TECHNISCHE UNIVERSITÄT DRESDEN

KNOWLEDGE GRAPHS

Lecture 15: Summary and Outlook

Markus Krötzsch
Knowledge-Based Systems

TU Dresden, 4th Feb 2020

Review: Community detection

Community structure can be very helpful to understand graphs:

- Underlying principles of their creation (and future growth)
- Inhomogeneous features
- Optimised processing

"Community" can mean many things, depending on what the graph models:

- For example, using betweenness (Girvan and Newman), modularity, closeness
- Similar to situation for centrality
- More criteria possible (coming up next)

Further approaches to community detection

Further approaches to community detection (1)

Minimum Cuts: cut graph into disconnected components by removing edges

- Prefer cuts that (1) cut only few edges and (2) lead to components of similar size (the approach of assigning a to-be-minimised value to a cut may vary)
- Several efficient algorithms are available
- Good for partitioning network into "local" neighbourhoods by cutting communication bottlenecks (e.g., data placement in distributed computing), less suitable for community detection

Graph partitioning is mostly used in slightly different application areas.

Further approaches to community detection (2)

Bi-Cliques ("Trawling"): discover communities starting from large complete bi-partite subgraphs

- On general graphs, the approach works as follows:
  (1) Split graph into two equal-sized vertex sets
  (2) Look for sub-sets where all vertices on the left are connected to all vertices on the right (mathematical background: in sufficiently dense communities, large bi-cliques must exist)
  (3) Grow communities around the vertices by adding highly linked vertices
- Efficient computation uses techniques from frequent itemset mining to search such "bi-cliques"
- Good for finding also small communities; applicable also to k-partite graphs

Very common approach in social network analysis; can be applied to rather large networks.

Further approaches to community detection (3)

Statistical inference: try to find a model that explains the observed data

- Consider some parametrised graph generator ("the model"), and try to find the parameter values that are most likely to lead to the given graph
- Generally hard to solve and depending on model; many algorithms exists, e.g., search methods for continuous parameter fitting (e.g., gradient descent)
- Capable of finding overlapping communities; might lead to even deeper insights into the nature of the graph than mere community detection

More complex but potentially more insightful and of greater predictive value

Outlook: Ontologies

Models beyond data

An ontology is a description of knowledge about a domain of interest, the core of which is a machine-processable specification with a formally defined meaning.

Ontologies combine several aspects:

1. A vocabulary of terms related to the domain
2. A specification of relationships between vocabulary terms
3. A formal semantics that defines how to evaluate the specified relationships in information systems (especially: which conclusions to draw)
4. A documentation that explains the intended informal meaning of the vocabulary

Related concepts:

- Aspects (1) and (4) constitute a glossary
- Aspects (2) and (3) constitute a (purely abstract) logical theory
- Aspect (3) distinguishes ontologies from other formal modelling languages like UML
- Ontologies often describe relationships of schema-level terms (classes, relations)
  - Ontologies that merely describe a (class) hierarchy are taxonomies
  - Specifications of instance-level relationships correspond to databases

Example: Snomed CT

A practically relevant ontology is Snomed CT (which stands for: Systematized Nomenclature of Medicine, Clinical Terms)

- Goal: provide nomenclature for interoperable health data records
- Vocabulary: some 300K medical terms (symptoms, anatomical structures, diagnoses, treatments, etc.)
- Many essential relationships are modelled formally

  Example 15.1: SNOMED CT may formalise statements such as "If a fracture occurs in a body part A, and A is a part of B, then the fracture also occurs in B." This allows us to conclude, e.g., that every "fracture of the femur" is also a "fracture of the leg".

  Using similar reasoning, an older version of SNOMED supported the conclusion that every amputation of a finger is also an amputation of a hand. Fortunately, SNOMED is only used for reporting medical treatments, not for planning them.

- A formal semantics is used to (pre)compute such consequences before distributing the result to hospitals and other users

Terminology vs. data

Ontologies traditionally speak of general classes and **relations**, and define general truths that must be true for all their instances

Example 15.2: The term "fracture of the femur" describes a class of findings. Its instances are individual fractures. In a hospital database, every patients can be assigned own findings that may be in some or more such classes.

