Knowledge Graphs & Entity Linking

(and their relation to LLMs)

Session 1 of the SIGIR'23 Tutorial
"Neuro-Symbolic Representations for IR"
Taipei, Taiwan, July 23, 2023

Hannah Bast
Chair of Algorithms and Data Structures
University of Freiburg, Germany

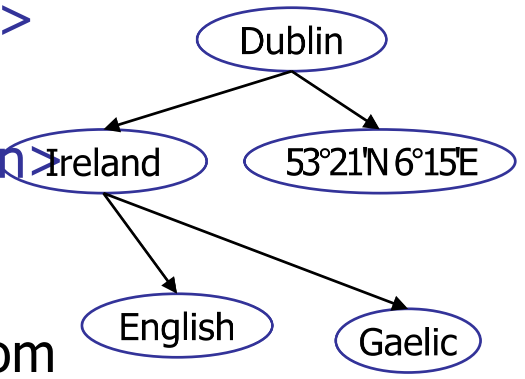
Knowledge Graphs 1/8

- 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:

<Taipei>	<capital of>	<Taiwan>
<Taipei>	<coordinates>	"25°02'N 121°34'E"
<Taiwan>	<language>	<Hokkien>
<Taiwan>	<language>	<Hakka>
<Taiwan>	<language>	<Mandarin>



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:Q865 ~ Taiwan
wdt:P625 ~ coordinates

- 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 Taipei is

    <https://www.wikidata.org/wiki/Q1867>

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

wd:Q1876	**rdfs:**label	"Taipeh"@de
wd:Q1876	**wdt:**P17	**wd:**Q865
wd:Q1876	**wdt:**P625	"Point(25.03 121.57)"^^**geo:**wktLiteral

Knowledge Graphs 3/8

- Some widely used knowledge graphs
  - **Wikidata:** Started 2013, successor of Freebase (bought by Google in 2010 for 99M $), crowd-sourced, amazing coverage

    18 **B** triples, 52 131 predicates
  - **UniProt:** Started 2002, protein sequences, genes, all kinds of metadata, …

    110 **B** triples, 310 predicates
  - **PubChem:** Started 2004, chemical compounds, substances, proteins, genes + interrelations

    124 **B** triples, 426 predicates

Knowledge Graphs 4/8

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

    14 **B** triples, 92 704 predicates

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

    0.8 **B** triples, 75 predicates

  - Most of the IT-heavy companies maintain their own (very large) knowledge graphs:

    Google, Amazon, Microsoft, Meta, Bloomberg, Walmart, Airbnb, …

Knowledge Graphs 5/8

- Historical knowledge graphs
  - **Freebase:** Started 2007, acquired by Google in 2010, closed down in 2015

    3 B triples, 784 977 predicates
  - **YAGO:** Started 2008, derived from Wikipedia info-boxes and WordNet

    0.1 B triples, 100 predicates
  - **DBpedia**: all kinds of structured data extracted from Wikipedia

    0.8 B triples, 54 780 predicates

Al three are still mentioned in a lot in papers (because bench-marks were based on them and academia is slow to adapt)

Knowledge Graphs 6/8

- The standard query language for RDF is SPARQL
  - 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

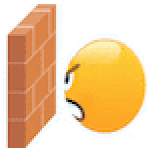
Knowledge Graphs 7/8

This allows "federated" SPARQL queries like

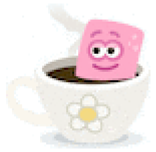
all power lines in the EU

- 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



wd:Q183 **wdtn:**P402 **osmrel:**51477 [in Wikidata]

osmrel:51477 **osm**:wikidata **wd:**Q183 [in OpenStreetMap]

wd:Q183	IRI for Germany in Wikidata
osmrel:51477	IRI for Germany in OpenStreetMap
wdtn:P402	predicate for "OpenStreetMap IRI" in Wikidata
osm:wikidata	predicate for "Wikidata IRI" in OpenStreetMap

Knowledge Graphs 8/8

- A variety of example queries

– Birth places of people with first name X	Wikidata
– Notable events that happened on July 23	Wikidata
– Proteins with organism and amino sequence	UniProt
– NSAID drugs with small molecular weight	PubChem
– All streets in X	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

