

# Social MINING

IIº SEM

2021



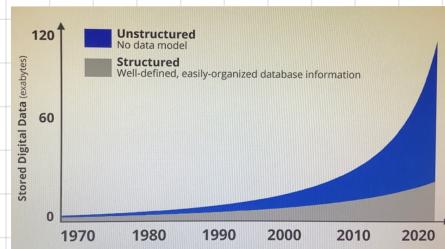
11/03

# INFORMATION RETRIEVAL (IR)

## IR - TASK

STRUCTURED vs  
UNSTRUCTURED DATA

- Trovarci materiali (docs, video, images...)
- Spesso i dati sono **UNSTRUCTURED**, addirittura nelle grandi compagnie
  - ↳ **SEMI-STRUCTURED** sono ad esempio le slides con **FIELDS** come **TITLO** e **CONTENUTO**.
- **STRUCTURED**: Relational DBs, XML, ...



IR riguarda solo i 9  
RETRIEVAL? → NO

- **CLOUDING**: cluster di documenti
- **CATEGORIZATION**: dato un insieme di topic assegnargli un documento
- **INFORMATION EXTRACTION**
- **QUESTION ANSWERING**: Rispondere a domande come: facts, How, Why
- **OPINION MINING**: analizzare il sentimento dietro i testi

## Terminology

**SEARCHING**: ricerca di una **specifico** informazione

**BROWSING**: **UNSTRUCTURED EXPLORATION**

**CRAWLING**: muoversi tra **HYPYERLINKS**

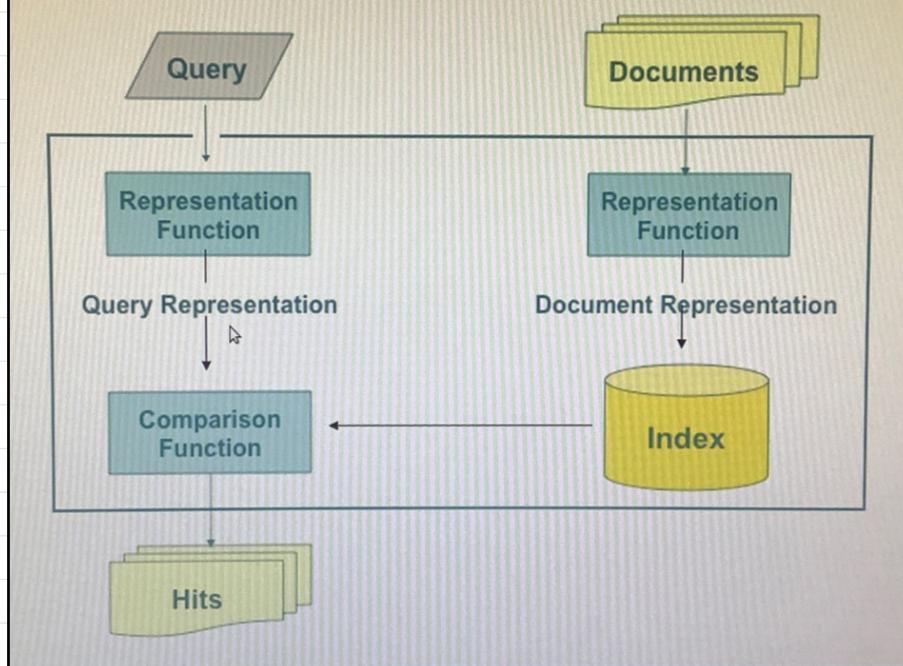
**SCRAPING**: estrarre contenuto da pagine

