as an AND nor as an OR query.

Need to define a method to compute a *similarity measure* between the query and the document. Results will be ranked according to the similarity measures.

We need to represent the document and the query in some mathematical form so to compute their similarity.

The most used format is a vector.

11.3. Document and Query as a binary vector

queries: “eat”; “hot porridge”.

d	Document vectors $\langle w_{d,t} \rangle$									
	col	*day*	*eat*	*hot*	*lot*	*nin*	*old*	*pea*	*por*	*pot*
1	1	0	0	1	0	0	0	1	1	0
2	0	0	0	0	0	0	0	1	1	1
3	0	1	0	0	0	1	1	0	0	0
4	1	0	0	1	0	0	0	0	0	1
5	0	0	0	0	0	0	0	1	1	0
6	0	0	1	0	1	0	0	0	0	0
eat	0	0	1	0	0	0	0	0	0	0
hot porridge	0	0	0	1	0	0	0	0	1	0

11.4. Similarity Measure

Given the pair (“hot porridge”,d1) there similarity measure can be obtained as their *inner product*

$$(0,0,0,1,0,0,0,0,0,1,0) \bullet (1,0,0,1,0,0,0,0,1,1,0) = 2$$

Drawbacks :

- *No* account of *term frequency* in the document (i.e. how many times a term appears in the document)
- *No* account of *term scarcity* (in how many documents the term appears)
- *Long documents* with many terms are favoured

12. Query-document matching scores I

How do we compute the score of a query-document pair?

Let's start with a one-term query.

- If the query term does not occur in the document: score should be 0.
- The more frequent the query term in the document, the higher the score

There are a number of alternatives for doing this.

12.1. Summing up: Weights

Term Frequency As first approximation we can consider the *frequency* of the term in the document.

Inverse Document Frequency Estimate the *rarity* of a term in the whole document collection. (If a term occurs in all the documents of the collection, its IDF is zero.)

TF-IDF A *combination* of the term frequency (TF) and the inverse document frequency (IDF).

Others: http://www.lans.ece.utexas.edu/~strehl/diss/note52.html

13. Recall: Vector Space Model: Document as vectors

Each document is now represented as a real-valued vector of tf-idf weights $\in \mathbb{R}^{|V|}$.

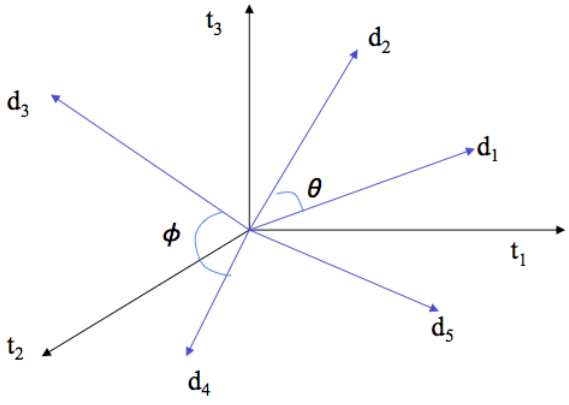
- So we have a $|V|$-dimensional real-valued vector space.
- Terms are axes of the space.
- Documents are points or vectors in this space.
- Very high-dimensional: tens of millions of dimensions when you apply this to web search engines
- Each vector is very sparse - most entries are zero.

$$\vec{d}_J = (t1_J, \ldots, tn_j)$$

$\vec{d}_j$ stands for a certain document j, and ti_J stands for a term that occurs in j at position i of the vector.

13.1. Assumption

Documents that are “close together” in the vector space talk about the same thing.

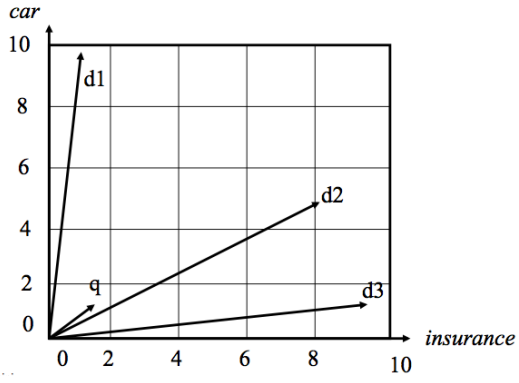


14. Query-document matching scores II

- Key idea 1: Do the same for queries: represent them as vectors in the high-dimensional space, $\vec{q}_k = (t1_k, \ldots, tn_k)$
- Key idea 2: Rank documents according to their proximity to the query

	car	insurance
q =	(1,	1)
d1=	(9,	1)
d2=	(5,	8)
d3=	(1,	9)

Which is the document more similar to *q*?



