Text Mining

Text Processing 1

structured & destructured textual data, text representation & boolean search model, inverted index, text pre-processing tecniques.

Prof. Gianluca Moro
DISI, University of Bologna, Cesena
name.surname@unibo.it

Structured Data

- E.g. the relational model

Employee	Manager	Salary
Smith	Jones	50000
Chang	Smith	60000
Ivy	Smith	50000

Efficient processing of complex SQL statements, range query, exact match, but limited for textual data

e.g. *Salary < 60000 AND Manager = 'Smith'*

Unstructured Data

- They are data without models or schemas that can semantically described them
- Text data without any predetermined organization are unstructured data
  - 99.99% of Web pages, with minimal expceptions (i.e. semantic web)
  - Emails, forums, blogs, social network posts (FB, Twitter …)
  - Digital Scientific Library (PubMed, Medline, CiteSeer …)
  - Documents Repository of Enterprises, Customer Relationship Management data, Patents …
  - Legal documents: laws, lawsuit docs, governmental docs …

semi-structured data

- they are partially described by some model, such as hierarchical or graphs (i.e. XML, RDF, OWL, RIF ...)
- E.g.

```
dna_sequence_entry:
        source:
                name: Human Cu/Zn SOD1 gene, exon 1.
                organism: Homo sapiens
        dna_sequence: gtaccctg...
        features:
                gene:
                        protein:
                                protein_id: AAB05661.1
```

- There are methods and languages to partially deal with these data types (such as XPath, XQuery ...)
  - Some data sources previously metioned may also contain semi-structued data

The Increasing Importance of unstructured data

- In 90's, according to some studies, people preferred receiving information from other people rather than from information retrieval systems
  - E.g. travels were mostly planned and booked through travel agencies
- In the last decade the result is overturned thanks to the success of Web technology and search engines
  - E.g. in 2004, 92% of internet users thought the Web was a good way to daily retrieve useful information *(Pew Internet Survey, 2004)*

The Increasing Importance of unstructured data (ii)

- **"85%** of all data stored is held in an unstructured format**"**

  Butler Group

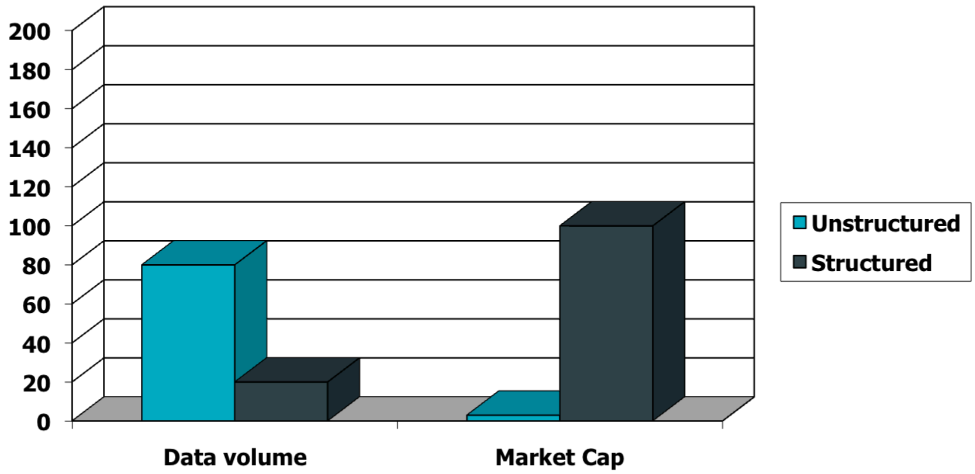
- **"80%** of business is conducted on unstructured information**"**

  Gartner Group

- *"*Unstructured data **doubles** every three months*"*

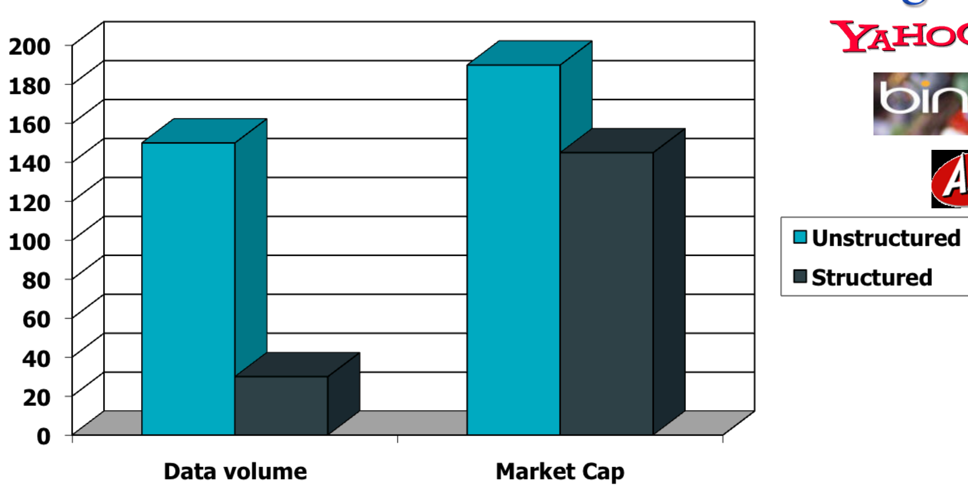
  Gartner Group

Unstructured vs. structured data in 1996



Text Processing

Unstructured vs. structured data in 2009



Some Recent Success Stories

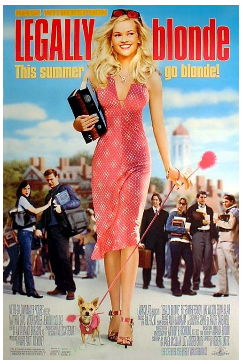
- **Stock Market Predictions via Twitter** – 86% accuracy to predict incr/decr Dow Jones, J. of Computational Science, 2011, J. Bollen et al.
- **Wavii** – App for gathering, classifications and distribution of news – start-up acquired by Google in April 2013 for 30 Milions USD
- **Summly**, iOS App for organizing and summarying news in 400 chars – start-up acquired by Yahoo! in March 2013 for a similar amount
- **Watson** – Born as a *question answering system* in english natural language without topic restrictions – it won the the TV Jeopardy quiz (USA) against the best world human competitors (IBM)
- **Social TV** – merging TV and social networks – several successful startups Yidio, Miso, Getglue, TVzap, Trendrr.tv, IntoNow, *Yahoo!, SKY TV, Nielsen ...*
- **Topsy** – tweet search engine acquired by Twitter for 200 mln USD
- **Publishing –** TwoReads: The right book for each reader, Italian startup
- **DeepMind** – funded by Oxford researchers, acquired by Google for 400 mln $

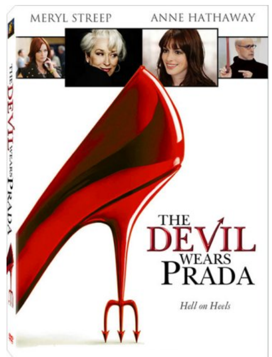
Crime Prediction Systems

- Criminal Reduction Utilising Statistical History (CRUSH)
- Large database of illegal events:
  - committed crimes, tens of features for each illegal event, information on known criminals and their behaviours, tip-off frominformants, video surveillance data
  - weather data (if at night rain, more cars are stolen)
- Goal: predicting crimes
- Experimentation since 2006 at Memphis (USA)
  - the system offers to police the prediction of robberies, vandalism after a sport match, possibility that cars are stolen
  - Experimentation even if Florida and UK
- 31% reduction of general crimes and 15% of violent crimes (Dept. of Criminology and Criminal Justice - University of Memphis)

Which Movies Will Get Success ?

