TECHNISCHE UNIVERSITÄT DRESDEN

International Center for Computational Logic

KNOWLEDGE GRAPHS

Lecture 1: Introduction

Markus Krötzsch

Knowledge-Based Systems

TU Dresden, 12th Oct 2021

More recent versions of this slide deck might be available.
For the most current version of this course, see
https://iccl.inf.tu-dresden.de/web/Knowledge_Graphs/en

Introduction and Organisation

Course Tutors

Markus Krötzsch
Lectures

Maximilian Marx
Exercises

Organisation

- **Lectures**
  Tuesday, DS 2 (9:20–10:50), Zoom, only after announcement through web page
  Videos will be published online:
  `https://youtube.com/channel/UCCvDWNsR8YlQrB1tSj9Xqsw`
  (see playlist “Knowledge Graphs”)
- **Exercise Sessions (starting 3 Nov 2020)**
  Tuesday, DS 3 (11:10–12:40), BBB (see OPAL for link)
- **Web Page**
  `https://iccl.inf.tu-dresden.de/web/Knowledge_Graphs_(WS2021/22)`
- **Lecture Notes**
  Slides of current and past lectures will be online.
- **Modules**: INF-B-510, INF-B-520, INF-BAS2, INF-VERT2, INF-BAS6, INF-VERT6, INF-E-3, INF-PM-FOR, MCL-KR, MCL-TCSL, CMS-COR-KM

How exercises work

Exercise classes are interactive events:

- Tasks are published online
- Students solve the tasks before the date of the class
- In the meeting, students ask questions and present their solutions
- The tutor answers questions and helps with problems

How exercises work

Exercise classes are interactive events:

- Tasks are published online
- Students solve the tasks before the date of the class
- In the meeting, students ask questions and present their solutions
- The tutor answers questions and helps with problems

Corollary: You must do your homework to benefit from the classes.

How the examination works

The details of the examination depend on your module:

- Most modules: oral examination, sometimes in combination with other subjects
- For students of Computational Modelling and Simulation in module CMS-LM-COR:
  - written (60min) if more than 10 examinees
  - oral (20min) otherwise

All examinations are "closed book" (no auxiliary materials allowed)

However, things might be different than usual due to the COVID pandemic

. . . we will have to wait and see what will be possible

How to be prepared for examinations

Here are some hints that can help you to succeed in the examination:

- Follow the course closely during the semester
- Do all your homework
- When watching videos or attending classes: take hand-written notes
- Be prepared to reproduce most of the material that is on the slides when asked
- Be prepared to solve tasks that are like those in the exercises
- Expect that you will have to write answers on paper (also during oral exams)

How to be prepared for examinations

Here are some hints that can help you to succeed in the examination:

- Follow the course closely during the semester
- Do all your homework
- When watching videos or attending classes: take hand-written notes
- Be prepared to reproduce most of the material that is on the slides when asked
- Be prepared to solve tasks that are like those in the exercises
- Expect that you will have to write answers on paper (also during oral exams)

Main conclusion: You need to be able to produce knowledge and solutions actively rather than merely understanding them passively.

Goals and Prerequisites

Goals

- Introduce basic notions of **graph-based knowledge representation(s)**
- Study important **graph data management approaches** (RDF, Property Graph) and **query languages**
- Learn about relevant **methods, tools, and datasets**
- Discuss aspects of **modelling and quality assurance**
- Get to know some **methods for analysing networks and graphs**