14.1. Proximity: Angle and Cosine

- Rank documents according to *angle* with query
- Thought experiment: take a document d and append it to itself. Call this document d'. d' is twice as long as d.
- "Semantically" d and d' have the same content.
- The angle between the two documents is 0, corresponding to maximal similarity ...
- ... even though the Euclidean distance (which is based on their length) between the two documents can be quite large.

From angles to cosines The following two notions are equivalent.

- Rank documents according to the *angle* between query and document in *decreasing* order
- Rank documents according to *cosine(query,document)* in *increasing order*

Background

14.2. Example: Length

D1: Anna and Mario eat the apple at home. Anna and Mario have not an apple at home. Anna and Mario go out of home to buy an apple.

D2: Anna and Mario eat the apple at home.

D3: Fabio and Paola went to school and took a lecture.

		anna,	mario,	apple,	home,	fabio,	paola,	school,	lecture
D1	=	(3,	3,	3,	3,	0,	0,	0,	0)
D2	=	(1,	1,	1,	1,	0,	0,	0,	0)
D3	=	(0,	0,	0,	0,	1,	1,	1,	1)

Recall definition: $|\vec{x}| = \sqrt{\sum_{i=1}^{n} x_i^2}$

$$
\begin{aligned}
|\vec{D1}| &= \sqrt{9+9+9+9} &&= \sqrt{36} &&= 6 \\
|\vec{D2}| &= \sqrt{1+1+1+1} &&= \sqrt{4} &&= 2 \\
|\vec{D3}| &= \sqrt{1+1+1+1} &&= \sqrt{4} &&= 2
\end{aligned}
$$

14.3. Example: Cosine Similarity

$$cos(\vec{x},\vec{y}) = \frac{\vec{x}\cdot\vec{y}}{|\vec{x}||\vec{y}|} = \frac{\sum_{i=1}^{n} x_i \times y_i}{\sqrt{\sum_{i=1}^{n} x_i^2} \times \sqrt{\sum_{i=1}^{n} y_i^2}}$$

		anna,	mario,	apple,	home,	fabio,	paola,	school,	lecture
D1	=	(3,	3,	3,	3,	0,	0,	0,	0)
D2	=	(1,	1,	1,	1,	0,	0,	0,	0)
D3	=	(0,	0,	0,	0,	1,	1,	1,	1)

$$cos(\vec{D1},\vec{D2}) = \frac{\vec{D1}\cdot\vec{D2}}{|\vec{D1}| \times |\vec{D2}|} = \frac{(3\times1)+(3\times1)+(3\times1)+(3\times1)}{6\times2} = \frac{12}{12} = 1$$

$$cos(\vec{D1},\vec{D3}) = \frac{\vec{D1}\cdot\vec{D3}}{|\vec{D1}| \times |\vec{D3}|} = \frac{0}{6\times2} = \frac{0}{12} = 0$$

14.4. Example: Query-Document Matching

Query: “best car insurance”. Document: “car insurance auto insurance”.

word	query					document				product
	tf-raw	tf-wght	df	idf	**weight**	tf-raw	tf-wght	weight	**n’lized**	
auto	0	0	5000	2.3	0	1	1	1	0.52	0
best	1	1	50000	1.3	1.3	0	0	0	0	0
car	1	1	10000	2.0	2.0	1	1	1	0.52	1.04
insurance	1	1	1000	3.0	3.0	2	1.3	1.3	0.68	2.04

Key to columns: tf-raw: raw (unweighted) term frequency, tf-wght: logarithmically weighted term frequency, df: document frequency, idf: inverse document frequency, weight: the final weight of the term in the query or document, n’lized: document weights after cosine normalization, product: the product of final query weight and final document weight

Final similarity score between query and document: $\sum_i w_{qi} \cdot w_{di} = 0 + 0 + 1.04 + 2.04 = 3.08$