- text mining: predicting successful movie from their scripts
- 1,000,000 US $ prize to whom predict which movies each person will go to see
  - www.netflixprize.com
  - first edition won in 2009. A new edition has been funded.





likes?
⇒

Watson

- Started in 2007 (by David Ferrucci - IBM)
- Goal:
  - recognizing english natural language, exctracting knowledge from milions of documents and replying more quickly than humans
- Successful test with TV quiz Jeopardy
  - finding the right question to supplied answers against human competors
  - beaten champions (71% of right questions)
  - Solutions in less than 3 seconds (1 milion of books analyzed per second)

- 2880 POWER7 cores
- 10 Gb Ethernet network
- 15 Terabytes of memory
- 20 Terabytes of disk
- Can operate at 80 Teraflops
- Runs IBM DeepQA software
- UIMA & Hadoop open source
- Linux
- 10 racks servers

Text Mining (TM)

- *Knowledge extraction from large unstructured textual data*
- **Typical Text Mining Task:**
  - Text Classification, Clustering
    - Classification, Clustering of documents, usually by topic
  - Text Extraction & Summarization
    - Extracting entities, such as persons, companies, brands, dates, events, places and generation of document abstracts
  - Sentiment & Opinion mining
    - Classifying reviews, posts, emails etc. by opinion orientation
  - Question answering
    - Supplying answers to questions asked in natural languages
  - Information Retrieval
    - finding relevant documents wrt searches

What's Information Retrieval (IR) ?

- Methods and algorithms to search for relevant docs wrt to user queries in repositories of unstructured docs
- IR example with databases
  - SQL: Select * from CUSTOMERS where NAME like '%Business%Intelligence%'
  - but generally documents are not organized in relational db
- IR should reply to more advanced queries, such as
  - Retrieving docs containing terms *'Information'* adjacent to *'Retrieval'* or *'Relevance'* but without *'Protocol'*

Information Retrieval & Text Mining

- Data selection in data mining is the first step after defining the mining goals
- In data collection it is important to check and improve the quality of data
  - goals can be modified according to the quality of data
- Information Retrieval & Text Mining:
  - IR offers efficient methods for **representation** & **selection** of unstructured data which can be useful for Text Mining
  - Text Mining offers techniques to improve complex IR searches
    - e.g. searching docs similar to one or more docs, within flat or hierarchical catalogs organized by topics

Representation of Documents

documents

terms

	Antonio and Cleopatra	Giulio Cesare	La Tempesta	Amleto	Otello	Macbeth
Antonio	1	1	0	0	0	1
Bruto	1	1	0	1	0	0
Cesare	1	1	0	1	1	1
Calpurnia	0	1	0	0	0	0
Cleopatra	1	0	0	0	0	0
mercy	1	0	1	1	1	1
worser	1	0	1	1	1	0

finding docs containing bruto, cesare but not calpurnia

Bruto** AND **Cesare** BUT NOT **Calpurnia

1 if it contains the word, else 0

Binary Vectors

- Vectors 0/1 for each term and document
- To resolve the previous query
  - Let's select the vector of ***Bruto, Cesare*** AND the complement of ***Calpurnia***
  - ➔ AND *bitwise* among vectors: in modern CPU it is more efficient than computations among numerical data
- Result
  - 110100
  - 110111
  - 101111
  - 100100

	Antonio and Cleopatra	Giulio Cesare	La Tempesta	Amleto	Otello	Macbeth
Antonio	1	1	0	0	0	1
Bruto	1	1	0	1	0	0
Cesare	1	1	0	1	1	1
Calpurnia	0	1	0	0	0	0
Cleopatra	1	0	0	0	0	0
mercy	1	0	1	1	1	1
worser	1	0	1	1	1	0

Boolean Queries: Exact match

- In the boolean retrieval model the query expressive power is based on boolean propositions
  - where query terms are combined with *AND, OR* and *NOT*
    - Each documents is considered as a sete of terms
    - the match is rigid: each doc satisfies or not satisfies the search condition
  - it is the simplest information retrieval model
- Employed in most popular IR tool for 30 years:
  - Online scientific library (e.g. sciencedirect.com)
- Often it's a predominant solution even in modern software:
  - Web Browsers, Email Clients, Editors, Mac OS X Spotlight ..

Limits of the Vector Representation

- Let's have *N* = 1 milion of documents, each with about 1000 terms
  - On average 6 bytes/term with spaces and punctuations
  - this amount to 6GB of documents
- Let's *M* = 500K *distinct terms* in the milion of docs
- 500K x 1M is a matrix with 500 billions 0 and 1 but with at most 1 billion of value 1
  - it's sparse matrix (1000 terms * 1M docs)
- which is the best representation ?
  - the one that stores only the position of value 1 (less frequent)

A Solution: Inverted Index

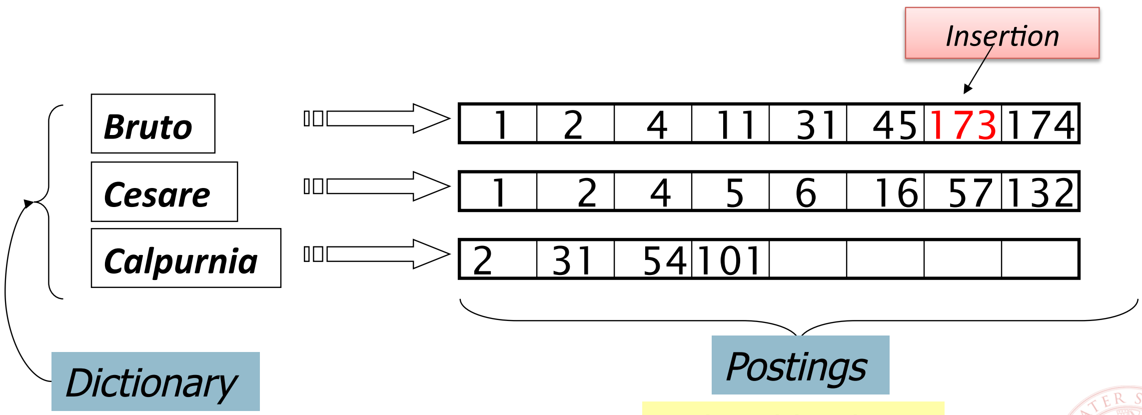
- For each term *t*, it stores a list of all docs containing *t*
  - Each doc is identified by a **docID** (a serial number)
- May we use fixed size arrays ?

Bruto	1	2	4	11	31	45	173	174
Cesare	1	2	4	5	6	16	57	132
Calpurnia	2	31	54	101				

what if we add the term ***Cesare*** to the document 14 ?

Inverted Index with Dynamic Structures

- We need to add terms specifying the insert position
  - The postings, i.e. list, of docID are stored on disk
  - The dictionary is stored in RAM as it is smaller than postings

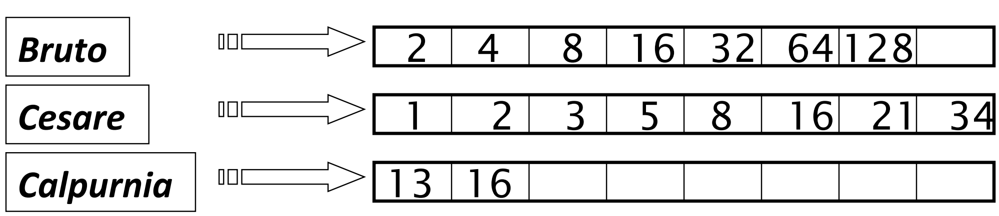


Boolean Query with Inverted Index

- Let's consider a conjunction query of *3* terms

Query: ***Bruto*** *AND* ***Calpurnia*** *AND* ***Cesare***

- We select the postings of the 3 terms and apply the *AND* to the 3 lists *(i.e. intersection of lists)*



- How can we make the query processing efficient ?