However, the boundary between schema and data is often blurry in practice.

Example 15.3: Wikidata also includes items about many medical conditions, but they are modelled just like other instances (with statements rather than descriptions of schema-level relationships).

One can always encode and manage ontological models as mere (instance) data. The distinction of instance and schema level comes from the formal or pragmatic semantics under which this data is evaluated.

Ontology languages

An ontology language is a formalism used to encode and exchange ontologies, especially the aspects that have a formal semantics

Example 15.4: The Web Ontology Language (OWL) is a W3C standard for describing many basic relationships over vocabularies of (unary) classes and (binary) relations. Its formal semantics is based on description logics, and it mostly compatible with RDF-based knowledge graphs.

Example 15.5: Several rule-based ontology languages have been proposed, e.g., fragments of existential rules. They can describe relationships over vocabularies of relations of any arity (as found in relational databases). There is no widely accepted standard for rule ontologies yet.

See other courses for details (e.g., Description Logics, Semantic Web Technologies, Database Theory)

Reasoning

The formal semantics of ontology languages gives a logical meaning to ontologies, which can be used to analyse and evaluate them
$\rightsquigarrow$ doing so is called reasoning

Basic reasoning tasks:

- Ontology entailment: find out which ontological relationships are logical conclusions that follow deductively from an ontology
- Consistency checking: find out if an ontology is free of (logical) contradictions
- Classification: compute all sub-class relations over an ontology
- Query answering: find deduced query results based on an ontology

Depending on the ontology language, these can be rather simple, but also very hard or even undecidable.

Ontologies vs. constraints

Constraint languages like SHACL and ShEx are also formal languages that allow us to specify relationships, but there are important differences:

Ontologies

- are descriptive (allow us to derive more knowledge)
- are semantic (refer to the intended knowledge model)
- are used to model domain knowledge

Constraints

- are prescriptive (require us to specify knowledge in a certain form)
- are often syntactic (refer to the encoding of the data)
- are used to declare quality requirements

. . . but both

- can be encoded in machine-readable standardised formats
- have a formal semantics
- may require complex evaluation procedures

Summary and Conclusion

Ontologies on data

Ontologies can be used to define virtual views on databases (incl. knowledge graphs)

Approach:

- The ontology might entail additional facts, not explicitly given in data

  Example 15.6: In a medical ontology as before, the fact $\mathrm{FractureOfFemur(frac123)}$ might entail $\mathrm{FractureOfLeg(frac123)}$.

- We can conceptually view the set of all entailed facts as an enlarged database (i.e., a view)
- Queries can be answered (virtually) against this enlarged view

$\rightsquigarrow$ ontology-based query answering and ontology-based data integration use such views

Summary

What we learned:

- How to represent knowledge graphs (in RDF, in Property Graph, in mathematics)
- How to model knowledge in graphs (basic data structures, RDBMS encoding)
- How to query knowledge graphs (in SPARQL, in Cypher)
- How recursive rules can be used for query and data analysis (Datalog, stratified negation)
- Where to find interesting graph data (Wikidata)
- How to ensure knowledge graph quality (general principles, SHACL, ShEx)
- How to analyse network structures (centrality, communities)

. . . and how to do some of this in code.

Summary of Unknowns

What we omitted:

- How to extract knowledge from unstructured sources
  (see courses on information extraction, natural language processing)
- How to build database systems for large graph data
  (see courses on database technologies)
- How to extend graph data with ontological models
  (see course “Semantic Web Technologies”)
- How to learn and predict knowledge graph structures
  (mostly open – significant research but no breakthrough yet)
- How to model knowledge in graphs
  (vaguely defined, but there are some methodologies/paradigms)
- How to apply knowledge graphs in specific scenarios (find your own!)

. . . and many more.

Summary

Ontologies are knowledge models that focus on machine readability and formal semantics

Important ontology language formalisms include description logics and rule languages

Reasoning is used to produce conclusions (including query results)

What’s next?

- Concentrated learning
- Successful examinations
- Free time

Questions about the material or the examination can be asked now.