Question Answering on Knowledge Graphs

Part 1.3 of the Tutorial
"Neuro-Symbolic Representations for Information Retrieval"
ECIR 2023, Dublin (April 6)

Hannah Bast
Chair of Algorithms and Data Structures
University of Freiburg

UNI FREIBURG

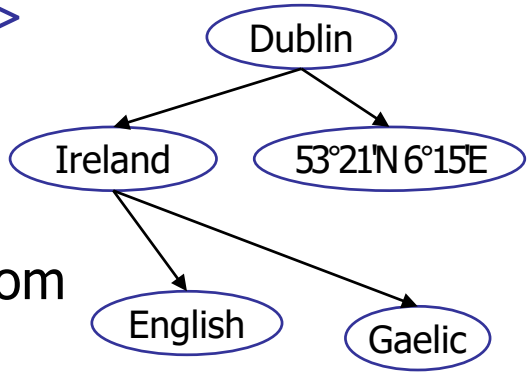
Knowledge Graphs 1/8

UNI FREIBURG

- Resource Description Framework
  - RDF is based on a strikingly simple principle: ALL DATA, whatever it is, is modelled as a **set of triples**

    subject predicate object
  - For example:

<Dublin>	<capital of>	<Ireland>
<Dublin>	<coordinates>	"53°21'N 6°15'E"
<Ireland>	<language>	<English>
<Ireland>	<language>	<Gaelic>



Graph view: each distinct subject or object is a **node**, each triple is a **directed edge** from subject to object, with the predicate as label

Knowledge Graphs 2/8

wdt:P17 ~ country
wd:Q27 ~ Ireland
wdt:P625 ~ coordinates

UNI FREIBURG

- International Resource Identifier (IRI)
  - Names have globally unique identifiers, so-called IRIs

    An IRI is like an address in a browser (URL), only that Unicode characters like ä, ö, ü, … are allowed
  - The IRI is often not human-readable but ID-like; instead there are dedicated triples for its names and aliases
  - For example, the Wikidata IRI for Dublin is

    <https://www.wikidata.org/wiki/Q1761>

    And some example triples (with **IRI prefixes**) look like this:

    wd:Q1761 **rdfs:**label "Baile Átha Cliath"@ga
    wd:Q1761 **wdt:**P17 **wd:**Q27
    wd:Q1761 **wdt:**P625 "Point(53.35 -6.26)"^^**geo:**wktLiteral

Knowledge Graphs 3/8

- Some widely used knowledge graphs
  - **Wikidata:** Started 2012, successor of Freebase (bought by Google in 2010 for 99M $), crowd-sourced, amazing coverage

    18 **B** triples, 99 M entities, 10K predicates
  - **UniProt:** Started 2002, protein sequences, genes, all kinds of metadata, …

    94 **B** triples, 16 B entities, 520 predicates
  - **PubChem:** Started 2004, chemical compounds, substances, proteins, genes + interrelations

    124 **B** triples, 4 B entities, 414 predicates

Knowledge Graphs 4/8

- And more ...
  - **OpenStreetMap:** Started 2004, all the geo-data of the world, crowd-sourced, amazing coverage and quality

    11 B triples, 1.1 B entities, 87K predicates

  - **DBLP:** Started 1993, publication meta-data for computer science and adjacent fields

    1 B triples, 150 M entities, 75 predicates

dblp

Many more datasets available as RDF, of high quality and maintained with great care

Knowledge Graphs 5/8

- Historical knowledge graphs
  - **Freebase:** Started 2007 by Metaweb, acquired by Google in 2010, closed down in 2015 (migrated to Wikidata)

    3.1 B triples, 47 M entities, 785 K predicates
  - **YAGO:** first released 2008 at the MPI for Informatics, derived from Wikipedia info-boxes combined with WordNet

    121 M triples, 15 M entities, 100 predicates
  - **DBpedia**: all kinds of structured data extracted from Wikipedia

    These are still mentioned in a lot in papers (because benchmarks were based on them)