(Non-)Prerequisites

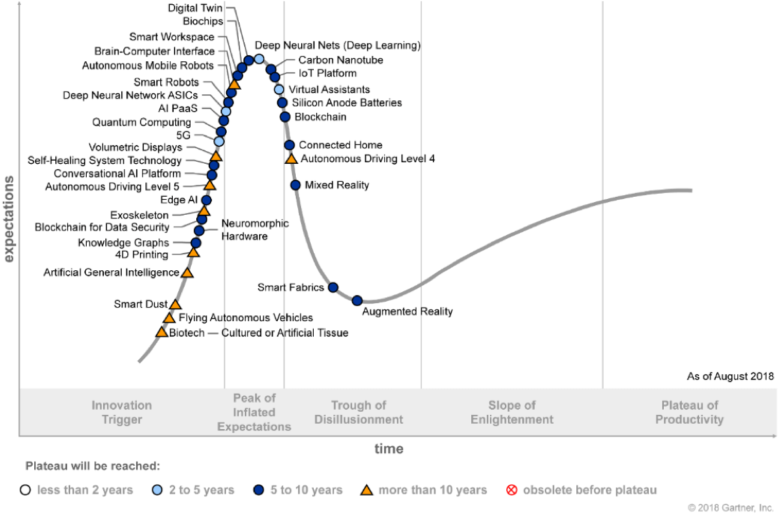
- No particular prior courses needed
- Basic programming skills are assumed; practical experience beyond basic courses will be helpful
- Interesting optional synergies: databases, machine learning, social networks, graph theory

Course Outline

- **Resource Description Framework (RDF)**
  Underlying graph model; URIs; syntax
- **SPARQL**
  Query features; syntax and semantics; expressive power and complexlity
- **Property graph**
  Underlying graph model; syntax and semantics of Cypher
- **Wikidata**
  Data model; applications; aspects of modelling; query answering
- **Ontologies and rules**
  Datalog; negation; existential rules; ontological models; OWL
- **RDF constraint languages**
  SHACL & ShEX; syntax and semantics; complexity and implementation
- **Network analysis**
  Centrality measures, PageRank, community detection

Motivation

The Hype



Digital Twin
Biochips
Smart Workspace
Brain-Computer Interface
Autonomous Mobile Robots
Smart Robots
Deep Neural Network ASICs
AI PaaS
Quantum Computing
5G
Volumetric Displays
Self-Healing System Technology
Conversational AI Platform
Autonomous Driving Level 5
Edge AI
Exoskeleton
Blockchain for Data Security
Knowledge Graphs
4D Printing
Artificial General Intelligence
Smart Dust
Flying Autonomous Vehicles
Biotech — Cultured or Artificial Tissue
Neuromorphic Hardware
Deep Neural Nets (Deep Learning)
Carbon Nanotube
IoT Platform
Virtual Assistants
Silicon Anode Batteries
Blockchain
Connected Home
Autonomous Driving Level 4
Mixed Reality
Smart Fabrics
Augmented Reality
As of August 2018
expectations
time
Innovation Trigger
Peak of Inflated Expectations
Trough of Disillusionment
Slope of Enlightenment
Plateau of Productivity
Plateau will be reached:
less than 2 years
2 to 5 years
5 to 10 years
more than 10 years
obsolete before plateau
© 2018 Gartner, Inc.

Knowledge Graphs Everywhere

Google

Bing

TechRepublic.

BIG DATA

Amazon Neptune is here: 6 ways customers use the AWS graph database

Customers including Samsung, Intuit, and Pearson previewed the database, building new graph applications and testing production workloads.

By Alison DeNisco Rayome | May 31, 2018, 7:44 AM PST

EDITION: EU

ZDNet

CENTRAL EUROPE MIDDLE EAST SCANDINAVIA AFRICA UK ITALY SPAIN MORE

MUST READ: Australian encryption-busting Bill would create backdoors: Cisco

Knowledge graphs beyond the hype: Getting knowledge in and out of graphs and databases

What exactly are knowledge graphs, and what's with all the hype about them? Learning to tell apart hype from reality, defining different types of graphs, and picking the right tools and database for you want to be like the Airbnbs, Amazons, Googles, and LinkedIns of the world.

Forbes

Billionaires Innovation Leadership Money Consumer Industry

Is The Enterprise Knowledge Graph Finally Going To Make All Data Usable?

TC

Startups
Apps
Gadgets
Events
Videos
—
Crunchbase
More

acquire Lattice Data over the weekend. The startup was working to transform the way businesses deal with paragraphs of text and other information that lives outside neatly structured databases. These engineers are uniquely prepared to assist Apple with building a next-generation internal knowledge graph to power Siri and its next generation of intelligent products and services.