**QUERY**: **Stringa**

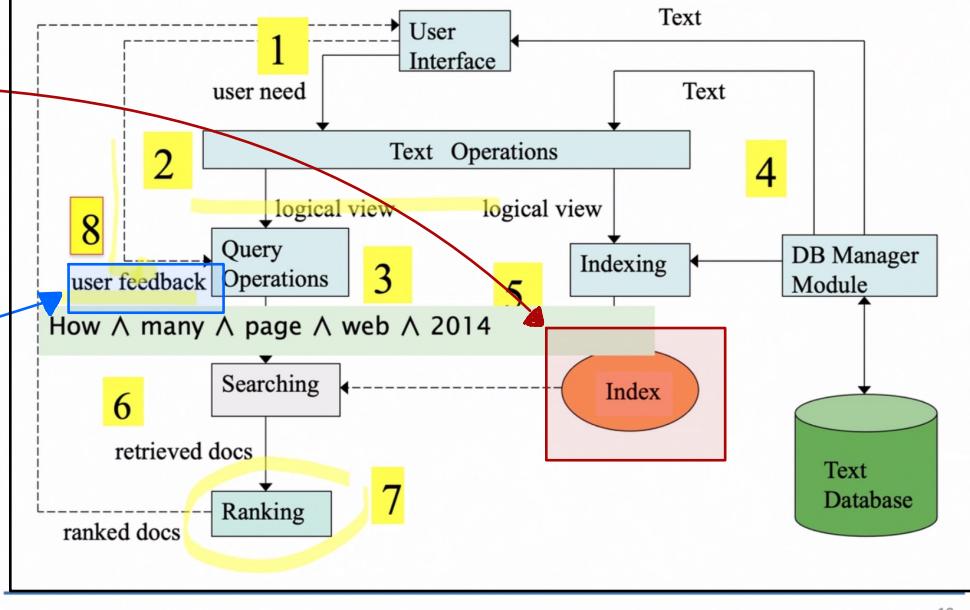
**FULL-TEXT SEARCHING**: comporre la query con ogni parola del testo

**FIELDED SEARCH**

## Inside The IR Black Box



# Workflow



L'INDEX è ciò che vogliamo creare al fine di indirizzare la query

Se l'utente dica un risultato, gli abbiamo dato ciò che cercava.

es. Può dipendere da FEEDBACK (oppure da PARTE RANK)

• Ottenuti i risultati come li presentiamo all'utente?

SORTING

RANKING by similarity tra query e documenti

RANKING by importance

## DOCUMENT REPRESENTATION

. Dato un documento non strutturato  $\Rightarrow$  strutturato

(1) BAG OF WORDS MODEL : Sia un array dei Token

LDVARIANTI

HOW TO WEIGHT A WORD

$\hookrightarrow$  es. BOOLEAN : Data una lista, assegna 1 se la Parola è nel documento

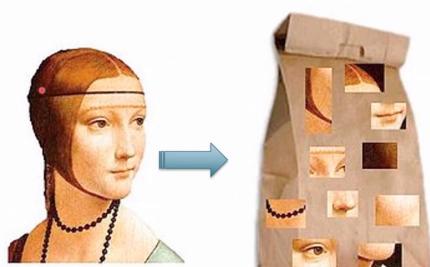
WHAT IS A WORD

$\hookrightarrow$  SINGLE WORD

DOUBLE " : Nome e cognome

INCONDIZIONE : {go, gone, going...}, o  
{polite, impolite, dormire, casa}

$\hookrightarrow$  usato anche con immagini



# DOCUMENT PARSENG

- composta scanning e trasformazione in bag of words
- ci servono 3 cose:
  - lingua (esiste MULTILINGUALITY)
  - tipo di file
  - character set
- {
  - è un singolo file?
  - è una singola pagina web o un sito?

## UNIT DOCUMENT

Cos'è un  
unità di  
documento?

## TOKENIZATION

### ISSUES

- Un TOKEN è una sequenza di caratteri

- Come scegli i token?

- ↳ Prendo <Nome> - <sogno> insieme?
- ↳ Prendo punteggiatura?
- ↳ Prendo STOP WORDS?
- ↳ i numeri (DATE, NUM. DI TELEFONO, ...)
- ↳ Lingue (arabo, cinese, ...)