15. Summary: Ranked retrieval in the vector space model

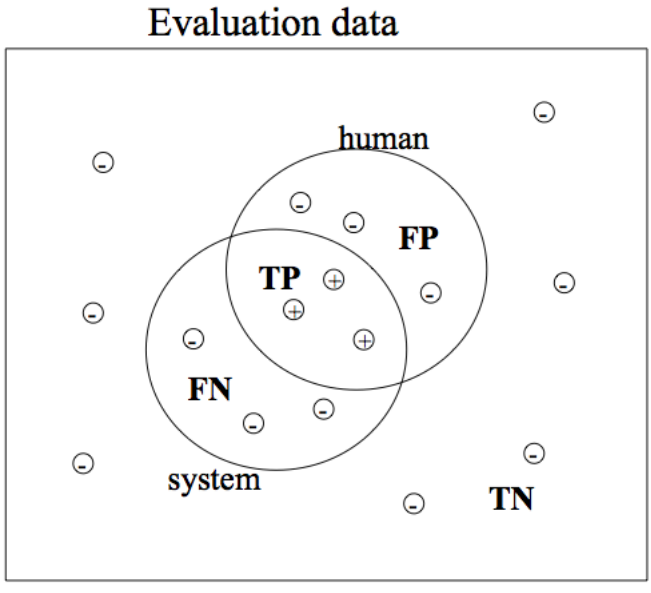
- Represent the query as a weighted tf-idf vector
- Represent each document as a weighted tf-idf vector
- Compute the cosine similarity between the query vector and each document vector
- Rank documents with respect to the query
- Return the top K (e.g., $K = 10$) to the user

16. Evaluation Measures: Contingeny Matrix

A contingency matrix allows to compare the results automatically obtained by an IR system with the correct judgments manually provided by a human on the same dataset.

	Relevant	Not Relevant
Retrieved	True Positive (TP)	False Positive (FP)
Not retrieved	False Negative (FN)	True Negative (TN)

16.1. Evaluation Setting



⊕ Relevant document

⊖ Not relevant document

TP: True Positive
TN: True-Negative
FP: False-Positive
FN: False-Negative

16.2. Evaluation Measure

Accuracy Percentage of documents correctly classified by the system.

$$\frac{\mathbf{TP + TN}}{TP+TN+FP+FN}$$

Error Rate Inverse of accuracy. Percentage of documents wrongly classified by the system

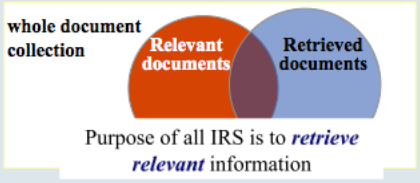
$$\frac{\mathbf{FP+FN}}{TP+TN+FP+FN}$$

But: accuracy is not a good measure for IR.

In IR the number of TN (true negative) is usually much bigger than the number of true positive, false positive and false negative.

Hence, we need a measure that does not consider TNs.

16.3. Precision



The ability of the search to find ***all*** of the relevant documents in the corpus

The ability of the search to retrieve ***top-ranked*** documents that are mostly relevant

Retrieved documents that are relevant

$$precision = \frac{Number\ of\ relevant\ documents\ retrieved}{Total\ number\ of\ documents\ retrieved}$$

16.4. Precision, Recall and F-Measure

- **Precision (P)**: percentage of relevant documents correctly retrieved by the system (TP) with respect to all documents *retrieved by the system* (TP + FP). (*how many of the retrieved books are relevant*?)

$$P = \frac{TP}{TP + \mathbf{FP}}$$

- **Recall (R)**: percentage of relevant documents correctly retrieved by the system (TP) with respect to all documents *relevant for the human* (TP + FN). (*how many of the relevant books have been retrieved*?)

$$R = \frac{TP}{TP + \mathbf{FN}}$$

- **F-Measure**: Combine in a single measure P and R giving a *global estimation of the performance* of an IR system

$$F = \frac{2PR}{R + P}$$

16.5. Exercise

	Relevant	Not-relevant
Retrieved	TP=10	FP =30
Not retrieved	FN= 5	FN=55

Accuracy = (10+55)/100 = 65/100 = 0,65
Error rate = (5+30)/100 = 35/100 = 0,35
Precision = 10/10+30 = 0,25
Recall = 10/10+5 = 0,66
F-Measure = (2 * 0,25 * 0,66)/(0,25+0,66) = 0,38