Broadly speaking, the Lattice Data deal was an acquihire. Apple paid roughly $10 million for each of Lattice's 20 engineers. This is generally considered to be fair market value. Google paid

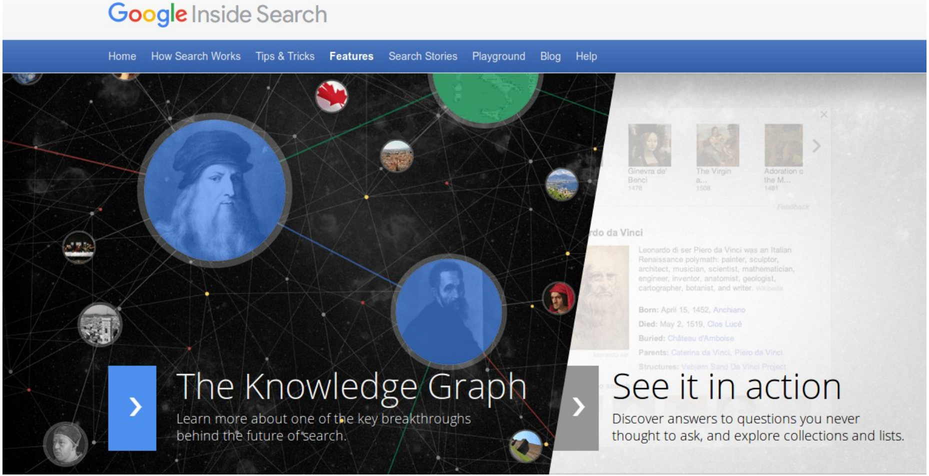
ebay

IBM

What is a Knowledge Graph?

What is a Knowledge Graph?

The original “Knowledge Graph” (Google, 2012):



Many knowledge graphs, many technologies

There are a number of widely used publicly available knowledge graphs:

. . . and a variety of technologies for working with them:

So what is a Knowledge Graph?

A first attempt at a definition:

A Knowledge Graph is a knowledge base that is a graph.

So what is a Knowledge Graph?

A first attempt at a definition:

A Knowledge Graph is a knowledge base that is a graph.

So what is a knowledge base?

So what is a Knowledge Graph?

A first attempt at a definition:

A Knowledge Graph is a knowledge base that is a graph.

So what is a knowledge base?

- "A knowledge base is a technology used to store complex structured and unstructured information used by a computer system. [. . .] [It] represents facts about the world" – Wikipedia (26 Oct 2020, id 983269427)
- "A collection of knowledge expressed using some formal knowledge representation language." – Free Online Dictionary of Computing, 15 Oct 2018
- 1. a store of information or data that is available to draw on.
  2. the underlying set of facts, assumptions, and rules which a computer system has available to solve a problem.

  – Lexico (Oxford University Press/Dictionary.com), 26 Oct 2020

So what is a Knowledge Graph?

A first attempt at a definition:

A Knowledge Graph is a knowledge base that is a graph.

So what is a graph?

So what is a Knowledge Graph?

A first attempt at a definition:

A Knowledge Graph is a knowledge base that is a graph.

So what is a graph?

- "a collection of points and lines connecting some (possibly empty) subset of them" – Wolfram MathWorld, 26 Oct 2020
- "a collection of vertices and edges that join pairs of vertices" – Merriam-Webster, 26 Oct 2020
- "a structure amounting to a set of objects in which some pairs of the objects are in some sense 'related'." – Wikipedia (26 Oct 2020, id 984093316)

(we'll have more to say about mathematical graphs later)

So what is a Knowledge Graph?

A first attempt at a definition:

A Knowledge Graph is a knowledge base that is a graph.

In summary:

So what is a Knowledge Graph?

A first attempt at a definition:

A Knowledge Graph is a knowledge base that is a graph.