Finding the right SPARQL query is hard 1/4

- Example question
  - 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

Finding the right SPARQL query is hard 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")
}
```

Finding the right SPARQL query is hard 3/4

- 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")

Finding the right SPARQL query is hard 4/4

- A technical solution: 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)

Knowledge Graphs and LLMs 1/3

- Ask questions directly in natural language
  - Let's try the following question on ChatGPT

    Which Oscars did Meryl Streep win and for which movies?

    Alas, it works like a charm

    So does this mean that we don't need knowledge graphs any longer, but we can just ask our LLM anything?

Knowledge Graphs and LLMs 2/3

- Ask questions directly in natural language
  - Let's try a slightly more difficult question

    First names and how common they are

    Older versions of ChatGPT provide wrong answers, newer versions realize that they cannot now

    A Large Language Model is no substitute for a database or knowledge graph for at least two reasons:

    1. Storing large amounts of information in a LLM is possible, but inefficient (think of the human brain)

    2. Some queries require a large "working memory", which LLMs do not have by design

Knowledge Graphs and LLMs 3/3

- Use LLMs to help formulate SPARQL queries
  - Let's try the following

    Give me the SPARQL query on Wikidata for all people with first name X and their birth place with coordinates

    ChatGPT gets the query almost right, but the IRI for the first name is wrong

    Again, it's hard (and inefficient) for a LLM to store large amounts of relational data, like which entity has which IRI in Wikidata

    If we give ChatGPT a hint about the right IRI, it gives us the correct query

Entity Linking 1/7

- Definition
  - Entity Recognition (ER): identify passages in a text that refer to an entity from a given knowledge graph
  - Entity Disambiguation (ED): identify, exactly which entity the passage refers to
  - Entity Linking (EL): Entity Recognition + Disambiguation

    American athlete Whittington failed to appear in the 2013–14 season due to a torn ACL.

    American [Q30] athlete **Whittington** [Q21066526] failed to appear in the **2013–14 season** [Q16192072] due to a **torn ACL** [Q18912826].

Entity Linking 2/7

- Research on Entity Linking
  - There are thousands of papers on the problem

    Top venues ER   Top venues ED   Top venus EL
  - Almost all of these papers claim to improve on previous work and have an evaluation to prove it
  - But when you actually try the methods in practice, you frequently make one of the following two experiences:

    1. There is no software or it does not run (anymore)
    2. Lot of hyperparameters and results cannot be replicated
    3. Poor results on datasets not evaluated in the paper

    What are the reasons for this?   See the following slides

Entity Linking 3/7

- Reason 1: Coarse evaluation metrics
  - The typical evaluation measures **precision**, **recall**, **F1**
  - That is fine to get an initial impression of an approach
  - But it is very prone to overfitting, especially for benchmarks that have been around a long time
  - An absolute must or practically useful entity linking

    1. Look at individual results
    2. Detailed error analysis

Entity Linking 4/7

- Reason 2: Benchmarks artifacts and biases
  - The following are very frequent in existing benchmarks:

    1. Almost exclusive focus on **named entities**

    Named entities are almost always capitalized in the English language and hence easy to recognize

    2. When going beyond named entities, what is an entity?

    No so easy: for example, if everything is an entity that has a Wikipedia article, then almost each piece of text qualifies

    3. Many entity mentions are ambigious

    For example, "American" → Q30 or Q7976 or Q846570 ?

    4. Many benchmarks have a bias towards types of entites

    For example, AIDA-CoNNL: many entities are sports teams

Entity Linking 5/7

- A general-purpose analysis tool
  - **ELEVANT**: **E**ntity **L**inking **Ev**aluation and **An**alysis **T**ool

    Let's you examine individual results very nicely (the kind of tool everybody want, but nobody wants to write)

    Automatic error analysis of any existing tool

    When you develop you own entity linker, it is highly recommended that you use this to inspect your results

Entity Linking 6/7

- In-depth analysis of existing linkers and benchmarks
  - Compares the best or most well-known linkers in-depth (on existing benchmarks as well as on two new benchmarks)

    ReFinED, REL, GENRE, Ambiverse, Neural EL, TagMe, and a simple baseline
  - Points out several problems with several widely used existing benchmarks

    AIDA-CoNNL, KORE50, MSNBC, DBpedia Spotlight
  - TODO: what else?

Entity Linking 7/7

- Entity Linking and LLMs
  - With the right prompt, LLMs can also perform entity linking

    Let's try it on ChatGPT (with the GPT-4 model)
  - Observations

    1. It works surprisingly well right out of the box
    2. To obtain great results requires significant fine-tuning
    3. In particular, the model often gets the QIDs wrong
    4. Entity linking using an LLM is **expensive**

    If you want high-quality results with high-performance, tailored solutions are currently still the way to go

References

- Knowledge Graphs
  - https://query.wikidata.org
  - https://github.com/ad-freiburg/qlever
  - https://qlever.cs.uni-freiburg.de
  - https://aqqu.cs.uni-freiburg.de
- Entity Linking
  - https://elevant.cs.uni-freiburg.de
  - https://arxiv.org/abs/2305.14937