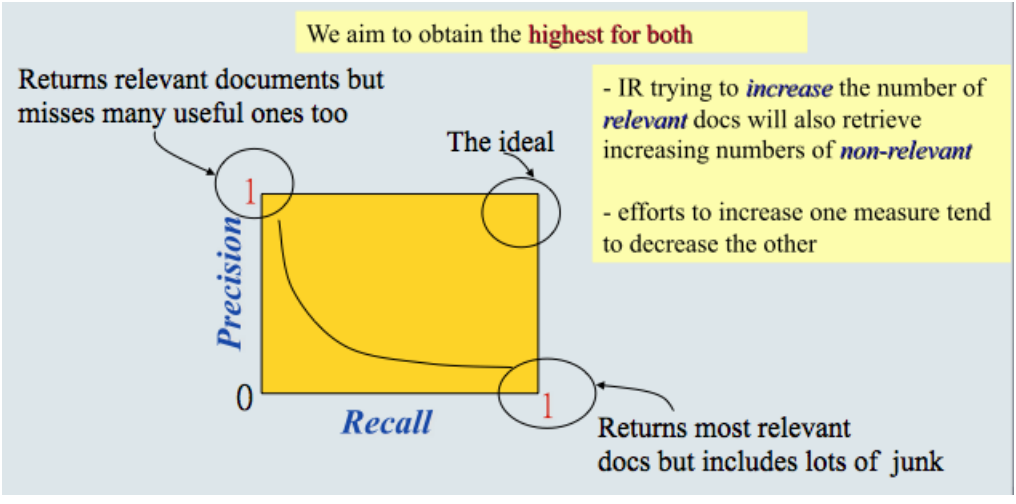
16.6. Problem with Recall

Determining Recall is difficult, the total number of relevant items is sometimes not available.

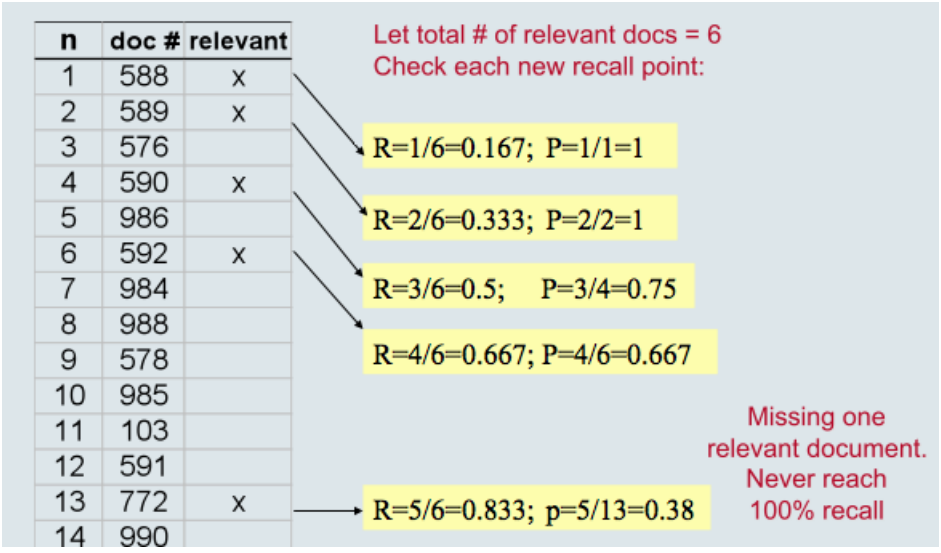
Sample across the database and perform relevance judgment on these items.

Apply different retrieval *algorithms* to the same database for the same query. The aggregate of relevant items is taken as the total relevant set.

16.7. Trade-off between Recall & Precision



16.8. Precision/Recall: at position



n	doc #	relevant
1	588	x
2	589	x
3	576	
4	590	x
5	986	
6	592	x
7	984	
8	988	
9	578	
10	985	
11	103	
12	591	
13	772	x
14	990	