Query Optimization: Example

- Visiting the lists in incresing order of frequency, i.e. the num. of docs containing the term
  - *Starting from the shorter list for better efficiency*

This the reason why we stored the term frequency

Calpurnia	13	16						
Bruto	2	4	8	16	32	64	128	
Cesare	1	2	3	5	8	16	21	34

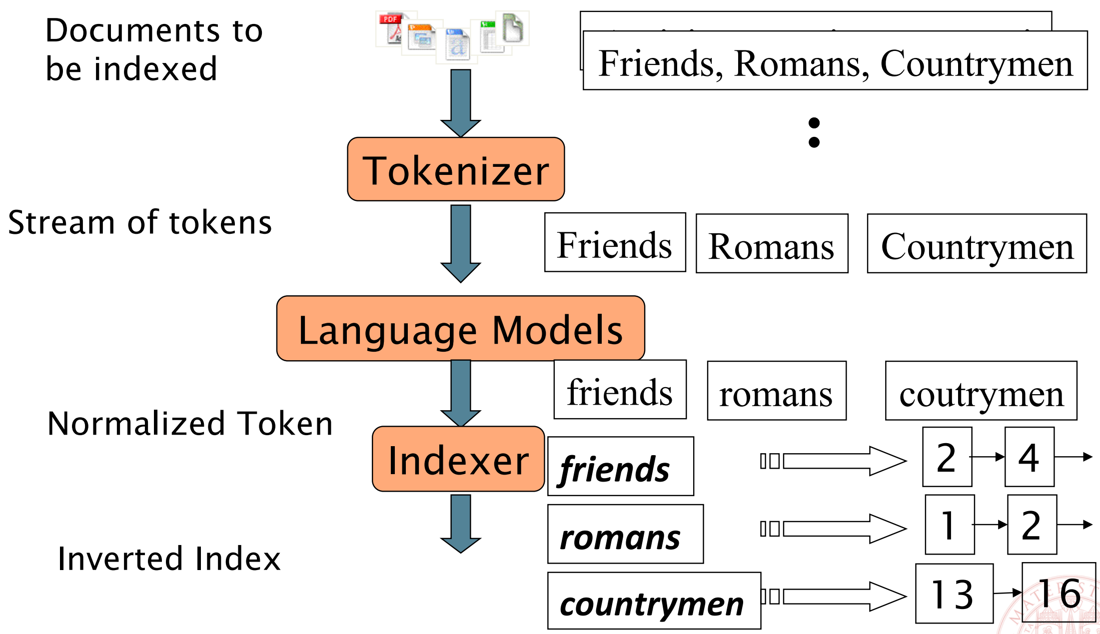
Executing the query as (***Calpurnia*** *AND* ***Bruto)*** *AND* ***Cesare***

Text Processing

Arbitrary Boolean Queries

- How to process more complex queries ?
- *E.g.* ***(Bruto*** *OR* ***Cesare)*** *AND NOT* ***(Antonio*** *OR* ***Cleopatra)***
- Naïve method (limits due to memory problems):
  - L1 = Lista(Bruto) U Lista(Cesare)
  - L2 = Lista(Antonio) U Lista(Cleopatra)
  - L3 = L1 – L2
- A solution based on query rewriting using boolean logics:
  - ***(Bruto*** *AND NOT* ***(Antonio*** *OR* ***Cleopatra))*** *OR* ***(Cesare*** *AND NOT* ***(Antonio*** *OR* ***Cleopatra))***
  - ***(Bruto*** *AND NOT* ***Antonio*** *AND NOT* ***Cleopatra)*** *OR* ***(Cesare*** *AND NOT* ***Antonio*** *AND NOT* ***Cleopatra)***
  - the union produces usually lists that are longer than intersection, therefore it is **more efficient** usually to **process intersection before union**

Building the Inverted Index (i)



Inverted Index: RDBMS Vs NoSQL DB

- Generalized Inverted Index (GIN)
  - it contains an index entry for each term, with a compressed list of matching locations
- Generalized Search Tree (GiST)
  - generalization of several traditional indexes, like B-Tree, with no limitation in text size, moreover it allows using arbitrary predicates
- When using GIN or GiST index ?
  - GIN index is best for static data because lookups are faster, while GiST index is best for dynamic data as is faster to update under 100K terms
- DB Management Systems (DBMSs) are equipped with such indexes for speeding up full text searches
  - NoSQL DB born for text manipulation, but less efficient than Relational
  - Relational (RDBMS) are incorporating efficient text operations, for instance now PostgreSQL has GIN, GiST, JSON data type like NoSQL DBMS

Text Tokenization

- A **token** is a sequence of chars è followed by a delimiter, which is one or more chars, usually the delimiter is the space
  - Text: "***Friends, Romans, Countrymen lend me your ears!***"
  - Token list: ***Friends Romans Countrymen lend me your ears***
- It's not only a mere syntactic operations, example:
  - ***Finland's capital*** → ***Finland*** ? ***Finlands*** ? ***Finland's*** ?
  - ***Hewlett-Packard*** → ***Hewlett*** and ***Packard*** 2 tokens ?
  - ***state-of-the-art*** ***co-author*** dividing always the terms ?
  - ***lowercase***, ***lower-case***, ***lower case*** several forms for the same concept ***San Francisco***: 1 or 2 tokens ?
  - date ***3/12/91*** ***Mar. 12, 1991*** ***12/3/91*** ***55 A.C.***
  - num. and codes with spaces ***(800) 234-2333*** ***connection error 25401***
- Indexing meta-data: creation date, format, dimension etc.

Token: Other Language Problems

- French
  - ***L'ensemble*** → 1 or 2 tokens ?
    - ***L*** ? ***L'*** ? ***Le*** ? it should match with ***un ensemble***
      - until 2003 Google did not perform this match
- German: long composed nouns without separations
  - ***Lebensversicherungsgesellschaftsangestellter***
  - 'employee of a life insurance company'
  - the efficacy of answers increases of 15% appying splitting techniques
- Chinese and Japanese don't have space among words
  - 莎拉波娃现在居住在美国东南部的佛罗里达。
  - it's difficult to guarantee an unique tokenization
- Arabic and Hebrew are written from right to left, but sometimes is the opposite, for instance with numbers

استقلت الجزائر في سنة 1962 بعد 132 عاما من الاحتلال الفرنسي.

Stop Words

- Very frequent terms but with a scarse semantic content
  - there are **general stop word lists**, independent from any domain
  - e.g. conjunctions, prepositions, articols ...: *the, a, and, to, be ...*
  - the 30 more frequent words are about 30% of tokens in any doc
  - **stop words for specific domain**: they are terms that do not contribute to distinguish documents in the domain, e.g. *heart* in *cardiovascular docs*
- Elimination or maintenance of stop words:
  - they are **a priori ignorated** for most of text mining tasks
  - **sometimes are included**, for instance in Natural Language Processing for the semantic analysis of sentences
  - "flights to London"
  - used in exact match searches (e.g. works, "Let it be" etc.)
  - there are efficient techniques of compression and searches efficienti to deal with stop words

Word Normalization

- Normalization means that different instances of a word are reduced to the same form
  - e.g. ***U.S.A.*** is equivalent to ***USA***, thus they should have the same form
- A term is a normalized word corresponding to a dictionary entry in the information retrieval system or in the respository
- A solution is the definition of equivalence classes of terms:
  - elimination of punctuactions and hyphens
    - ***U.S.A., USA → USA (the first two forms are transformed in the third term)***
    - ***anti-discriminatory, antidiscriminatory → antidiscriminatory***
- Accented words are transformed to terms without accents
  - e.g. german: ***Tuebingen, Tübingen, Tubingen → Tubingen***
  - e.g., french ***résumé*** vs. ***resume***
  - in searches of tweets, web pages etc., users tend not to use accents

Normalization: Language Identification and Expansion

- Tokenization and normalization depend from the language, therefore both are related to the automatic language recognition

"mit" is german ?