Knowledge Graphs 6/8

UNI FREIBURG

- The standard query language for RDF is
  - SPARQL is a variant of SQL, adapted to the (simple) RDF

```
SELECT ?title ?author ?year WHERE {
    ?paper dblp:title ?title .
    ?paper dblp:authoredBy ?author.
    ?paper dblp:yearOfPublication ?year .
    FILTER (?year <= 1940)
}
```

All papers in DBLP published before 1940

Query on QLever

  - The result of a SPARQL query is always a **table**, in the example:

    All assignments to ?title ?author ?year such that the triples exist in the knowledge graph and the FILTER condition is true

    Note: if a paper has **k** authors, there will be **k** rows for it

Knowledge Graphs 7/8

wd:Q183 ~ Germany
osmrel:51477 ~ Germany
wd:P37 ~ population
wdt:Q9142 ~ Gaelic

UNI FREIBURG

- Interoperability
  - One of the great strengths of RDF and SPARQL is the great ease, with which different datasets can be **combined**

    For standard databases, this is a nightmare: different and often complex schemas, different identifiers, etc.

    In RDF, all you need are additional triples relating the IDs from the two datasets you want to combine

osmrel:62773	**osm:**wikidata	**wd:**Q27
osmrel:62773	**geo:**hasGeometry	"MULTIPOLYGON(…)"^^**geo:**…
wd:Q27	**wdt:**P37	**wdt:**Q9142

    The first two triple are from OSM, the third is from Wikidata
    (OSM knows the geometric shapes, Wikidata knows the language)

Knowledge Graphs 8/8

- A variety of example queries
  - Birth places of people with first name X — Wikidata
  - Notable events that happened on April 6 — Wikidata
  - Genes for human diseases — UniProt
  - NSAID drugs with small molecular weight — PubChem
  - All streets in Ireland — OpenStreetMap
  - All countries with official language X — Wikidata+OSM
  - Average number of authors per year — DBLP

Side note: These are all running with the same system, with very little configuration per dataset, that is the power of RDF+SPARQL

SPARQL Autocompletion 1/4

UNI FREIBURG

- Finding the right SPARQL query is hard
  - Consider the following simple search request

    Oscars of Meryl Streep with corresponding movie

  - The result we are looking for is

Academy Award for Best Supporting Actress	Kramer vs. Kramer
Academy Award for Best Actress	Sophie's Choice
Academy Award for Best Actress	The Iron Lady

  - On the next slide, you see the correct SPARQL query on Wikidata

SPARQL Autocompletion 2/4

```
PREFIX rdfs:  <http://www.w3.org/2000/01/rdf-schema#>
PREFIX wdt:   <http://www.wikidata.org/prop/direct/>
PREFIX pq:    <http://www.wikidata.org/prop/qualifier/>
PREFIX ps:    <http://www.wikidata.org/prop/statement/>
PREFIX p:     <http://www.wikidata.org/prop/>
PREFIX wd:    <http://www.wikidata.org/entity/>
SELECT ?movie ?award WHERE {
  wd:Q873     p:P166     ?statement .
  ?statement  ps:P166    ?award_id .
  ?statement  pq:P1686   ?movie_id .
  ?award_id   wdt:P31    wd:Q19020 .
  ?award_id   rdfs:label ?award . FILTER (LANG(?award) = "en")
  ?movie_id   rdfs:label ?movie . FILTER (LANG(?movie) = "en")
}
```

SPARQL Autocompletion 3/4

UNI FREIBURG

- What's hard about finding this query?
  - Knowing the right prefix definitions

    PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
  - Knowing the right entity names

    wd:Q873 ("Meryl Streep"), wd:Q19020 ("Academy Award")
  - Knowing the right predicate names **very hard, even for experts**

p:P166	"won award"	from awardee to statement
ps:P166	"won award"	from statement to award entity
pq:P1686	"for work"	from statement to movie
wdt:P31	"instance of"	from "instance" entity to "class" entity
  - Knowing the syntax for filtering by language