## APPROCCI

**STOP WORDS**  
• AND, A, TO,  
↳ tempo stopgo?

## NORMALIZATION

### ACRONIMO

- U.S.A. => USA

### SINONIMO

- CAR => AUTOMOBILE

### TYPO

- GOOLGE => GOOGLE

## LEMMATIZATION

- Riduzione a un LEMMA

es. {are, is, am} → be

{car, cars, car's} → car

## STEMMING

- Riduzione alla RADICE comune

for example compressed  
and compression are both  
accepted as equivalent to  
compress.

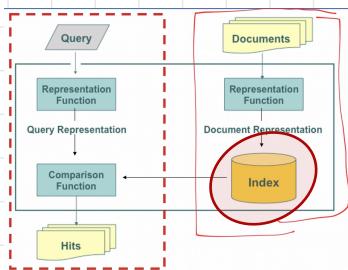
for example compress and  
compress are both accepted  
as equivalent to compress

## CASE-FOLDING

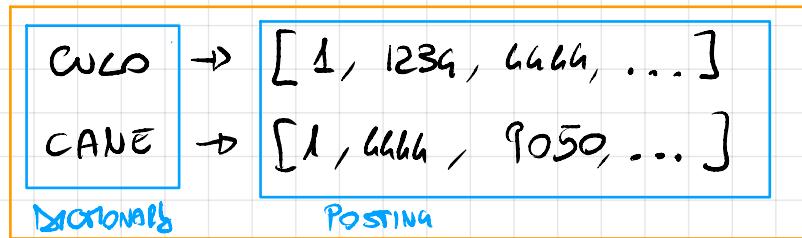
- tutto in lowercase

17/03

## INVERTED INDEX



- L'index è una lista che mappa ogni termine agli IDs dei documenti che lo contengono (POSTING LIST). È detto INVERTED perché non mappa DOC → TERMS MA TERM. → Docs.

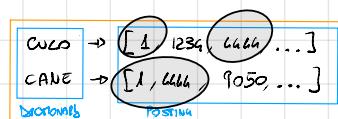


- Inoltre nel DICTIONARY non teniamo le occorrenze del termine totale, ma solo il numero di documenti in cui appare

## INDEX SEARCH

### Query

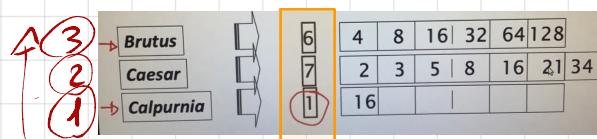
- query: "CUCO AND CANE"  
→ Merghiamo i Postings di entrambi e riportiamo quindi i Docs {1, 6666}



### OPTIMIZZAZIONE

- Merghiamo a partire da quello con POSTING più corto.

Lo ecco perché abbiamo tenuto i **num. di docs** in cui appare, così possiamo sapere a priori l'ordine di MERGING



## PHRASE query

- query: "RED BRICK HOUSE" → non va bene, il modo di prima, perché le parole sono legate

## BI-WORD INDEXIS

- Rappresentiamo i due dizionari con dopp. termini
- Ad esempio dalla query: "STANFORD UNIVERSITY PAO ALTO" esco:

"STANFORD UNIVERSITY" AND "UNIVERSITY PAO" AND "PAO ALTO"

- Ovviamente la grandezza dell'INDEX cresce di bretto

- Per ogni termine stiamo sia i doc sia la posizione del termine nel documento, c'è un termine nella doc.freq.

```
<term, number of docs containing term;
doc1: position1, position2 ... ;
doc2: position1, position2 ... ;
etc.>
```

to, 993427: \*

1, 6: (7, 18, 33, 72, 86, 231);  
2, 5: (1, 17, 74, 222, 255);  
4, 5: (8, 16, 190, 429, 433);  
5, 2: (363, 367);  
7, 3: (13, 23, 191); ...)

be, 178239:

1, 2: (17, 25);  
4, 5: (17, 191, 291, 430, 434);  
5, 3: (14, 19, 101); ...)

