Data Semantics

Sommario

Data Semantcs si occupa di comprendere il significato dei dati, nella pratica della scrittura del programma. È necessario prestare attenzione al significato dei dati nell'integrazione di più dataset; inoltre la semantica del dato è necessaria per la condivisione di dataset (cioè renderli fruibili da chi non ha prodotto il dataset). Altro problema centrale è la capacità di usare dati non strutturati, in modo tale da facilitare query.

Scopo del corso è strutturare dei modelli per la semantica dei dati in modo tale da facilitarne l'uso; inoltre si stabiliscono strategie per attribuire semantica ai dati. Bisogna inoltre capire il ruolo della semantica nell'integrazione dei dati. Il corso tratta della semantica dei dati nei *big data*, dell'estrazione dello *knowledge graphs* (ovvero le relazioni tra gli elementi di un database, *data linkage*) o la costruzione di sistemi di raccomandazione. Inoltre saranno analizzate alcune tecniche di *natural language processing* e la costruzione di rappresentazioni a partire dai dati.

Le esercitazioni si occuperanno di interrogare *knowledge graphs*, modellare e costruire grafi di conoscenza e integrare fonti di dati.

L'esame orale sarà accompagnato da un progetto software (effettuato in gruppo di, circa, 3 persone), di cui sarà fatta una presentazione orale; in alternativa al progetto è possibile scrivere un articolo di approfondimento su una tematica. La preparazione sarà "ragionevolmente" dettagliata su tutti gli argomenti, e sarà approfondito l'argomento del progetto.

Parte I
Grafi di conoscenza

Il termine è coniato da Google per indicare uno strumento usato dal suo motore di ricerca. Il grafo permette di essere processato facilmente da una macchina e allo stesso tempo permette un livello di astrazione soddisfacente; si possono anche effettuare query come in un database a grafo. Il modello a grafo permette inoltre una facile integrazione di sorgenti diverse. La costruzione di grafi di conoscenza è spesso effettuata a mano da una moltitudine di utenti. Il modello *Semantic Web* ha costruito linguaggi e strumenti, approvati dal W3C, per definire, interrogare e fare inferenza su grafi di conoscenza. Nel mondo reale tuttavia non sono usati questi linguaggi.

Internet produce enormi quantità di dati diversi tra di loro, usati spesso per altri fini: la semantica dei dati si occupa di integrare grandi quantità (*data volume*) da diverse fonti di dati (*data variety*). Questo permette la costruzione di intelligenze artificiali, ovvero di programmi che eseguono task tipicamente umani con risultati simili.

Esistono grafi di conoscenza aperti, quali DBpedia, Yago o Wikidata; anche alcune aziende private sviluppano il proprio per facilitare la propria attività imprenditoriale.

I dati possono essere *strutturati* (tabelle ordinate), *semi-strutturati* (tabelle annidate) o *non strutturati* (testi).

È impossibile effettuare a mano certi compiti particolarmente ardui, come l'integrazione di serie temporali con altri documenti riguardanti lo stesso tema, soprattutto con una scarsa conoscenza del dominio.

1 *Linking Data.*

I dati sono spesso collegati in modo automatico grazie a programmi di apprendimento automatico. Una ricerca su internet non è fatta per documenti ma per contenuti: un motore di ricerca in passato offriva una serie di documenti senza offrire informazioni aggiuntive; oggi invece un motore di ricerca tenta direttamente di rispondere con *factual information* rilevanti nella ricerca. Per *fatto* si intende un'informazione interpretabile come vera o falsa (al contrario, alcuni dati quali immagini o suoni non sono interpretabili per veridicità). I fatti costituiscono un elemento centrale per l'analisi.

Le risposte fattuali sono personalizzate in base alla natura della ricerca, secondo criteri *data driven* (ovvero statistico). Informazioni di tipo diverso hanno caratteristiche diverse: si produce un grafo di conoscenza per gestire meglio entità di tipo diverso con caratteristiche peculiari diverse. Le *preview* dei contenuti sono generate chiedendo agli sviluppatori di inserire dei contenuti nel codice HTML che sono poi interpretati dal motore di ricerca.

