Data Semantics

Sommario

La *Data Semantcs* si occupa di comprendere il significato dei dati durante la scrittura di un programma. Nell'ambito dell'analisi è fondamentale nell'integrazione di più dataset e nella loro condivisione. Altra problematica che la *Data Semantics* si pone di risolvere è l'organizzazione di dati non strutturati in modo da facilitarne l'analisi.

Scopo del corso è costruire dei modelli per attribuire valore semantico ai dati, in modo tale da facilitarne la fruizione; si tenta inoltre di capire il ruolo della semantica nell'integrazione dei dataset. Le finalità del corso rientrano nell'ambito dei *big data*, nella costruzione di *knowledge graphs* (ovvero la costruzione di relazioni interne a un database) e la costruzione di sistemi di raccomandazione. Saranno inoltre analizzate alcune tecniche di *natural language processing* per costruire rappresentazioni a partire dai dati.
Le esercitazioni si occuperanno di interrogare *knowledge graphs* 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 a scelta. La preparazione dovrà essere "ragionevolmente" dettagliata su tutti gli argomenti, e particolarmente approfondita sull'argomento del progetto.

Indice

Parte I
Grafi di conoscenza

Il termine è coniato da Google per indicare uno strumento usato dal suo motore di ricerca per fornire informazioni aggiuntive durante le ricerche. La struttura a grafo permette di essere processata facilmente da una macchina e allo stesso tempo garantisce un livello di astrazione soddisfacente; si possono anche effettuare query come in un database a grafo (Neo4j). Il modello a grafo permette inoltre una facile integrazione di sorgenti diverse rendendo naturali collegamenti e relazioni. Il web *semantico* ha costruito linguaggi e strumenti, approvati dal W3C, per definire, interrogare e fare inferenza su grafi di conoscenza; tuttavia nel mondo reale questi strumenti non hanno largo utilizzo. La costruzione di grafi di conoscenza è spesso effettuata a mano da una moltitudine di utenti (chiedendo recensioni tramite *app*).

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*). Tali dati possono essere sfruttati nella costruzione di intelligenze artificiali, ovvero di programmi che eseguono task tipicamente umani con risultati simili. In particolare è così possibile effettuare compiti particolarmente ardui per un umano, come l'integrazione di serie storiche con fonti multiple appartenenti a domini vari o poco noti.

Esistono grafi di conoscenza *open*, quali DBpedia, Yago o Wikidata, ma alcune aziende private sviluppano il proprio grafo per facilitare l'attività imprenditoriale.

1 Spazio vettoriale.

In un grafo di conoscenza, query e documenti sono interpretabili come vettori: per calcolarne la similarità 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).

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

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.

2 *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.

3 Standard.

RDF (*Resource Description Framework*) 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, gli URI, appunto): si usano generalmente identificativi web (abbreviati, detti prefissi), 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 *blank node* (ovvero costanti *locali*), il predicato può essere composto solo da un URI mentre gli oggetti possono essere URI, *blank nodes* oppure letterali, che possono essere *plain literal*, ovvero letterali puri, oppure *typed literal*, a cui è attribuito un tipo (stringhe, date o booleani) per permettere operazioni. I *blank 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

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

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 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 *blank 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 foaf: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 .
         }
```

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