Ad esempio per la query: "To BE or NOT To BE"

- Extract inverted index entries for each distinct term: **to, be, or, not.**
- Merge their *doc:position* lists to enumerate all positions with "to be or not to be".

• **to:**

- 2:1,17,74,222,551;
- 4:8,16,190,429,433;
- 7:13,23,191; ...

• **be:**

- 1:17,19;
- 4:17,191,291,430,434; 5:14,19,101; ...

① MELCO I DOCUMENTI  
IN CW APPARE

② CONTROLLA SE LE  
POSIZIONI IN UNO DOC.  
SONO CONSECUTIVE

## RELAXATION

- Possiamo rilassare la tecnica riferendoci a una certa distanza nei documenti

- For example: *employment /4 place*  
**Find all documents that contain EMPLOYMENT and PLACE within 4 words of each other.**
- "*Employment agencies that place healthcare workers are seeing growth*" is a hit.
- "*Employment agencies that have learned to adapt now place healthcare workers*" is **not a hit.**

## RANKING

## BOOLEAN MODEL

- Query e Docs rappresentati come espressioni booleane



$$\begin{aligned} q = a \wedge (b \vee (\neg c)) &= \\ (a \wedge b \wedge c) \vee (a \wedge b \wedge (\neg c)) \vee (a \wedge (\neg b) \wedge (\neg c)) &\quad (\text{DNF form}) \\ \square \vee (q_{\text{dnf}}) &= (1,1,1) \quad (1,1,0) \quad (1,0,0) \end{aligned}$$

» Disjunctive Normal Form

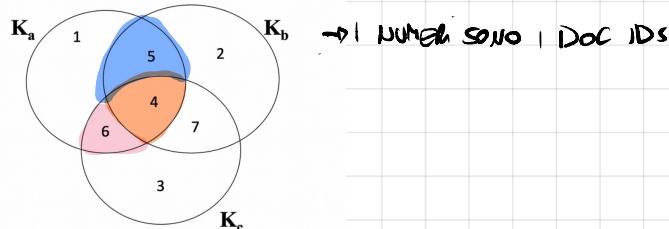
» Ex:  $(apple, computer, red) \vee (apple, computer) \vee (apple)$

»  $\square \vee (q_{\text{cc}}) = (1,1,0)$

» Conjunctive Component

c'è un MATH perché i documenti ci interessano

- Similar/Matching documents
- $md_1 = [apple apple blue day] \Rightarrow (1,0,0)$
- $md_2 = [apple computer red] \Rightarrow (1,1,1)$



$$\begin{aligned} q = k_a \wedge (k_b \vee k_c) \\ (1 \wedge 1 \wedge 1) \vee (1 \wedge 1 \wedge 0) \vee (1 \wedge 0 \wedge 1) \end{aligned}$$

Which one?

- Il problema è che tutti i documenti sono allo stesso livello, e magari a una query possono essere radati TROPPI / POCO risultati. Perché un documento o match o non lo fa quindi questi query generano risultati.
- Però se cercassimo un metodo di scoring delle query, TROPPI risultati non sarebbero un problema, perché l'utente guarda i primi risultati.

OBS

## VECTOR WEIGHTED MODEL

- Abbiamo un modello **BAG OF WORDS VECTOR N-DIMENSIONALE** dove  $|V|$  è la dimensione del vocabolario, abbiamo inoltre un **WEIGHTING SCHEME**  $w_{t,d}$  che esprime la frequenza della TERM  $t$  nel DOCUMENTO  $d$ .

Come siamo un valore  $w$ ?

TERM FREQ.

