Semantic Search (Part II)

[DAT640] Information Retrieval and Text Mining

Krisztian Balog
University of Stavanger

October 15, 2019

Recap

- Semantic search
  - "Search with meaning" (beyond literal matches)
  - Revolves around entities
- Knowledge bases
  - Organize information around entities using the RDF data model
  - Each entity is uniquely identified by its URI (Uniform Resource Identifier) and its properties are described in the form of subject-predicate-object (SPO) triples

Exercise #1

- Fetching data from Wikipedia and DBpedia
- Code skeleton on GitHub: `exercises/lecture_14/exercise_1.ipynb` (make a local copy)

Ad hoc entity retrieval

Entity retrieval is the task of answering queries with a ranked list of entities[1]

Definition

Given a keyword query q and an entity catalog $\mathcal{E}$, *ad hoc entity retrieval* is the task of returning a ranked list of entities $\langle e_1, \ldots, e_k \rangle, e_i \in \mathcal{E}$ with respect to each entity's relevance to q. The relevance of entities is inferred based on a collection of unstructured and/or (semi-)structured data.

[1] Ad hoc refers to the standard form of retrieval in which the user, motivated by an ad hoc information need, initiates the search process by formulating and issuing a query

Example queries

martin luther king
disney orlando
Apollo astronauts who walked on the Moon
Winners of the ACM Athena award
EU countries
Hybrid cars sold in Europe
birds cannot fly
Who developed Skype?
Which films starring Clint Eastwood did he direct himself?

Main strategy

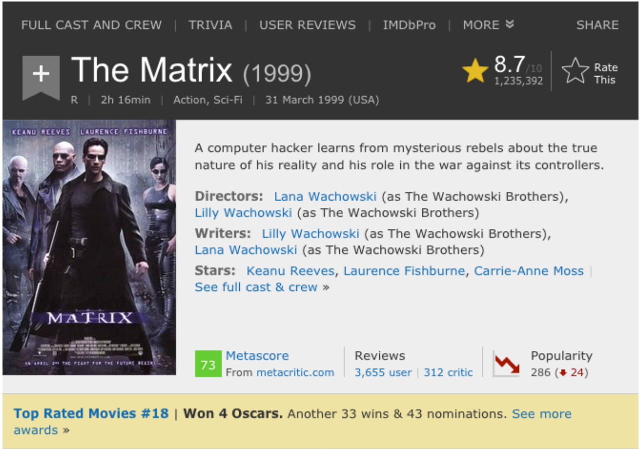
- Build on work on document retrieval
- Create and entity description or “profile” document is to be compiled for each entity in the catalog
  - Specifically, a fielded *entity document*
- Those entity description documents can be ranked the same way as documents

Constructing term-based entity representations

From semi-structured documents

- E.g., Wikipedia article, IMDB page, LikedIn profile, ...
- Field content is typically extracted using wrappers (template-based extractors)

Example



FULL CAST AND CREW | TRIVIA | USER REVIEWS | IMDbPro | MORE
SHARE
+
Rate This
KEANU REEVES LAURENCE FISHBURNE
MATRIX

The Matrix (1999)

8.7/10
1,235,392

R | 2h 16min | Action, Sci-Fi | 31 March 1999 (USA)

A computer hacker learns from mysterious rebels about the true nature of his reality and his role in the war against its controllers.

Directors: Lana Wachowski (as The Wachowski Brothers), Lilly Wachowski (as The Wachowski Brothers)

Writers: Lilly Wachowski (as The Wachowski Brothers), Lana Wachowski (as The Wachowski Brothers)

Stars: Keanu Reeves, Laurence Fishburne, Carrie-Anne Moss | See full cast & crew »

73 Metascore From metacritic.com | Reviews 3,655 user | 312 critic | Popularity 286 (↓ 24)

Top Rated Movies #18 | **Won 4 Oscars.** Another 33 wins & 43 nominations. See more awards »

Figure: Web page of the movie The Matrix from IMDb (`http://www.imdb.com/title/tt0133093/`).

Example

Name	The Matrix
Genre	Action, Sci-Fi
Synopsis	A computer hacker learns from mysterious rebels about the true nature of his reality and his role in the war against its controllers.
Directors	Lana Wachowski (as The Wachowski Brothers), Lilly Wachowski (as The Wachowski Brothers)
Writers	Lilly Wachowski (as The Wachowski Brothers), Lana Wachowski (as The Wachowski Brothers)
Stars	Keanu Reeves, Laurence Fishburne, Carrie-Anne Moss
Catch-all	The Matrix Action, Sci-Fi A computer hacker learns from mysterious rebels about the true nature of his reality and his role in the war against its controllers. Lana Wachowski (as The Wachowski Brothers), Lilly Wachowski (as The Wachowski Brothers) Lilly Wachowski (as The Wachowski Brothers), Lana Wachowski (as The Wachowski Brothers) Keanu Reeves, Laurence Fishburne, Carrie-Anne Moss

Discussion

Question

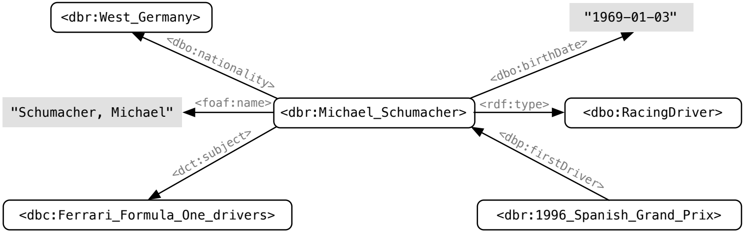
What is the role of the catch-all field?