Morgen will ich in $\boxed{\textit{MIT}}$ …

- Search expansion: as an alternative to equivalence classes
  - Example:
  - searching for: ***window*** expanded to: ***window, windows***
  - searching for: ***windows*** expanded to: ***Windows, windows, window***
  - advanced methods use expansion on the basis of the user profile built from preceding searches, shoppings, visited web pages, whatsapp, facebook and twitter posts etc.
- Usually it leads to more expressive power but is less efficient

Uppercase and Lowercase

- in all char codifications such as ASCII, UTF ... uppercase and lowercase chars are absolutly different
  - case sensitive: editors, programming languages, operating systems etc.
- in natural language this strong difference is instead weaker
  - there are rules for uppercase/lowercase; the same word with the same semantic can be written in different forms and vice versa
  - the meaning depends from the context
- Usually all uppercase chars are reduced to lowercase
  - exceptions: what's about uppercase not at the sentence beginning ?
    - e.g., ***General Motors, Fed*** vs. ***fed, SAIL*** vs. ***sail***
  - often is better to reduce all to lowercase as the users, in short texts, don't care much about using uppercase chars
- An example of a Google search (now fixed)
  - searching for C.A.T. – the first result was **"cat"** and not **Caterpillar Inc.**

Synonyms and phonetic equivalence: Soundex

- Treatment of synonyms and homonyms with class of equivalence
  - E.g., ***car*** = ***automobile*** ***color*** = ***colour***
  - the equivalence is applied for the rewriting of terms
    - docs with ***automobile***, it's indexed as ***car-automobile*** (and vice versa)
  - or by expanding each search
    - if the search contains ***automobile***, lookups also for ***car***
- Term equivalence by phonetic heuristics
  - developed by international police department to unify wanted criminal names differently registered in different countries
  - e.g. mispelling: *Hermann* and *Herman*, *Rupert* and *Robert*
  - idea: generating for each term a phonetic hash so that terms with a similar sound have the same hash code – **soundex algorithms**
  - the same transformation is applied to each search string

Lemmatization

- In text we use different inflected forms of words according to the language grammar rules
  - e.g. *organize*, *organizes*, *organizing* *car, cars, car's, cars'*
- Lemmatization is the process of reducing the inflected forms of a word to a single dictionary term called **lemma**
  - *democratic*, *democratization* -> *democracy*
  - in this way searching for words in a given inflected form, can also return documents with the same words in different inflected forms
- Lemmatization
  - the lemma of a word is also called the base form
  - E.g., *the boy's cars are different colors* → *the boy car be different color*
  - differently from other methods, the lemma is always a word belonging to the dictionary (it is employed the morphological analysis)

Stemming

- Reduction of terms their "root" called theme
  - In general this reduction does not correspond to the lemma
  - the goal is to reduce co-related words to the same root
  - e.g., **andare, andai, andò** reduced to **and**, even if it is not a valid morphological form of the word
- It is based on heuristics that cut suffixes
  - the reduction rules depend from the language
  - e.g., ***automate(s), automatic, automation*** -> ***automat***.
  - There are several Stemming algorithms

for example compressed and compression are both accepted as equivalent to compress.	for exampl compress and compress ar both accept as equival to compress

Stemming: Porter Algorithm

- The most popular stemming algorithm for the English language
  - empiric results show that it is as effective as other more complex algorithms – it includes 60 suffixes
- It contains rules and 5 progressive reduction phases
  - each phase, sequentially applied, contains a set of rules
  - e.g. 'hopefulness' → 'hopeful' → 'hope';
  - general rule: in each phase, and for each word, are applied the rules that maximize the suffix length to be removed
- Typical Porter rules (phase 1)
  - *sses* → *ss* *ies* → *i* *ational* → *ate* *tional* → *tion*
  - Rules are sensitive to word lengths (i.e. dependence from syllables)
  - example: the rule (*measure* > 1) EMENT →
    - *replacement* → *replac* *cement* → *cement* ~~*cement -> c*~~

Other Stemming Algorithm

- Lovins stemmer
  - the longest suffix removal in a single step
  - it has a dictionary with 294 suffixes, each of them is associated with several exceptions
- In short:
  - given an inflected word, if it ends with one of the existing suffixes ***s*** in its dictionary
    - if the word is not an exception wrt the suffix ***s*** then it removes ***s***
  - e.g. suffix '-ation' and its list of exception words containing 'nation'
  - the idea is that any exception is a word that contains only apparently the suffix and not as a real morphological costituent
- modest benefits in IR and not negligible computational load

Performances in Different Languages

- How much normalization, stemming etc. help in IR ?
  - **Enghlish**: results are not always consistent. Given the same searches, they generally increase retrieved documents, both relevant and unrelevant
    - operate (*dentistry*) ⇒ oper
    - operational (*research*) ⇒ oper
    - operating (*systems*) ⇒ oper
  - e.g. the following searches lose precision
    - operational AND **research** operating AND **system** operative AND **dentistry**
    - *operating AND **system*** returns also sentences with *operate AND system*
  - Much more benefits in languages such as Spanish, German, Finnish etc.
    - 30% improvement for Finnish
- They are dependent from languages and applications
- they are part of the processing and indexing of textual data
- available in commercial and open source tools (WEKA, R, RapidMiner ...)

Problems with Boolean Search Model

- We examined the boolean search model and text processing tecniques, the latter are ortogonal to search models
- Boolean search, **cons:** often generate extreme results, either no result or too large results
  - E.g. Q1: "*standard user dlink 650*" → 200,000 answers
  - E.g. Q2: "*standard user dlink 650 no card found*" 0 answer
  - Boolean search model are good for expert users who know the text set and are capable of formulating precise searches
- **Further cons**: expressing complex boolean searches is not easy for most of people
  - users dislike to navigate among thousands of results which **CANNOT BE ORDERED BY RELEVANCE**
- **pros:** very efficient processing algorithms for boolean searches

Text Mining

Text Processing 2

ranking models, bag of words, term weightings, similarity metrics, Wordnet, evaluation methods

Prof. Gianluca Moro
DISI, University of Bologna, Cesena
name.surname@unibo.it

Beyond the Boolean Search Model

- Limists of boolean search model
  - results are not ordered/ranked, too much or too few results, only exact match, not proximity search
- Proximity Searches: *find* ***Gates*** near ***Microsoft***
  - this requires to index terms and their positions in docs also
- Semi-structured searches:
  - *find documents where* (*author* = ***Jim Gray***) *AND* (*abstract* contains ***transaction***)
- Searches by frequency:
  - *find docs with at least 3 times* "Text" AND *5* "Mining"
- Wildcards, search by similarity ...

Example: WestLaw http://www.westlaw.com/

- Since 1975 the biggest commercial system for legal searches (ranking added in 1992)
  - Milion of searches every day
- Tens of Terabytes of data; +700,000 users
- Almost all users use boolean searches
- Example of information need and the search:
  - *"Information on legal theories in preventing the disclosure of trade secrets by employees formerly employed by a competing company"*.
  - "trade secret" /s disclos! /s prevent /s employe!
    - ! = wildcard, /S = in the same sentence, space = OR, "" consecutive words

Scoring in Ranked Retrieval Models

- The goal is to return a list of documents in an order as relevant as possibile wrt the user query
- Several methods and algorithms to rank each document in the interval [0, 1]
  - It's should be a mesaure of how much the doc satisfy the query (i.e. search), generally 1 means max relevance
- Results with a large number of answers are no longer a problem being ordered by relevance
  - Only query terms also in the doc increase the doc rank
- The more the term is frequent in the document, the more the score of the document should be high

Jaccard Coefficient

- it measure the grade of intersection of 2 sets *A* and *B*
- jaccard$(A,B) = |A \cap B| / |A \cup B|$
- jaccard$(A,A) = 1$; jaccard$(A,B) = 0$ if $A \cap B = 0$
- The result is between 0 and 1
- The distance $J_d(A,B)$ = 1-jaccard(A,B) is a metric:
  - For each x, y, z in X, it satisfies the following conditions:
  - $d(x, y) \geq 0$ (not-negativity)
  - $d(x, y) = 0$ se e solo se $x = y$
  - $d(x, y) = d(y, x)$ (simmetry)
  - $d(x, z) \leq d(x, y) + d(y, z)$ (triangular inequality)