CUMULATIVE  
MODEL

- Frequenza  $tf_{t,d}$  del termine  $t$  nel documento  $d$

- Ci serve una sommatoria tra query e documento, sommiamo l'occorrenza di  $t$  tra i termini che  $q$  e  $d$  hanno in comune

$$\text{sim}(q,d) = \sum_{t \in q \cap d} tf_{t,d}$$

- Dobbiamo normalizzare i TERM FREQ.

$$tf'_{t,d} = tf_{t,d} / \sum_{i \in d} (tf_i)$$

$$\text{sim}(q,d) = \sum_{t \in q \cap d} (tf'_{t,d} / \sum_{i \in d} (tf_i))$$

- Alternativa alla normalizzazione

$$w_{t,d} = \sum_{i=0}^{\infty} (1 + \log tf_{t,d}) \quad \text{if } tf_{t,d} > 0, \text{ else } 0$$

$$\text{sim}(q,d) = \sum_{t \in q \cap d} (1 + \log tf_{t,d})$$

es  
 $0 \rightarrow 0$   
 $1 \rightarrow 1$   
 $2 \rightarrow 1.3$   
 $10 \rightarrow 2$   
 $1000 \rightarrow 4$

È abbastanza?

- Non ci stiamo  $\rightarrow$ , termini rari daranno più informazioni di quei frequenti

## INVERSE DOCUMENT FREQUENCY

NOTA: c'è anche la colection  
freq., cioè il numero di  
occorrenze in tutta la  
colectione (non è il numero  
dei documenti in cui appare)

- $\text{df}$  è il numero di documenti che contiene  $t$  in una collezione di  $N$  documenti (DOCUMENT FREQUENCY)

- L' INVERSE DOC. FREQUENCY e:

$$\text{idg}_t = \log_{10} N/dg_t$$

**Quindi si posso come li calcolano alla fine?**

- $$w_{t,d} = (1 + \log t f_{t,d}) \cdot \log N/df$$

↓                              ↓

CUMULATIVE APPROXIMATION CON  
LOG FREQ.

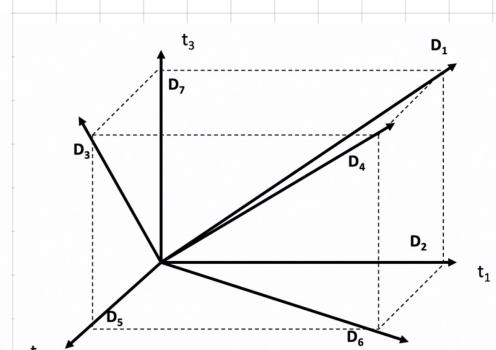
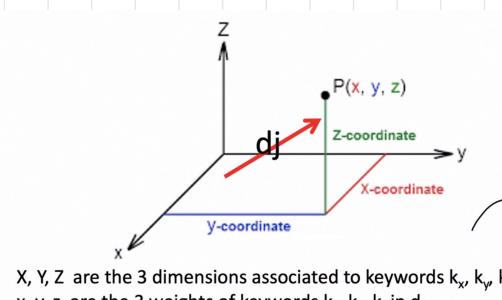
INVERSE TERM FREQUENCY

\* Più alta è da più tempi è comune

\* P15 altre i N/dl P16,9 tecniche e Rete

\* Più alto è  $w_{td}$  più i termini di ER sono

- So we have a  $|V|$ -dimensional vector space, one dimension for each term.
  - Terms are axes of the space
  - Documents are points or vectors in this space.
  - The coordinate of a vector  $d_j$  on dimension  $i$  is the tf-idf weight of word  $i$  in document  $j$ .
  - **Very high-dimensional:** hundreds of thousand of dimensions when you apply this to a web search engine
  - It is a very sparse vector - most entries are zero (will see later in this course how to reduce dimensionality).