In summary:

- a collection of facts, rules, or other forms of knowledge
- that express some kind of relationships or connections

$\rightsquigarrow$ a paradigm rather than a specific class of things

What is special about Knowledge Graphs?

A second attempt at a definition:

A Knowledge Graph is a data set that is:

- structured (in the form of a specific data structure)
- normalised (consisting of small units, such as vertices and edges)
- connected (defined by the – possibly distant – connections between objects)

Moreover, knowledge graphs are typically:

- explicit (created purposefully with an intended meaning)
- declarative (meaningful in itself, independent of a particular implementation or algorithm)
- annotated (enriched with contextual information to record additional details and meta-data)
- non-hierarchical (more than just a tree-structure)
- large (millions rather than hundreds of elements)

(Counter-)Examples

Typical knowledge graphs:

(Counter-)Examples

Typical knowledge graphs:

- Wikidata, Yago 2, Freebase, DBpedia (though hardly annotated)
- OpenStreetMap
- Google Knowledge Graph, Microsoft Bing Satori (presumably; we can’t really know)

(Counter-)Examples

Typical knowledge graphs:

- Wikidata, Yago 2, Freebase, DBpedia (though hardly annotated)
- OpenStreetMap
- Google Knowledge Graph, Microsoft Bing Satori (presumably; we can't really know)

Debatable cases:

(Counter-)Examples

Typical knowledge graphs:

- Wikidata, Yago 2, Freebase, DBpedia (though hardly annotated)
- OpenStreetMap
- Google Knowledge Graph, Microsoft Bing Satori (presumably; we can't really know)

Debatable cases:

- Facebook's social graph: structured, normalised, connected, but not explicit (emerging from user interactions, without intended meaning beyond local relations)

(Counter-)Examples

Typical knowledge graphs:

- Wikidata, Yago 2, Freebase, DBpedia (though hardly annotated)
- OpenStreetMap
- Google Knowledge Graph, Microsoft Bing Satori (presumably; we can't really know)

Debatable cases:

- Facebook's social graph: structured, normalised, connected, but not explicit (emerging from user interactions, without intended meaning beyond local relations)
- WordNet: structured dictionary and thesaurus, but with important unstructured content (descriptions); explicit, declarative model

(Counter-)Examples

Typical knowledge graphs:

- Wikidata, Yago 2, Freebase, DBpedia (though hardly annotated)
- OpenStreetMap
- Google Knowledge Graph, Microsoft Bing Satori (presumably; we can't really know)

Debatable cases:

- Facebook's social graph: structured, normalised, connected, but not explicit (emerging from user interactions, without intended meaning beyond local relations)
- WordNet: structured dictionary and thesaurus, but with important unstructured content (descriptions); explicit, declarative model
- Global data from schema.org: maybe not very connected

(Counter-)Examples

Typical knowledge graphs:

- Wikidata, Yago 2, Freebase, DBpedia (though hardly annotated)
- OpenStreetMap
- Google Knowledge Graph, Microsoft Bing Satori (presumably; we can't really know)

Debatable cases:

- Facebook's social graph: structured, normalised, connected, but not explicit (emerging from user interactions, without intended meaning beyond local relations)
- WordNet: structured dictionary and thesaurus, but with important unstructured content (descriptions); explicit, declarative model
- Global data from schema.org: maybe not very connected
- Document stores (Lucene, MongoDB, etc.): structured, but not normalised; connections sub-ordinary

(Counter-)Examples

Typical knowledge graphs:

- Wikidata, Yago 2, Freebase, DBpedia (though hardly annotated)
- OpenStreetMap
- Google Knowledge Graph, Microsoft Bing Satori (presumably; we can't really know)

Debatable cases:

- Facebook's social graph: structured, normalised, connected, but not explicit (emerging from user interactions, without intended meaning beyond local relations)
- WordNet: structured dictionary and thesaurus, but with important unstructured content (descriptions); explicit, declarative model
- Global data from schema.org: maybe not very connected
- Document stores (Lucene, MongoDB, etc.): structured, but not normalised; connections sub-ordinary

Primarily not knowledge graphs:

(Counter-)Examples

Typical knowledge graphs:

- Wikidata, Yago 2, Freebase, DBpedia (though hardly annotated)
- OpenStreetMap
- Google Knowledge Graph, Microsoft Bing Satori (presumably; we can't really know)

Debatable cases:

- Facebook's social graph: structured, normalised, connected, but not explicit (emerging from user interactions, without intended meaning beyond local relations)
- WordNet: structured dictionary and thesaurus, but with important unstructured content (descriptions); explicit, declarative model
- Global data from schema.org: maybe not very connected
- Document stores (Lucene, MongoDB, etc.): structured, but not normalised; connections sub-ordinary

Primarily not knowledge graphs:

- Wikipedia: mostly unstructured text; not normalised; connections (links) important but sub-ordinary (similar: The Web)

(Counter-)Examples

Typical knowledge graphs:

- Wikidata, Yago 2, Freebase, DBpedia (though hardly annotated)
- OpenStreetMap
- Google Knowledge Graph, Microsoft Bing Satori (presumably; we can't really know)

Debatable cases:

- Facebook's social graph: structured, normalised, connected, but not explicit (emerging from user interactions, without intended meaning beyond local relations)
- WordNet: structured dictionary and thesaurus, but with important unstructured content (descriptions); explicit, declarative model
- Global data from schema.org: maybe not very connected
- Document stores (Lucene, MongoDB, etc.): structured, but not normalised; connections sub-ordinary

Primarily not knowledge graphs:

- Wikipedia: mostly unstructured text; not normalised; connections (links) important but sub-ordinary (similar: The Web)
- Relational database of company X: structured and possibly normalised, but no focus on connections (traditional RDBMS support connectivity queries only poorly)

Graphs in Computer Science and Mathematics

What is a graph?

Definition 1.1: A simple undirected graph G consists of a set V of vertices and a set E of edges, where each edge is a set of two vertices. Two vertices $v_1, v_2 \in V$ are adjacent (in G) if there is an edge $\{v_1, v_2\} \in E$.

Vertices are sometimes also called nodes; undirected edges are sometimes also called arcs.

Unless otherwise noted, we assume all graphs to be finite.

What is a graph?

Definition 1.1: A simple undirected graph G consists of a set V of vertices and a set E of edges, where each edge is a set of two vertices. Two vertices $v_1, v_2 \in V$ are adjacent (in G) if there is an edge $\{v_1, v_2\} \in E$.

Vertices are sometimes also called nodes; undirected edges are sometimes also called arcs.

Unless otherwise noted, we assume all graphs to be finite.

Discrete mathematics considers a variety of other kinds of “graphs”:

- Directed or undirected
- Simple graph or multi-graph
- Possibly labelled edges or vertices
- Possibly with self-loops
- Possibly with higher arity edges (hypergraphs)

Directed and other graphs

Definition 1.2: A simple directed graph (a.k.a. simple digraph) G consists of a set V of vertices and a set $E \subseteq V \times V$ of (directed) edges from a source vertex to a target vertex.

Other terms are similar to undirected graphs; directed edges are also known as arrows and are often denoted as such, e.g., $v_1 \xrightarrow{e_1} v_2$.

Directed and other graphs

Definition 1.2: A simple directed graph (a.k.a. simple digraph) G consists of a set V of vertices and a set $E \subseteq V \times V$ of (directed) edges from a source vertex to a target vertex.

Other terms are similar to undirected graphs; directed edges are also known as arrows and are often denoted as such, e.g., $v_1 \xrightarrow{e_1} v_2$.

Definition 1.3: The following generalisations apply to directed and to undirected graphs.

- A graph with self-loops is a graph extended with the option of having edges that relate a vertex to itself.
- A multi-graph is a graph that may have multiple edges with the same vertices (in the same direction).
- An edge-labelled graph is a graph that has an additional labelling function $\lambda : E \to L$ that maps each edge in E to an element from a set of labels L (similarly for vertex-labelled graphs).