Jaccard Distance: Example

- Query: *ides of march*
- Doc1: *Cesare died in march*
- Doc2: *the long march*
- J_d (Query,Doc1) = 1 – 1/6 = 5/6
- J_d (Query,Doc2) = 1 – 1/5 = 4/5 *(most relevant)*
- It does not rely the frequency of terms
- Rare terms are most informative than frequent terms, but Jaccard ignores this aspect
- Jaccard and trigrams are suited for short texts
  - products' similarity by their descriptions

Binary Matrix of terms-documents

	Antonio and Cleopatra	Giulio Cesare	La Tempesta	Amleto	Otello	Macbeth
Antonio	1	1	0	0	0	1
Bruto	1	1	0	1	0	0
Cesare	1	1	0	1	1	1
Calpurnia	0	1	0	0	0	0
Cleopatra	1	0	0	0	0	0
mercy	1	0	1	1	1	1
worser	1	0	1	1	1	0

Each document is a binary vector $\in \{0,1\}^{|V|}$

Term-doc Matrix with frequency

- Let's consider also the number of occurences of a terms in each documents
- Each document is a vector of vaues in $\mathbb{N}^v$

	Antonio and Cleopatra	Giulio Cesare	La Tempesta	Amleto	Otello	Macbeth
Antonio	157	73	0	0	0	0
Bruto	4	157	0	1	0	0
Cesare	232	227	0	2	1	1
Calpurnia	0	10	0	0	0	0
Cleopatra	57	0	0	0	0	0
mercy	2	0	3	5	5	1
worser	2	0	1	1	1	0

Bag of words Model

- the previous model the vector representation ignores the word order in each doc
- *John is faster than Mary* and *Mary is fater than John* generate identical vectors
- This known as the bag of words model
- It's a backward step wrt other solutions
  - such as the positional index that instead distinguishes these two documents
  - or paragraph vectors (PV) in deep learning
  - of course there are more complex bag of words model that includes also the word positions

Frequency of terms

- the term frequency $\text{tf}_{t,d}$ of a term *t* in a doc *d* is the number of occurences of *t* in *d*
- the **tf** value should be used to determine which documents are more relevant for a given query
- However it is inappropriate to directly use such a computed value:
  - a document with 10 occurences of a term is more relevant than a doc with only 1 occurence
  - but not 10 times more relevant
- The relevance does not grow linearly with the frequency of terms

Logarithmic Weight of Frequency

- The frequency with log weight of a term *t* in *d* is

$$w_{t,d} = \begin{cases} 1 + \log_{10} \text{tf}_{t,d}, & if\ \text{tf}_{t,d} > 0, \text{tf}_{t,d} \in \mathbf{N} \\ 0, & else \end{cases}$$

- 0 → 0, 1 → 1, 2 → 1.3, 10 → 2, 1000 → 4, etc.
- The relevance of doc *d* for a given query *q*, is the sum of log frequency over the terms *t* in both *q* and *d*:

$$\sum_{t \in q \cap d} w_{t,d}$$

- The relevance is zero when query and document do not share any term

Relevance of Terms

- Rare terms are generally more important than frequent terms (attention to rare mispelling terms)
  - E.g. articles, prepositions, conjunctions are so frequent that as we know are stop words often totally ingnored
- however the **tf** gives instead more importance to much more frequent terms
- The more a query term is rare, the more a doc with such term has higher probability to be relevant
  - **but rare respect to what ?**
- We should reweight the **tf** to give more importance to less frequent terms in the doc repository

Inverse Document Frequency (IDF)

- $\mathbf{df}_t$ is the number of documents in the corpus with the term *t*
  - $\mathbf{df}_t$ is an inverse misure of the info about the term ***t*** *in* ***d***
  - $\mathbf{df}_t \leq N$ = *number of docs in the data set*
- It is called **idf** = inverse document frequency of *t* as

$$\mathrm{idf}_t = \log_{10}(N/\mathrm{df}_t)$$

  - analogously to **tf** we use log (N/df_t) instead of N/df_t because the relevance is not inversely proportional wrt the frequency
  - **idf** does not take into account the repetition of **t** in each document of the corpus

Example: **idf** with 1 milion documents

$$\text{idf}_t = \log_{10}(N/\text{df}_t)$$

termine	df_t	idf_t
calpurnia	1	6
animal	100	4
sunday	1,000	3
fly	10,000	2
under	100,000	1
the	1,000,000	0

Reuters: text set with more than 800000 news

termine	df_t	idf_t
auto	6723	2.08
car	18165	1.65
insurance	19241	1.62
best	25235	1.50
the	806791	0.00

the **idf** of each terms **t** depends from the corpus

Solution: Combining TF and IDF

- it is called TF-IDF definide as the following product

$$w_{t,d} = \log(1 + \text{tf}_{t,d}) \times \log(N / \text{df}_t)$$

  - it's a standard in IR since '70s to compute the relevance of each term within a document according to the corpus
  - synonyms in literature: tf.idf, TF-IDF, tf x idf
- Conclusions: *the term relevance in any doc increases with the number of occurences in the doc …*
- *… and decreases with the number of occurences of the term in all documents of the corpus*

Relevance of a Doc *d* wrt a Query *q*

$$\text{Rank}(q,d) = \sum_{t \in q \cap d} \text{tf-idf}_{t,d}$$

- Several versions have been also defined:
  - **tf** computed with and withou log
  - weighting of query terms
  - **tfc** that takes into account even the length of docs
  - **ltc** that better smoothes the high difference in frequencies
  - **IDF** modified by computing the entropy of **t**
  - other variants that deal with not independent terms

TF-IDF: documents as vectors of reals

	Antonio e Cleopatra	Giulio Cesare	La Tempesta	Amleto	Otello	Macbeth
Antonio	5,25	3,18	0	0	0	0,35
Bruto	1,21	6,1	0	1	0	0
Cesare	8,59	2,54	0	1,51	0,25	0
Calpurnia	0	1,54	0	0	0	0
Cleopatra	2,85	0	0	0	0	0
mercy	1,51	0	1,9	0,12	5,25	0,88
worser	1,37	0	0,11	4,15	0,25	1,95

- It's a muldi-dimensional space
  - each term is an axis and each doc a point in a such space
  - the doc coordinates are its TD-IDF values
- High dimensionality and sparsity
  - typical milions of dimensions when it is adopted such representation by web search engines

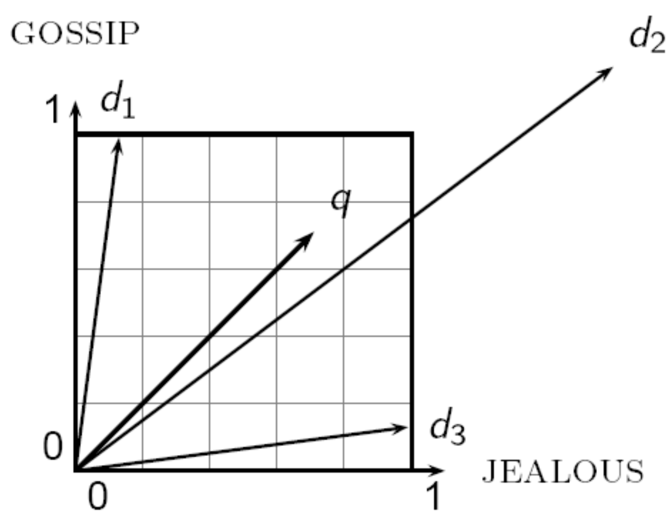
Query as vector in the doc space

- 1: let's represent also the query as a vector
- 2: the relevance of a doc wrt the query is based on their proximity in the space
- proximity = vector similarity, i.e. of documents
- We need a ranking method to overcome the boolean model where each match can be only true or false
- How computing the distance among vectors ?
  - by computing the euclidean distance among their extremes … ??
  - … but this distance depends on the length of vectors …..