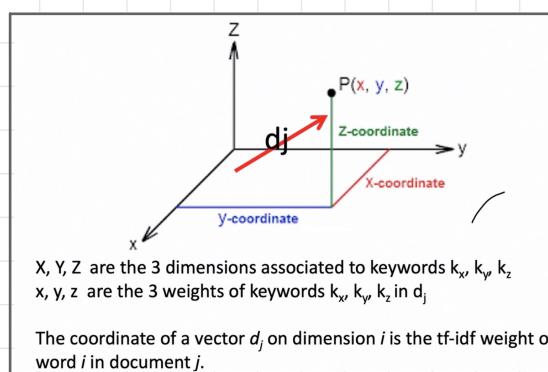


# DOCUMENT AS VECTORS

- So we have a  $|V|$ -dimensional vector space, one dimension for each term.
- Terms are axes of the space
- Documents are points or vectors in this space.
- The coordinate of a vector  $d_j$  on dimension  $i$  is the tf-idf weight of word  $i$  in document  $j$ .
- **Very high-dimensional:** hundreds of thousand of dimensions when you apply this to a web search engine
- It is a very sparse vector - most entries are zero (will see later in this course how to reduce dimensionality).



- A 3-dimensional



## VECTOR SPACE SCORING MODEL

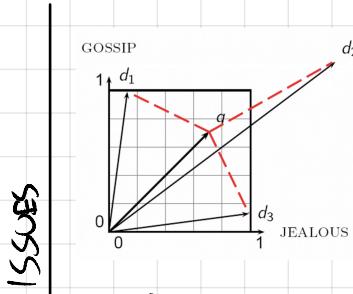
- IDEA 1: facciamo lo stesso per le query
- IDEA 2: RANK dei documenti in base alla proximità con la query

## VECTOR SPACE PROXIMITY

- Proximità di due punti, query e doc

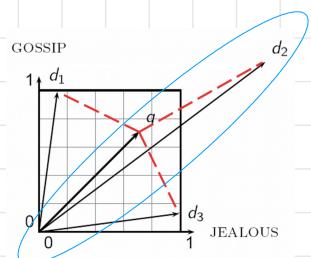
**• CHE DISTANZA?**

$$\text{• EUCLIDEA? } d(d_3, q) = \sqrt{\sum_i (w_{i3} - w_{iq})^2}$$



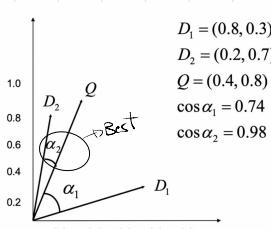
• 19 documenti da parla sia di GOSSIP, ma la query è più vicina ai documenti che parlano SOLO DI GESSA (d3) o SOLO DI GOSSIP (d1)

- Inoltre due vettori sullo stesso argomento possono avere lunghezze diverse, anche nonché angoli con la stessa lunghezza non funzionano.



- ANGOLI tra documenti, se l'angolo tra q e d1 è < o > allora la proximità tra i due è massima

↳ RANKING dei docs sulla query  
ORDINE DECREScente dell'angolo  
ORDINE CREScente del COSENO



## ANGLE RANKING

$$\cos(\vec{q}, \vec{d}) = \frac{\vec{q} \bullet \vec{d}}{\|\vec{q}\| \|\vec{d}\|} = \frac{\vec{q}}{\|\vec{q}\|} \bullet \frac{\vec{d}}{\|\vec{d}\|} = \frac{\sum_{i=1}^{|V|} q_i d_i}{\sqrt{\sum_{i=1}^{|V|} q_i^2} \sqrt{\sum_{i=1}^{|V|} d_i^2}}$$

$q_i$  is the tf-idf weight of term  $i$  in the query  
 $d_i$  is the tf-idf weight of term  $i$  in the document  
 $\cos(\vec{q}, \vec{d})$  is the cosine similarity of  $\vec{q}$  and  $\vec{d}$  ... or,  
equivalently, the cosine of the angle between  $\vec{q}$  and  $\vec{d}$ .