I chatbot sono costruiti con l'ausilio di reti neurali e rappresentazioni del mondo reale.

Internet può essere interrogato

2 Spazio vettoriale.

Le query e i documenti sono interpretabili come vettori; per calcolare la similarità tra due vettori, si usa la distanza coseno.

$$\begin{aligned} sim(r,u) &= cos(\theta) \\ &= \frac{uq}{|u||q|} \\ &= \end{aligned}$$

La rappresentazione vettoriale è alla base dell'analisi testuale. Si usa un sistema di pesi più sofisticato per valutare diversamente l'importanza di una parola all'interno di un documento. Una parola è tanto più importante nel documento quante più ricorrenze ci sono nel documento

(in percentuale).

$$tf_{i,j} = \frac{n_{i,j}}{|d_j|}$$
$$idf_i = \log \frac{|D|}{|\{d : i \in d\}|}$$
$$tfidf_{i,j} = tf_{i,j} \times idf_i$$

Le entità, le loro proprietà e i collegamenti tra di loro sono iscritti nel grafo di conoscenza. Esistono linguaggi formali, definiti a inizio '900 che permettono di dichiarare relazioni tra entità, ma che non sono leggibili da una macchina. Inoltre si possono usare assiomi logici per definire le ricorrenze nel testo.

3 Standard.

RDF (*Resource Des...*) è lo standard per il web semantico, ovvero per l'internet elaborabile dalle macchine. L'unità base per rappresentare l'informazione è rappresentata da triple (affermazioni), ovvero grafi etichettati, identificati da URI (*unique resource identifier*). Esempi di triple sono:

```
<Electric Piano, label,
                    "Electric Piano"@en>
<Elton John, instrument, Electric Piano>
<Sails, artist, Elton John>
```

Le triple sono rappresentabili come un grafo diretto, aciclico ed etichettato:

```
     Elton John -[artist]- Sails
     /          \
  [instrument]   [artist]- Empty Sky
   /
 Electric Piano
    \
     [label]- "Electric Piano"@en
```

Alle triple si applicano degli identificativi globali (una sorta di chiave primaria SQL): si usano generalmente identificativi web (abbreviati), che specificano come recuperare l'informazione (qualsiasi cosa a cui si può attribuire un valore costante è definibile come URI). Tutti gli URI sono risorse, e rappresentano l'ontologia del dominio.

Le triple sono strutturate con un soggetto, un predicato e un oggetto. Il soggetto è composto da un URI o da un *black node* (ovvero costanti), il predicato da un URI, un *black node* o da un letterale, a cui può essere attribuito un tipo (stringhe, stringhe, date o booleani) per permettere operazioni. I *black node* sono nodi anonimi per consentire una buona costruzione del grafo.

3.1 Pubblicazione.

I *linked data* riassumono delle pratiche per pubblicare e unire dati provenienti dal web:

- usare URI come nomi delle cose;
- usare indirizzi HTTP come URI;
- inserire informazioni utili su un URI;
- includere link ad altri URI.

Sono solamente una serie di principi, non rendono i dati *open* (*Linked Open Data*).

Altro approccio per pubblicare i dati sul web è usare l'*annotazione semantica*: si inseriscono annotazioni di una pagina web che specifichino il significato del contenuto o altri meta-dati. La sintassi è simile a XML: RDF in precedenza usava il formato XML, non Turtle, ma è stato abbandonato per la sua struttura ad albero in favore di una struttura a grafo. Con l'aggiunta di contenuti strutturati si permette l'elaborazione da parte di agenti intelligenti. L'approccio è tornato in auge grazie ai *crawler* dei motori di ricerca, che annotano le pagine in base al loro significato semantico[1].

Non tutti i formati sono compatibili con RDF (Microdata e hCard non lo sono) mentre altri sì (RDFa, JSON-LD); nonostante questo non ci sono differenze significative tra i vari formati (il modello è il medesimo).

hCard e vCard sono basati su HTML, ponendo delle convenzioni su come gestire le informazioni.
I microdati usano tag speciali introdotti in

[1] `https://webdatacommons.org` è un archivio di crawl trovati sul web.