Problems with Euclidean Distance

the euclidean distance between $\vec{q}$ e $\vec{d_2}$ is large even if the term distribution in both the query and the doc are very similar

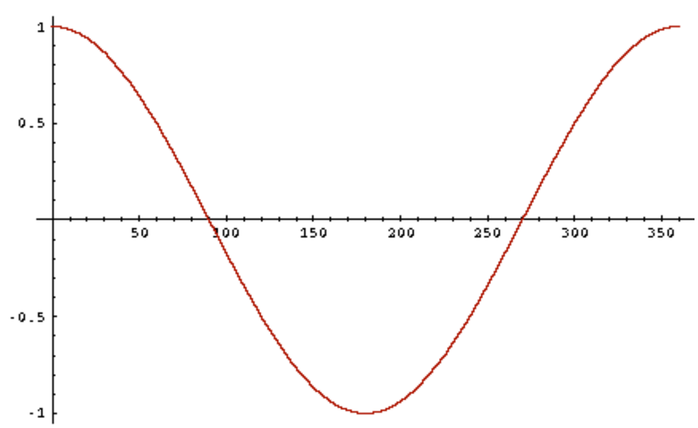
in $\vec{d_2}$ each component value of the vector is about twice of each component value of $\vec{q}$



Angles instead of Distances

- Experiment: let's add to a doc *d* the document itself forming a new doc *d'*
- d e d' are semantically identical having the same content …
- … but their euclidean distance is large
- Their angle instead is ZERO, that is maximum similarity
- We need a monotone similarity function that returns a value within [0, 1]
  - 0 means the maximum angle of 90° and 1 is angle zero, namely maximum similarity

From Angles to the Cosine function



- The cosine function is monotone decreasing in [0°, 180°]
- How can we use it for computing the relevance between each couple query and doc ?

Cosine (query, document)

$\cos(\vec{q},\vec{d})$ is the cosine of the angle between vector $\vec{q}$ e $\vec{d}$

scalar product -> a real number

$$\cos(\vec{q},\vec{d}) = \frac{\vec{q} \bullet \vec{d}}{\|\vec{q}\|\|\vec{d}\|} \qquad \vec{q} \bullet \vec{d} = \sum_{i=1}^{k} q_i \times d_i$$

k = terms of the vector

$$\|\vec{q}\| = \sqrt{\sum_{i=1}^{k} q_i^2}$$

Norm L_2 of the vector = vector length (scalar result)

q_i can be the tf-idf of the term *i* in the query
d_i can be the tf-idf of the term *i* in the document

Vector Normalization

- A vector can be normalized dividing each component by its norm L_2

$$\|\vec{x}\|_2 = \sqrt{\sum_i x_i^2}$$

$$\vec{x}^u = \frac{\vec{x}}{\|\vec{x}\|_2}$$

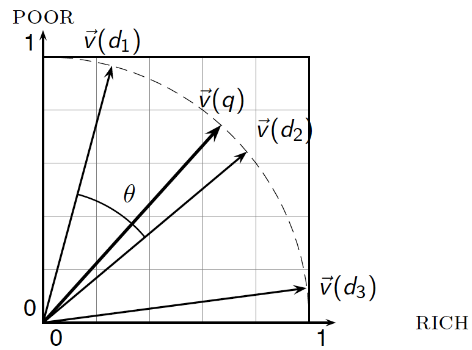
- the result vector is unitary and the impact on doc d and d' (i.e. d added to itself as previously mentioned) become equals
  - short and long documents in this way can be comparable

Cosine with Normalized Vector

- The similarity based on the cosine of vectors q and d normalized is simply their scalar product:

$$\cos(\vec{q}^u, \vec{d}^u) =$$

$$= \vec{q}^u \bullet \vec{d}^u = \sum_{i=1}^{k} q_i^u d_i^u$$



Cosine Similarity among 3 Documents

- How are similar these works ?
- SS: *Sense and Sensibility*
- PP: *Pride and Prejudice*
- WH: *Wuthering Height*

term	SS	PP	WH
affection	115	58	20
jealous	10	7	11
gossip	2	0	6
stormy	0	0	38

Frequency of terms

Similarity among 3 Documents (ii)

Terms with Log TF

term	SS	PP	WH
affection	3.06	2.76	2.30
jealous	2.00	1.85	2.04
gossip	1.30	0	1.78
stormy	0	0	2.58

After normalization

term	SS	PP	WH
affection	0.789	0.832	0.524
jealous	0.515	0.555	0.465
gossip	0.335	0	0.405
stormy	0	0	0.588

$$\text{Cos(SS,PP)} \approx 0.789 \times 0.832 + 0.515 \times 0.555 + 0.335 \times 0.0 + 0.0 \times 0.0 \approx \mathbf{0.94}$$

$$\text{Cos(SS,WH)} \approx \mathbf{0.79}$$

$$\text{Cos(PP,WH)} \approx \mathbf{0.69}$$

Sense and Sensibility is more similar to **Pride and Prejudice** than to **CWuthering Height** ...... *is reasonable, why ?*

PageRank: Relevance from Links (i)

- the basis algorithm of Google
  - L. Page, S. Brin, R. Motwani, T. Winograd (1999) *The PageRank Citation Ranking: Bringing Order to the Web.* Technical Report. Stanford InfoLab.
  - measure the **relevance** of web pages from the **num. of entry links** *rather than only their contents*
  - **entry links are in turn weighted according to the PageRank** of pages to which links belong to
    - e.g. *page* ***C*** *has a single entry link, but its rank is high thanks to the rank of* ***B***

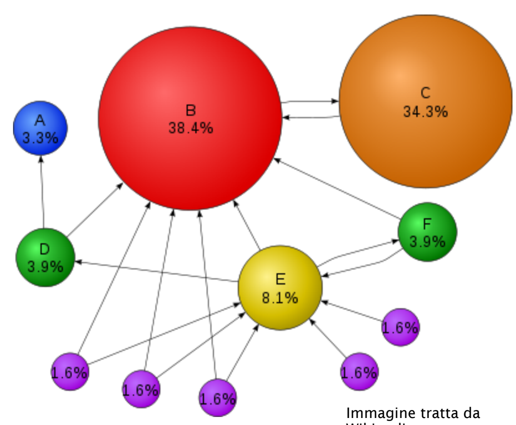
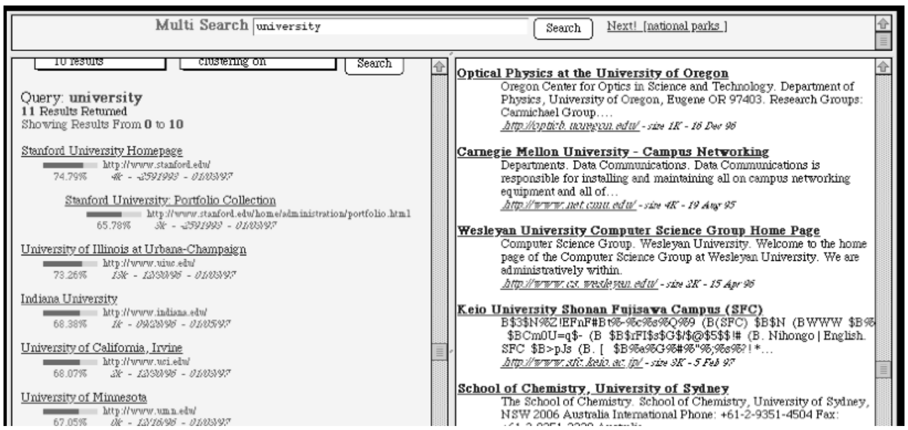


Immagine tratta da Wikipedia

- It extracts from links among Web pages the implicit **knowledge** of pages' importance according to the relevance given by authors that linked them
  - the knowledge is a probability distribution funded on **random walk**