Catch-all field

- Amasses the contents of all fields
    - Can help to quickly filter entities (e.g., in first-pass retrieval)
    - Fields are often sparse; combining field-level scores with an entity-level ("catch-all" score) often improve performance

From structured knowledge bases

- Assemble text from all SPO triples that are about a given entity
  - Note that the entity may also stand as object



Discussion

Question

How to turn SPO triples into a fielded document?

Issue #1

- The number of potential fields is huge (in the 1000s)
  - The representation of an entity is sparse (each entity has only a handful of predicates)
  - Estimating field weights becomes problematic
- Solution: *predicate folding*
  - Grouping predicates together into a small set of predefined categories
  - Grouping may be based on predicate type or (manually determined) importance

Commonly used fields

- **Name** contains the name(s) of the entity
  - The two main predicates mapped to this field are `<foaf:name>` and `<rdfs:label>`
  - One might follow a simple heuristic and additionally consider all predicates ending with "name," "label," or "title"
- **Name variants** (aliases) may be aggregated in a separate field
  - In DBpedia, such variants may be collected via Wikipedia redirects (via `<dbo:wikiPageRedirects>`) and disambiguations (using `<dbo:wikiPageDisambiguates>`)
- **Attributes** includes all objects with literal values, except the ones already included in the name field
  - In some cases, the name of the predicate may also be included along with the value, e.g., "founding date 1964" (vs. just the value part, "1964")

Commonly used fields (2)

- **Types** holds all types (categories, classes, etc.) to which the entity is assigned
  - Commonly, `<rdf:type>` is used for types
  - In DBpedia, `<dct:subject>` is used for assigning Wikipedia categories, which may also be considered as entity types
- **Outgoing relations** contains all URI objects, i.e., names of entities (or resources in general) that the subject entity links to
  - If the *types* or *name variants* fields are used then those predicates are excluded
  - Values might be prefixed with the predicate name, e.g., "spouse Michelle Obama"
- **Incoming relations** is made up of subject URIs from all SPO triples where the entity appears as object
- **Top predicates** may be considered as individual fields
  - E.g., top-100 most frequent DBpedia predicates
- **Catch-all** is a field that amasses all textual content related to the entity

Issue #2

- Object values are either URIs or literals
- While literals can be treated as regular text, URIs are not suitable for text-based search
  - Some URIs are “user-friendly”: `http://dbpedia.org/resource/Audi_A4`
  - Others are not: `http://rdf.freebase.com/ns/m.030qmx`
- *URI resolution* is the process of finding the corresponding human-readable name/label for a URI

URI resolution

- Goal: find the name/label for a URI
- The specific predicate that holds the name of a resource depends on the RDF vocabulary used
    - Commonly, `<foaf:name>` or `<rdfs:label>` are used
- Given an SPO triple, for example

```
<dbr:Audi_A4> <rdf:type> <dbo:MeanOfTransportation>
```

- The corresponding resources's name is contained in the object element of this triple:

```
<dbo:MeanOfTransportation> <rdfs:label> "mean of transportation"
```

Example

Name	Audi A4
Name variants	Audi A4 … Audi A4 Allroad
Attributes	The Audi A4 is a compact executive car produced since late 1994 by the German car manufacturer Audi, a subsidiary of the Volkswagen Group [...] … 1996 … 2002 … 2005 … 2007
Types	Product … Front wheel drive vehicles … Compact executive cars … All wheel drive vehicles
Outgoing relations	Volkswagen Passat (B5) … Audi 80
Incoming relations	Audi A5
`<foaf:name>`	Audi A4
`<dbo:abstract>`	The Audi A4 is a compact executive car produced since late 1994 by the German car manufacturer Audi, a subsidiary of the Volkswagen Group [...]
Catch-all	Audi A4 … Audi A4 … Audi A4 Allroad … The Audi A4 is a compact executive car produced since late 1994 by the German car manufacturer Audi, a subsidiary of the Volkswagen Group [...] … 1996 … 2002 … 2005 … 2007 … Product … Front wheel drive vehicles … Compact executive cars … All wheel drive vehicles … Volkswagen Passat (B5) … Audi 80 … Audi A5

Exercise #2

- Indexing DBpedia data
- Code skeleton on GitHub: `exercises/lecture_14/exercise_2.ipynb` (make a local copy)

Reading

- Entity-Oriented Search (Balog)[2]
    - Chapter 3

[2] PDF: https://rd.springer.com/content/pdf/10.1007%2F978-3-319-93935-3.pdf