HTML5 per definire *itemscope*, *itemtype* e *itempropriety*: si definisce (*itemscope*) un oggetto di un certo tipo (*itemtype*) con determinate proprietà (*itemprop*), che in RDF si inseriscono tramite *black node*. I microdati sono parti di codice integrate dentro al testo, visualizzabili solamente nel codice sorgente. Si usano dei tipi definiti da vocabolari[2], utilizzabili anche in RDF, adottati come standard.
RDFa specifica all'inizio il vocabolario adottato, mentre il codice rimane scritto in HTML.
JSON-LD (JSON *Linked Data*) integra un file `.json` (in un tag `<script>`) in una pagina HTML per attribuire significati semantici. Non si ha bisogno di un parser apposito, e le triple sono facilmente ricostruibili a partire dalla pagina web. I motori di ricerca, processando in modo automatico, mostrano il contenuto in base al significato semantico.

4 SPARQL.

Introdotto come linguaggio per interrogare *linked data*, è un linguaggio simile a SQL caratterizzato (dalla versione 1.1) da quattro elementi:

- SPARQL Query;
- SPARQL Algebra;
- SPARQL Update;
- SPARQL Protocol.

Permette di interrogare (query), modificare (update) e integrare (federated) database composti da triple. Si definisce il concetto di *triple pattern*: si sostituiscono uno o più elementi di una tripla con una variabile.

```
dbpedia:The_Beatles foaf:name
                  "The Beatles" .

dbpedia:The_Beatles foad:name ?album .
?album mo:track ?track .
?album ?p ?o .
```

Si possono così costruire query strutturate grazie al concetto di *pattern match*: si definisce un sotto-grafo in cui fare interrogazioni più sofisticate.

```
// prologo: si definiscono le fonti
PREFIX dbpedia:
          <http://dbpedia.org/resource/>
PREFIX foaf:<http://xmlns.com/foaf/0.1/>
...

// query di tipo SELECT
// ASK | SELECT | DESCRIBE | CONSTRUCT
SELECT ?album
FROM <http://musicbrainz.org/20130302/>
WHERE {
  // il cuore della query:
  // il titolo di tutte le tracce degli
  // album dei Beatles
  dbpedia:The_Beatles foaf:made ?album .
  ?album a mo:Record ; dc:title ?title
}
ORDER BY ?title
```

`ASK` ritorna `true`/`false` se esiste una soluzione alla query:

```
ASK WHERE {
  dbpedia:The_Beatles mo:member
      dbpedia:Paul_McCartney .
}

=> true
```

`SELECT` ritorna una lista di elementi che hanno corrispondenza nel grafo:

```
SELECT ?album_name ?track_title
       ?date       ?album       .
WHERE {
  // tutti gli album dei Beatles
  dbpedia:The_Beatles foaf:name
                          ?album .
  ?album dc:title ?album ;
         mo:track ?track .
  // tutte le tracce degli album
  ?track dc:title ?track_title ;
         mo:duration ?duration ;
```

[2] `https://schema.org/` è un dizionario che definisce in modo non eccessivamente prescrittivo oggetti e proprietà.

```
  // filtra per durata
  FILTER (?duration > 300000 &&
          ?duration < 400000)
}
```

`CONSTRUCT` è simile a `SELECT` ma ritorna un grafo e non una lista:

```
CONSTRUCT {
  // riscrive con nuova terminologia
  ?album dc:creator
              dbpedia:The_Beatles
} WHERE {
  // esegue la query
  dbpedia:The_Beatles foaf:made
                          ?album .
  ?album mo:track ?track .
}
```

Le basi di conoscenza sono interrogate grazie ai vocabolari (anche detti *ontologie*), che permettono quindi la costruzione di query.