PageRank: Relevance from Links (ii)

- roughly speaking, the search process select Web pages according to their contents using standard and advanced techniques of information retrieval
- and results are ordered according to the rank computed by PageRank
- E.g.: the search of *"University"* - **Google** vs **Altavista** (1999): Google returns top level university, while Altavista irrelevant pages containing *"University"*



L. Page, et. al. 1999

Similarity based on Lessical Matching: **Limits**

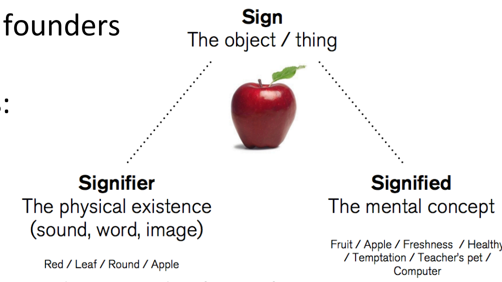
- Up to now the similarity among texts has been based on lessical matching of their words
- For example these two senteces

  1) *powerful car engine* 2) *potent auto motor*
  - they have equivalent meaning, but no lessical match
  - thus the cosine similarity of their two vectors is ZERO

- For example
  a) *powerful math model* b) *powerful car model*
  - different meaning, but higher lessical match than previous sentence
  - thus the cosine similarity of their two vector is 0.67

What is the Meaning of Words ?

- Definition of meaning from the Webster dictionary
  - the idea that is represented by a word, phrase, etc.
  - the idea that a person wants to express by using words, signs, etc.
  - the idea expressed in a work of writing, art
- Commonest linguistic way of thinking of meaning
  - Ferdinand de Saussure was one of the founders of semiotics and sign theory
  - He divided the sign into 2 components: the **signifier** or sound-image and the **signified** or concept
  - signifier = *material form*
    signified = *mental concept*



  - He argued that a sign's meaning can be understood when the relationship between its signifier and signified are agreed
  - moreover the meaning of a word depends on its relations to other words
    - e.g. understanding "tree" requires understanding "bush" and their relation

What about **computational** word meaning ?

An usual answer: a taxonomy like **WordNet** that has 29 relationships: *hypernyms (is-a), synonyms, merenomy ...*

```
from nltk.corpus import wordnet as wn
panda = wn.synset('panda.n.01'); bear = wn.synset('bear.n.01')
hyper = lambda s: s.hypernyms()
list(panda.closure(hyper)); list(bear.closure(hyper));
```

[Synset('procyonid.n.01'),
Synset('carnivore.n.01'),
Synset('placental.n.01'),
Synset('mammal.n.01'),
Synset('vertebrate.n.01'),
Synset('chordate.n.01'),
Synset('animal.n.01'),
Synset('organism.n.01'),
Synset('living_thing.n.01'),
Synset('whole.n.02'),
Synset('object.n.01'),
Synset('physical_entity.n.01'),
Synset('entity.n.01')]

[Synset('carnivore.n.01'),
Synset('placental.n.01'),
Synset('mammal.n.01'),
Synset('vertebrate.n.01'),
Synset('chordate.n.01'),
Synset('animal.n.01'),
Synset('organism.n.01'),
Synset('living_thing.n.01'),
Synset('whole.n.02'),
Synset('object.n.01'),
Synset('physical_entity.n.01'),
Synset('entity.n.01')]

wn.synset('good')
S: (adj) full, good
S: (adj) estimable, good, honorable, respectable S: (adj) beneficial, good
S: (adj) good, just, upright
S: (adj) adept, expert, good, practiced, proficient, skillful
S: (adj) dear, good, near S: (adj) good, right, ripe
...
S: (adv) well, good
S: (adv) thoroughly, soundly, good S: (n) good, goodness
S: (n) commodity, trade good, good

wn.synset('bear.n.01').**lowest_common_hypernyms**(wn.synset('panda.n.01')) *[Synset('carnivore.n.01')]*
panda.wup_similarity(bear) *0.89*

Text Similarity Using Wordnet

- **Wordnet:** a directed graph where each node is an english word and each edge an oriented relation between 2 words
  - e.g. of an **hypernym** relation: *painting* IS-A *graphic_art*
  - **synonym** relation: *car -> auto, automobile, machine, motorcar*
  - e.g. sculpture and painting are independent, i.e. orthogonal
  - instead in wordnet they share a common ancestor: ***art***
- What's the **semantic similarity** between the 2 sentences ?
  - 1) *powerful car engine* *2) potent auto motor*
  - using wordnet and a bit of coding their similarity **is almost 1**
- How to get the right semantic of the other 2 sentences ?
  - a) *powerful math model* b) *powerful car mode*
  - in this case we should preliminarly perform a part of speech tagging (POS)
- Drawbacks: computationally expensive, limited dictionary
- Recent methods: **Word Embeddings** are overcoming such issues

EVALUATION OF RESULTS

INTRODUCTION

User Evaluations

- Each user translate an **information need** in a search **query ….**
- …. and evaluate the relevance of results according to its **information need** *and not on the basis of the used search terms*
- E.g., information need: *I'm interested in knowing if red wine is more effective against the heart failure than white wine.*
- Query: ***wine red white effective heart failure***
- i.e. the user evaluates if the answer satisfies the information need and not if contains these terms

Test Bed for the Relevance Evaluation

- Approach commonly used to measure the docs relevance wrt a query:
  1. one or more reference sets of documents
  2. a reference set of queries
  3. known binary evaluations for each query and the set of documents in order to determine relevant/irrelevant doc
- Human experts evaluate for each query which docs are relevant/irrelevant
  - Text Retrieval Conference (TREC), and National Institute of Standards and Technology (NIST), performed a large number of such tests since 1992

Some Reference Corpora

Collection	NDocs	NQrys	Size (MB)	Term/Doc
ADI	82	35		
AIT	2109	14	2	400
CACM	3204	64	2	24.5
CISI	1460	112	2	46.5
Cranfield	1400	225	2	53.1
LISA	5872	35	3	
Medline	1033	30	1	
NPL	11,429	93	3	
OSHMED	34,8566	106	400	250
Reuters	21,578	672	28	131
TREC	740,000	200	2000	89-3543

Accuracy Evaluation

- **Accuracy (for a given query)**:
  - (num. of relevant retrieved docs + irrelevant not retrieved) / all docs of the corpus

Confusion Matrix	Retrieved	Not Retrieved	doc in the Corpus
Relevant	true positives = tp	false negatives = fn	Relev = tp+fn
Irrelevant	false positives =fp	true negatives = tn	Irrelev = fp+tn

- Accuracy = (tp+tn)/(tp+tn+fn+fp)
- but this measure is insufficient with data sets that have classes with a num. of unbalanced instances

Evaluation: Accuracy

- **Accuracy**: inappropriate measure with unbalanced classes
- Example:
  - corpus with 10200 documents, of which 100 relevant for a given query, but the query achieves 100 docs irrelevant

Confusion Matrix	Retrieved	Not Retrieved	doc in the corpus
Relevant	tp = 0	fn = 100	Relev =100
Irrelevant	fp = 100	tn = 10000	Irrelev=10100

- Acc = (tp+tn)/(tp+tn+fn+fp) = (0+10000)/10200= 0.98

Evaluation with Precision and Recall

- **Precision**: Fraction of docs retrieved that are relevant = *P(relevant|retrieved)* = tp/(tp + fp)
- **Recall**: Fraction of docs relevant that have been retrieved = *P(retrieved |relevant)* = tp/(tp + fn)
- Previous example with accuracy 0.98:

Confusion Matrix	Retrieved	Not Retrieved	doc in the corpus
Relevant	tp = 0	fn = 100	Relev =100
Irrelevant	fp = 100	tn = 10000	Irrelev=10100