q1 = t1 t2 t4  
q2 = t2 t3  
q3 = t2 t4

$$\cos(\vec{q}, \vec{d}) = \frac{\vec{q} \bullet \vec{d}}{\|\vec{q}\| \|\vec{d}\|} = \frac{\vec{q}}{\|\vec{q}\|} \bullet \frac{\vec{d}}{\|\vec{d}\|} = \frac{\sum_{i=1}^{|V|} q_i d_i}{\sqrt{\sum_{i=1}^{|V|} q_i^2} \sqrt{\sum_{i=1}^{|V|} d_i^2}}$$

LENTH OF VECs

N=970

DocID	Term Frequency (TF)			Log-TF			DocFreq (DF)	N/DF	LOG(N/DF) : IDF	Informativeness	L2		
	d1	d2	d3	d1	d2	d3					d1	d2	d3
t1	1000	100	100	4	3	3	652	1,487730061	0,172524139	Common	0,4762332105	0,26788121	
t2	0	30	2	0	2,47712125	1,30103	59	16,44067797	1,215919723	Medium rare	0	9,072027068	2,50255957
t3	0	0	2	0	0	0	1,30103	5	194	Rare	0	0	8,85954436
t4	0	10	0	0	2	0	254	3,818897638	0,581938018	Common	0	1,354607426	0
							970				0,690096554	3,270247039	3,4102764

t1 t2 t4	q as Binary Vector	Term Frequency (TF)			Log-TF			TF-IDF			COSINE			
		sim(q1,d1)	sim(q1,d2)	sim(q1,d3)	sim(q1,d1)	sim(q1,d2)	sim(q1,d3)	sim(q1,d1)	sim(q1,d2)	sim(q1,d3)	L2(Q1)	(q1,d1)	(q1,d2)	(q1,d3)
t1	1	1000	100	100	4	3	3	0,690096554	0,517572416	0,517572416	0,029764579	0,119058314	0,089293735	0,08929374
t2	1	0	30	2	0	2,47712125	1,30103	0	3,011980589	1,581948031	1,478460772	0	3,662326602	1,92352181
t3	0	0	0	0	0	0	0	0	0	0	0	0	0	0
t4	1	0	10	0	0	2	0	0	1,163876035	0	0,338651856	0	0,677303713	0
Comulative		1000	140	102	4	7,47712125	4,30103	0,690096554	4,69342904	2,099520447	1,358998604	0,12694946	0,996549007	0,43430556
Rank		1	2	3	3	1	2	3	1	2	3	1	2	2

## SIMILARITY VS DISTANCES

All data mining problems, such as clustering, outlier detection, and classification, require the computation of similarity.

A methodical way of quantifying similarity between data objects is required.

A formal statement is:

Given two objects  $O_1$  and  $O_2$ , determine a value of the similarity  $\text{Sim}(O_1, O_2)$  (or distance  $\text{Dist}(O_1, O_2)$ ) between the two objects.

In similarity functions, larger values imply greater similarity, whereas in distance functions, smaller values imply greater similarity.

In some domains, such as spatial data, it is more natural to talk about distance functions, whereas in other domains, such as text, it is more natural to talk about similarity functions.

- Le distance functions sono necessarie per algoritmi di DATA MINING
- Di solito hanno una forma chiusa, ma spesso sono Algoritmi
  - come  $\cos(q, d_i)$

- 
- Esprimere la distanza tra due punti,  $X = (x_1, \dots, x_d)$ ,  $Y = (y_1, \dots, y_d)$

$$\text{Dist}(\bar{X}, \bar{Y}) = \left( \sum_{i=1}^d |x_i - y_i|^p \right)^{1/p}$$

- $L^1$  EUCLIDEAN NORM è una specializzazione con  $P=2$
- Oppure un'altra è la MANHATTAN DISTANCE con  $P=1$



- Il problema di queste distanze è che non vanno bene con troppe dimensioni per via di caratteristiche dei dati come NOISE, SPARSE, ...