    FILTER (LANG(?award) = "en")

SPARQL Autocompletion 4/4

UNI FREIBURG

- Let's look at two query building tools
  - **Tool 1:** Wikidata's own query builder

    A simple (context-insensitive) autocompletion

    Requires **deep expert knowledge** of the prefixes and the right predicate names
  - **Tool 2:** QLever's query builder

    Context-sensitive suggestions as you type

    Requires only relatively basic knowledge of RDF/SPARQL, and little or no knowledge about the dataset

    Fun fact: the suggestions are themselves computed via SPARQL queries, on the same knowledge graph on which we are trying to formulate a query

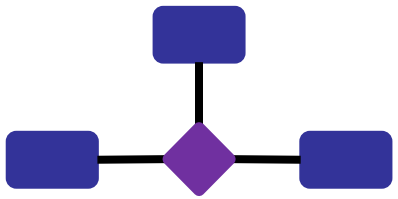
QA on Knowledge Graphs 1/6

- Goal
  - Ask questions in **natural language**:

    Which character did Ellen DeGeneres play in Finding Nemo?
  - Or even more informally / telegraphically:

    who did ellen play in finding nemo?
  - **Goal:** automatically translate such a natural-language or keyword question into the corresponding SPARQL query

```
SELECT  ?x  WHERE {
  ?m   actor       Ellen DeGeneres .
  ?m   film        Finding Nemo .
  ?m   character   ?x
}
```



Natural-Language Queries 2/6

- Challenge 1: Linguistic variation
  - The same question can be asked in dozens of ways:

    which character did ellen degeneres play in finding nemo

    which character did ellen play in finding nemo

    who did ellen play in finding nemo

    ellen's role in finding nemo

    whose voice did ellen do in finding nemo

    role ellen nemo

    …

    This rules out simple pattern-based approaches

Natural-Language Queries 3/6

- Challenge 2: Ambiguous entity names
  - Ellen could mean

    Ellen DeGeneres

    Ellen Page

    Ellen Burstyn

    anyone called "Ellen"

    The Ellen Show

    The Ellen DeGeneres Show

    ...

    Over 100 different entities named "ellen" in Freebase

Natural-Language Queries 4/6

UNI FREIBURG

- Challenge 3: Ambiguous relation names
  - Like for entity names, but worse, because the relation can be implicit in the question, for example:

Question: who is the ceo of apple

Query:

```
SELECT  ?x  WHERE {
   ?m   job-title    "Managing Director" .
   ?m   company  "Apple Inc." .
   ?m   person     ?x .
}
```

None of the relation words "job title", "company", "person" appear in the question … nor synonyms of them

Natural-Language Queries 5/6

- General Approach 1: Generate candidate +rank them
  - Step 1: Entity linking in the query

    No need to commit to a perfect solution at this stage, having multiple candidates is OK
  - Step 1: Generate candidate queries

    Generate all possible SPARQL queries that connect those entity (candidates)
  - Step 3: Rank these candidates by their similarity to the question using a learned model

    Can be trained only from question-answer pairs (no need to have also have the correct SPARQL query for each question)

Natural-Language Queries 6/6

- General Approach 2: Use a language model
  - Step 1: Write a prompt

    Write a SPARQL query to the Wikidata Query Service for the following question: Which Oscars did Meryl Streep win and for which movie?

  - Step 2: Send the query

    Send the query to the respective API

    For example: OpenAI Codex

    A descendant of GPT-3 trained on both natural language and code, see https://openai.com/blog/openai-codex

References

- Links
  - https://query.wikidata.org
  - https://github.com/ad-freiburg/qlever
  - https://qlever.cs.uni-freiburg.de
  - https://aqqu.cs.uni-freiburg.de
- Papers
  - TODO: will be added after the tutorial