- ***Precision = 0*** ***Recall = 0***

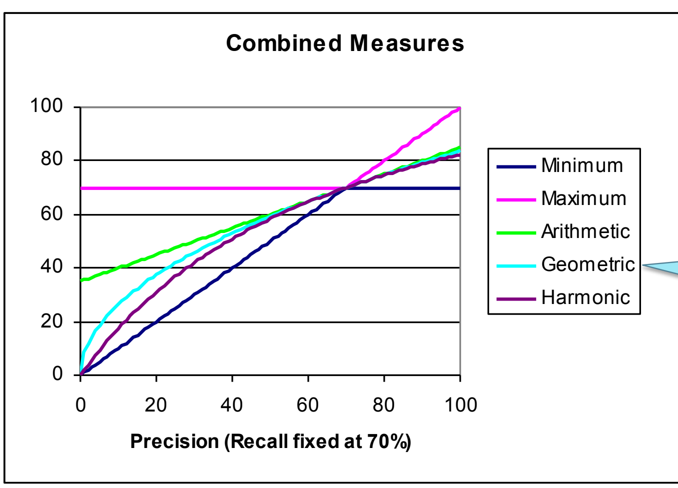
Precision and Recall: F-Measure

- Retrieving for each query all corpus docs, the recall would be 1, but the precision would be low
  - up to the limit given by the number of relevant docs for the query over the total num. of docs
- Generally in a good system, the recall tends to decrease as the precision increases and vice versa
- A measure that combines them is the F-Measure:

$$Fmeasure = \frac{1}{\alpha\frac{1}{P} + (1-\alpha)\frac{1}{R}} = \frac{(\beta^2+1)PR}{\beta^2 P + R}$$

- si utilizza spesso con β=1 ➔ $F_1 measure = 2 \cdot \frac{PR}{P+R}$

F_1 measure and other combined measures

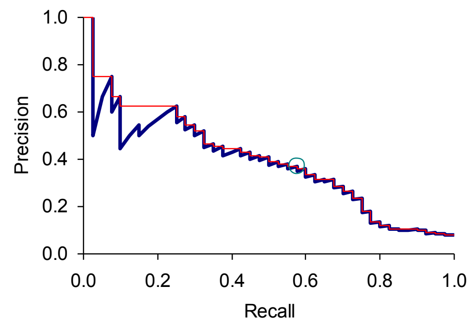


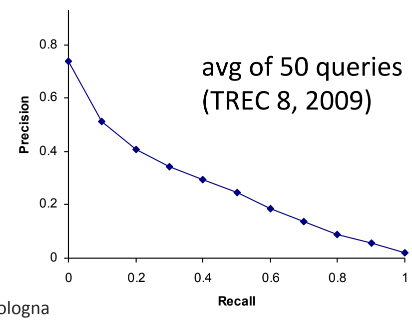
$$\sqrt[n]{\prod_{i=1}^{n} x_i}$$

- F1 measure is also known as the armonic measure
- the result is closer to the smallest between Precision & Recall

Evaluation of Ranked Results

- Given a system and a set of queries with their results
- we plot for each query a graph *precision-recall* with the first k answers
- in general it is plotted the average graph from the graphs of queries
- How many points (R,P) to plot ?
- Standard TREC: 11 Recall levels with 0.1 interpolated increments





Precision-Recall: Example

n	docID	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	

Let 6 be the total number of relevant docs {1,2,4,6,13,20}

Evaluation of each recall point

R=1/6=0.167; P=1/1=1

R=2/6=0.333; P=2/2=1

R=3/6=0.5; P=3/4=0.75

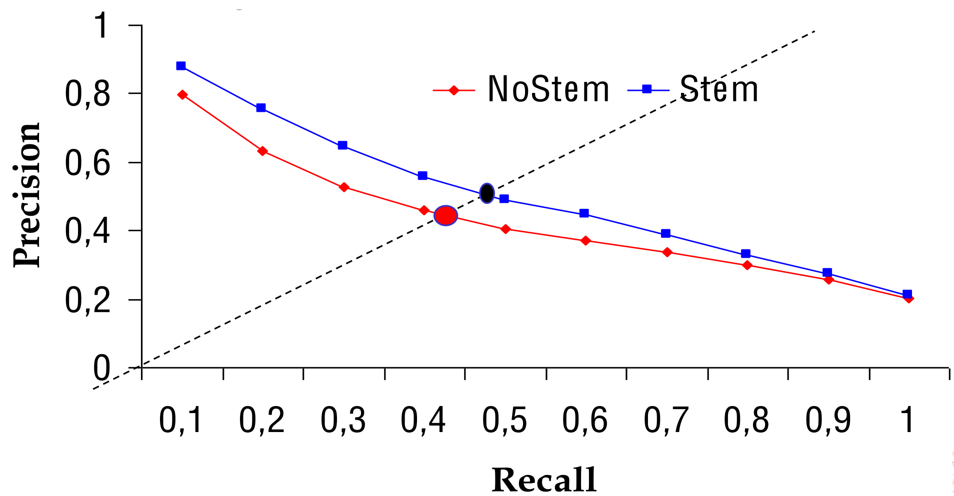
R=4/6=0.667; P=4/6=0.667

R=5/6=0.833; p=5/13=0.38

Missing the relevant doc 20 therefore the recall is not 100%

Breakeven Point of Precision-Recall

It is the interpolated value such that precision and recall are identical



Precision-Recall: Example

n	docID	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	

Let 6 be the total number of relevant docs {1,2,4,6,13,20}

Evaluation of each recall point

R=1/6=0.167; P=1/1=1

R=2/6=0.333; P=2/2=1

R=3/6=0.5; P=3/4=0.75

R=4/6=0.667; P=4/6=0.667

Breakeven point Prec. = Recall

R=5/6=0.833; p=5/13=0.38

Missing the relevant doc 20 therefore the recall is not 100%

Further evaluation measures

- R-precision
  - let R be the num. of docs relevant for a given query, R-precision is the num. of docs relevant in the first R doc retrieved divided by R
  - e.g.: the query in fig. has R=6 docs relevant (the last not retrieved), the 6-precision is 4/6
- Mean Average Precision (MAP)
  - MAP of a query q_j is the average of **R-precision** from 1 to m_j relevant doc of q_j ; the MAP of a set Q of queries is the following

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	

$$MAP(Q) = \frac{1}{|Q|} \sum_{j=1}^{|Q|} \frac{1}{m_j} \sum_{k=1}^{m_j} Precision(R_{jk})$$

Mean Average Precision: Example

n	doc #	relevant
1	320	x
2	401	
3	756	x
4	891	x
5	467	x
6	301	
7	222	x
8	591	x
9	191	
10	668	

n	doc #	relevant
1	420	
2	411	x
3	956	
4	821	x
5	467	
6	321	x
7	223	x
8	551	
9	971	
10	268	

- Let q_1 , q_2 two queries to which corresponds 10 and 8 relevant docs respectively, in the corpus *D*

$$MAP(Q) = \frac{1}{|Q|} \sum_{j=1}^{|Q|} \frac{1}{m_j} \sum_{k=1}^{m_j} Precision(R_{jk})$$

  - MAP($\{q_1, q_2\}$): $\frac{1}{2}\left(\frac{1}{10}\cdot\left(\frac{1}{1}+\frac{1}{2}+\frac{2}{3}+\frac{3}{4}+\frac{4}{5}+\frac{4}{6}+\frac{5}{7}+\frac{6}{8}+\frac{6}{9}+\frac{6}{10}\right)+ + \frac{1}{8}\cdot\left(\frac{0}{1}+\frac{1}{2}+\frac{1}{3}+\frac{2}{4}+\frac{2}{5}+\frac{3}{6}+\frac{4}{7}+\frac{4}{8}\right) \right)$

$$= \frac{0.711+0.413}{2} = 0.562$$

  - MAP approximates the area average of the precision-recall graph of a set of queries
  - each query has the same weight in the MAP, also with large difference in the num. of relevant docs among queries