Other basic notions

Definition 1.4: An edge are said to be incidental to the vertices it connects. The degree of a vertex is the number of edges that are incidental to it. In a digraph, the in-degree of a vertex is the number of edges pointing towards it; analogously for out-degree.

Other basic notions

Definition 1.4: An edge are said to be incidental to the vertices it connects. The degree of a vertex is the number of edges that are incidental to it. In a digraph, the in-degree of a vertex is the number of edges pointing towards it; analogously for out-degree.

Definition 1.5: A directed path in a digraph is a sequence of consecutive edges $v_0 \xrightarrow{e_1} v_1 \xrightarrow{e_2} \cdots \xrightarrow{e_n} v_n$. An undirected path is a sequence of edges that may point either way (or that are simply undirected).

A simple path (directed or undirected) is a path without repeated vertices other than possibly the first and last node.

Other basic notions

Definition 1.4: An edge are said to be incidental to the vertices it connects. The degree of a vertex is the number of edges that are incidental to it. In a digraph, the in-degree of a vertex is the number of edges pointing towards it; analogously for out-degree.

Definition 1.5: A directed path in a digraph is a sequence of consecutive edges $v_0 \xrightarrow{e_1} v_1 \xrightarrow{e_2} \cdots \xrightarrow{e_n} v_n$. An undirected path is a sequence of edges that may point either way (or that are simply undirected).

A simple path (directed or undirected) is a path without repeated vertices other than possibly the first and last node.

Definition 1.6: Two vertices are connected if there is an undirected path from one to the other. A graph is connected if any pair of two distinct vertices is connected. A digraph is strongly connected if there is a directed path from any vertex to any other vertex (hence: one directed path in either direction).

Representing graphs (1)

There are several obvious ways of representing graphs in computer science.

Definition 1.7: The adjacency matrix of a graph $G = \langle V, E \rangle$ is the boolean $|V| \times |V|$ matrix that contains, at any coordinate $\langle v_1, v_2 \rangle$, the value 1 if there is an edge connecting v_1 and v_2.

Notes:

- Adjacency matrices for undirected graphs are symmetric.
- Loops (if allowed) show up as 1 in the diagonal.
- The matrix could be adapted to multi-graphs by storing the numbers of edges.
- The matrix could be adapted to labelled simple graphs by storing the labels.

Representing graphs (2)

There are several obvious ways of representing graphs in computer science.

Definition 1.8: The adjacency list of a graph $G = \langle V, E \rangle$ is the list of all of its edges.

Notes:

- We can write edges as pairs (order is irrelevant for undirected graphs).
- Loops (if allowed) show up as edges with repeated vertices.
- The list could be adapted to multi-graphs by adding the number of edges to each line, or by allowing repeated lines.
- The list could be adapted to labelled graphs by adding labels to each line (for multi-graph: repeat lines rather than also storing number).
- The list does not encode V: vertices without edges are missing (might be listed separately if relevant to application)

Representing graphs (3)

There are also other options.

Example 1.9: We could also encode the adjacency matrix by giving, for each row, a list of all vertices whose column is set to 1. This is equivalent to ordering edges by first vertex and combining them into a single line.

Which graph representation to pick?

Each representation has its pros and cons:

- Matrix:
  - \+ space efficient for dense graphs (1 bit per edge);
  - \+ can be processed with matrix operations (highly parallel);
  - – space inefficient for sparse graphs;
  - – not natural for labelled multi-graphs
- List:
  - \+ space efficient for sparse graphs;
  - \+ easy to use for labelled multi-graphs;
  - – harder to process (esp. if edge order can be random);
  - – not space efficient for dense graphs

Note: Knowledge graphs are typically sparse and labelled, but parallel processing still makes matrices attractive in some applications.

Summary

The course will be 50% lectures and 50% tutorial classes

Active participation is important to succeed

Knowledge Graphs are a data management concept of practical importance

Mathematics studies various types of graphs, which can be represented by several common data structures

What's next?

- Encoding graphs in technical systems: the Resource Description Format RDF
- Formats for exchanging RDF
- Modelling knowledge in RDF graphs