```
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 {
         dbpedia:The_Beatles foaf:name ?album .              # tutti gli album
                                                             # dei Beatles
         ?album              dc:title  ?album .
         mo:track ?track .                                   # tutte le tracce
                                                             # degli album
         ?track              dc:title  ?track_title .
         mo:duration ?duration .
         FILTER (?duration > 300000 && ?duration < 400000)   # filtra per durata
       }
```

`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 as`) 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*).

5 Vocabolari (o ontologie).

In RDF si ha un modello dei dati che prevede di utilizzare URI per rappresentare sia le risorse sia le proprietà. L'uso di URI per specificare proprietà e relazioni definiscono separatamente l'insieme di proprietà e tipi usati per descrivere un dominio: si definisce così un vocabolario per descrivere i dati, che permettono una standardizzazione della tipizzazione del dato. Un esempio è il predicato `rdf:type` introduce un vocabola-

rio che permette di catalogare risorse definendo i tipi di variabile. Si definiscono quindi ontologie comuni sfruttabili da più *data provider* per uniformare la rappresentazione dei dati.

Per definire un vocabolario si definiscono prima le classi (ovvero i tipi di dato) con proprietà, eventualmente diverse (ovvero usando certe proprietà solo su certe classi).

Uno dei vocabolari più importanti (perché introdotto da tempo) è FOAF (*Friend of a Friend*); si ricordano anche DC (*Dublin Core*), che specifica una serie di predicati per rappresentare i documenti, e SKOS (*Simple Knowledge Organizating System*). Più vocabolari possono riferirsi allo stesso oggetto e per definizione non è lecito unificare i vocabolari in quanto dovrebbero adattarsi alla specificità del dominio. Alcuni dei più grandi provider internet hanno però unificato vocabolari col progetto `schema.org`, molto utilizzato nel web e in continua evoluzione. Nel caso siano insufficienti, è opportuno espandere un dizionario già esistente piuttosto che crearne uno nuovo dal nulla. Si possono unire più vocabolari grazie a collegamenti, a livello di istanza o di schema.

Un computer ha bisogno di definire un sistema per attribuire un significato al significante. Si possono definire tre[3] sistemi per definire un'ontologia, ovvero un sistema per definire gli oggetti esistenti.

L'uso di sistemi diversi rende complicata l'integrazione di più fonti.

5.1 Tesauri.

Si definisce un dizionario di entità linguistiche, composto dalle parole proprie della lingua e da un numero fisso di relazioni tra di loro. Generalmente definisce sostantivi (semplici o composti), aggettivi e verbi. Le relazioni lessicali sono:

- sinonimia: più parole con significati simili;
- antinomia: più parole hanno significato opposto;
- polisemia: più significati della stessa parola;
- iponimia (e iperonimia): quando un significato rappresenta un sottoinsieme dell'altro (casa - costruzione);
- associazione: più parole strettamente legate tra di loro.

Tuttavia esistono dei problemi: i sinonimi non sono mai perfettamente interscambiabili e l'iponimia (o iperonimia) sottintendono molte sfumature.

WordNet è un vasto database lessicale sviluppato dagli anni '90 che rappresenta i significati (o concetti) in una struttura ad albero. La polisemia è gestita definendo più entità diverse con lo stesso nome. Data una parola, si ha quindi un insieme finito di significati distinti tra di loro.

5.2 Tassonomia.

Si definisce un grafo di classificazione, generalmente con struttura ad albero. Le tassonomie sono strutture usate in informatica e nello specifico in Data Science, prese in prestito dalla biologia (per classificare le forme viventi), in cui tuttavia sono presenti dibattiti. Si usa una forma ad albero per facilitare la condivisione tra varie fonti. La valenza può anche non essere perfetta ma ha grande valenza in ambito scientifico ed economico. Il significato è spiegato in classi, superclassi e sottoclassi grazie al meccanismo dell'ereditarietà. È necessario specificare anche quale criterio usare per effettuare la divisione dell'albero. Si può anche effettuare una ripartizione dell'oggetto dividendolo nei sotto-oggetti specifici che lo compongono.

5.3 Ontologie assiomatiche.

Il significato è attribuito attraverso una serie di assiomi logici per specificare le definizioni dei termini utilizzati con riferimento alle implicazioni. Si definisce quindi un linguaggio L di assiomi logici con una semantica formale che determina il significato in modo univoco.