I dataset sono integrati grazie a predicati (`same`) che indicano quali entità si riferiscono allo stesso oggetto reale nelle varie fonti. SPARQL 1.1 permette di integrare un sistema di ragionamento automatico per l'uguaglianza delle entità (introduce il concetto di *entailment*).

Parte II
Esercitazioni.

1 SPARQL.

Si interroga `https://dbpedia.org` per ottenere una lista di vulcani:

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX dbo: <http://dbpedia.org/ontology/>

SELECT distinct ?v
  FROM <http://dbpedia.org>
 WHERE {
         ?v rdf:type dbo:Volcano .
       }
 LIMIT 100
```

I prefissi si aggiungono inserendo l'url tra simboli `<` e `>`. `rdf:type` può essere sostituito da una `a`:

```
?v a dbo:Volcano .
```

Una query SPARQL è strutturata specificando:

- prefissi
- tipo di query
- (fonte)
- (filtri)
- eventuali altre clausole

Le specifiche SPARQL possono essere ottenute all'indirizzo `https://www.w3.org/TR/rdf-sparql-query/`

Esercizio 1.

Si tenta di trovare il numero di dipendenti Google dalla pagina `http://dbpedia.org/page/Google`:

```
PREFIX res: <http://dbpedia.org/resource/>
PREFIX dbo: <http://dbpedia.org/ontology/>

SELECT ?num
  FROM <http://dbpedia.org>
 WHERE {
         res:Google dbo:numberOfEmployees ?num .
       }

# o più semplicemente:
SELECT ?num
  FROM <http://dbpedia.org>
 WHERE {
         dbr:Google dbo:numberOfEmployees ?num .
       }
```

Esercizio 2.

Si cerca di ottenere una lista di trombettisti leader di band:

```
SELECT ?person
 WHERE {
       # soggetto   proprietà       oggetto
         ?person    dbo:instrument  dbr:Trumpet .
         ?person    dbo:occupation  dbr:Bandleader .
       }
```

Esercizio 3.

Si cercano i Paesi con più di due grotte:

```
  SELECT ?country
   WHERE {
           ?cave     rdf:type      dbo:Cave .
           ?cave     dbo:location  ?country .
           ?country  rdf:type      dbo:Country .
         }
GROUP BY ?country
  HAVING (COUNT(?cave) > 2)
```

Esercizio 4.

Si vogliono trovare tutti i software sviluppati da società californiane.

```
SELECT DISTINCT ?software
 WHERE {
         ?company rdf:type dbo:Organisation .
         {           # dbpedia validata
           ?software dbo:developer ?company .
         } UNION {   # dbpedia da validare
           ?software dbp:developer ?company .
         }
         ?software rdf:type dbo:Software .
         ?company dbo:foundationPlace dbr:California .
       }
```

Esercizio 5.

Verificare se Natalie Portman è nata negli Stati Uniti:

```
  ASK
WHERE {
        dbr:Natalie_POrtman dbo:birthPlace ?city .
        ?city dbo:Country dbr:United_States .
      }
```

Esercizio 6.

Verificare se Roma è la capitale d'Italia:

```
  ASK
WHERE {
        dbr:Italy dbo:capital dbr:Rome .
      }
```

Esercizio 7.

Verificare se è disponibile la data di nascita di Elizabeth Taylor.

```
  ASK
WHERE {
        dbr:Elizabeth_Taylor dbo:birthDate ?_ .
      }
```

Esercizio 8.

Verificare se Milano è in Germania.

```
  ASK
WHERE {
        dbr:Germany dbo:city dbr:Miland .
      }
```

Esercizio 9.

Verificare se Michelle Obama è sposata con Barack Obama.

```
  ASK
WHERE {
        dbr:Michelle_Obama dbo:spouse dbr:Barack_Obama .
      }
```

Esercizio 10.

Si cercano i giocatori di Football americano che pesano più di 100kg.

```
CONSTRUCT {
            ?x rdf:type dbo:AmericanFootballPlayer .
          }
    WHERE {
            ?x dbo:weight ?kg .
            FILTER(?kg > 1000000000)
          }
```