## MINKOWSKI DIST

- Introduce un coefficiente per ogni dimensione che pesa specifiche feature dei dati.

$$Dist(\bar{X}, \bar{Y}) = \left( \sum_{i=1}^d a_i \cdot |x_i - y_i|^p \right)^{1/p}$$

• detta anche  $L_p$ -GENERALIZZATA

CHE IMPATTO HA CON HIGH DIMENSIONALITY? → SLIDES!

## OBJECT ORIENTED PROGRAMMING

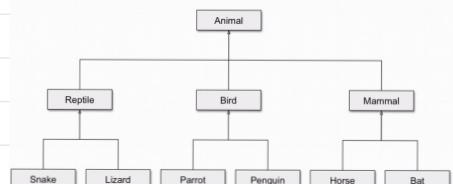
### OBJECT CLASS INHERITANCE

- Composed by ID, ATTRIBUTES, SERVICES
- È un TIPO che definisce caratteristiche (ATTR) e comportamenti di oggetti simili

#### SPECIALIZATION

- sottoclassi ereditano ATTR e SERV
- " possono aggiungere nuovi ATTR e SERV
- " riuscire il codice delle sottoclassi
- " forniscono comportamenti specializzati (OVERLOAD e DYN. BINDING)

#### Class hierarchy



#### GENERALIZATION

- superclassi generalizzano ATTR e comportamenti comuni alle sottoclassi

## ENCAPSULATION

- Separa lo stato interno e esterno degli oggetti (IS PRIVATE <vs> IS PUBLIC)

## INFORM. HIDDEN

- Possibile rigenerare l'implementazione dell'oggetto

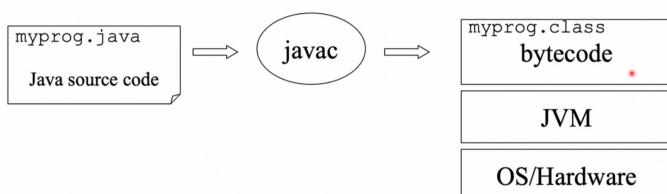
## MODULARITY

- Un oggetto è indipendente dagli altri → facile codificare e riuscirlo

## DRAWBACKS

- I dati condivisi devono essere gestiti con DESIGN PATTERNS

- JAVA VIRTUAL MACHINE → Java È eseguito su un CPU VIRTUALE
  - ↳ PLATFORM INDEPENDENCY



# WRAPPERS vs PRIMITIVE TYPES

## Wrapper

```
Integer n = new Integer("4");  
int m = n.intValue();
```

## Primitive

```
int p = 4;
```

---

### Hello.java

```
class Hello {  
    public static void main(String[] args) {  
        System.out.println("Hello World !!!");  
    }  
}
```

C:\javac Hello.java

(compilation creates Hello.class)

C:\java Hello

(Execution on the local JVM)

oh no! hammo  
ammazzato kenny!

---

```
class Kyle {  
    private Boolean kennyIsAlive_;  
  
    Default C'tor  
    public Kyle() { kennyIsAlive_ = true; }  
    public Kyle(Kyle aKyle) {  
        kennyIsAlive_ = aKyle.kennyIsAlive_;  
    }  
  
    Copy C'tor  
    public String theyKilledKenny() {  
        if (kennyIsAlive_) {  
            kennyIsAlive_ = false;  
            return "Oh Nooooo!!!";  
        } else {  
            return "?";  
        }  
    }  
    public static void main(String[] args) {  
        Kyle k = new Kyle();  
        String s = k.theyKilledKenny();  
        System.out.println("Kyle: " + s);  
    }  
}  
  
import java.util.Scanner;  
  
public class God {  
    public static void main(String args[]) {  
        Scanner scan = new Scanner(System.in); → Se scrivo da keyboard "resurrection", Dio  
        Kyle k = new Kyle();  
        String text;  
  
        while(true) {  
            text = scan.nextLine();  
  
            if (text.equals("resurrection")) {  
                k = new Kyle();  
            }  
  
            String s = k.theyKilledKenny();  
            System.out.println("Kyle: " + s);  
        }  
    }  
}
```