[3] Esiste un quarto metodo che sarà approfondito più avanti.

6 Inferenza nell'ontologia.

Definendo un oggetto, si hanno sempre delle implicazioni: è necessario fare inferenza, ovvero si procede con un procedimento deduttivo. Dal principio dell'informatica, l'uso della logica (proposizionale e del primo ordine) permette la deduzione da parte di calcolatori. Oltre all'induzione classica, esistono altre pratiche per fare inferenza. Le implicazioni fatte sulla base di ciò che è definito nel dizionario, dal punto di vista inferenziale, sono la semantica reale delle informazioni. L'inferenza ha come obiettivi di ridurre la complessità del sistema trovando un modo per ottenere informazioni velocemente e per aumentare la flessibilità dello schema per poter essere aggiornato in tempo reale con le modifiche dell'ambiente.

6.1 Schema dell'ontologia.

Lo schema prescrive come deve essere organizzato il dato per entrare nel database; tuttavia i dati sono flessibili per loro natura, dunque l'operazione non è immediata. È necessario definire uno schema per dare una struttura ai dati: non è lecito usare nell'approccio semantico un database NoSQL troppo flessibile. Rispetto all'approccio relazionale, lo schema semantico è più flessibile: SQL non permette di modificare la struttura una volta dichiarata. Si procede dunque con un approccio *deduttivo* (*validazione indiretta*): anzichè specificare qual è la struttura del dato, si modifica lo schema al bisogno. Lo schema quindi è flessibile e disaccoppia lo schema dai dati (ovvero lo schema può essere modificato successivamente).

Potendo esserci contraddizione nei dati, si sono costruiti linguaggi come RDF Schapes (SHACL) che permettono una validazione dei dati.

6.2 *Reasoning.*

I linguaggi RDFS e OWL permettono l'operazione di *reasoning*. OWL è costruito su RDFS, aggiungendo però funzionalità aggiuntive. Oltre ad avere sistemi deduttivi, la logica proposizionale garantisce che l'algoritmo termini (è cioè *decidibile*, ovvero date le premesse è sempre possibile stabilire se una Formula Ben Formata FBF è vera o meno), mentre la logica del primo ordine è semi-decidibile. Anche con una procedura che termina e confermi la verità di una formula però non è utile in pratica: il tempo impiegato potrebbe essere particolarmente lungo e quindi rendere inutile il programma. In caso di linguaggi non decidibili, bisogna semplificare il linguaggio per trovare un compromesso tra tempo impiegato ed espressività: ci sono più linguaggi perchè il compromesso è trovato in modo diverso. Inoltre esistono altri linguaggi *a regole* utilizzabili per fare deduzioni su RDF.

6.2.1 RDF Schema.

RDFS costruisce un *reticolo* che parte da un nodo e garantisce ereditarietà (anche multipla) delle classi: è costruito un *ordine parziale*. RDFS introduce implicazioni sul tipo di dato: si introduce un approccio secondo cui il mondo è composto da *classi* e *individui*; quindi si hanno insiemi ed elementi degli insiemi. Una classe è composta da individui, i quali tramite *proprietà* entrano in relazione tra di loro. Questo approccio è coerente con il modello relazionale, e caratterizza in modo molto forte la semantica dei dati strutturati. Il concetto di classe vincola l'uso del tipo: una classe non può essere del tipo di un'altra.

I letterali sono divisi esplicitamente dagli URI.

Con RDFS si possono rappresentare e vincolare relazioni tra classi, organizzandole in gerarchie: si organizzano così tassonomie che garantiscono nativamente inferenze semplici (con proprietà transitiva). Anche le proprietà possono essere organizzate gerarchicamente, con proprietà che derivano da altre (`figlioDi` è sotto-proprietà di `parenteDi`).

La semantica dei predicati è definita da regole, che determinano l'ereditarietà (classi derivate), la transitività di classi e relazioni e simili.

I predicati `rdfs:domain` e `rdfs:range` indicano che tutti gli oggetti di una classe appartengono anche a un'altra classe. Questi due predi-

cati non sono usati in `schema.org` perché troppo rigidi.

Lo schema non può dedurre contraddizioni, dunque.

6.2.2 SPARQL 1.1.

La versione 1.1 introduce gli *Entailment Regimes*: procedure che calcolano tutte le implicazioni degli assiomi; uno dei sistemi più noti è Virtuoso.

6.2.3 OWL.

È un motore inferenziale composto da più sottolinguaggi, con l'obiettivo di ridurre la confusione lasciata da RDFS. OWL2 è diviso in tre progetti:

- OWL 2 EL: limitato a classificazioni di base, ma la complessità è polinomiale;
- OWL 2 QL: usa una sintassi simile a SQL;
- OWL 2 RL: è usato per essere implementato efficientemente in sistemi *rule-based*.

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)
          }
```