Towards a format for describing networks

Networks and knowledge graphs

Vladimir Batagelj[1,2,3,4][0000−0002−0240−9446],
Tomaž Pisanski[1,2][0000−0002−1257−5376], Iztok Savnik[1][0000−0002−3994−4805],
Ana Slavec[1][0000−0002−0171−2144], and Nino Bašić[1,2][0000−0002−6555−8668]

[1] UP FAMNIT Koper
[2] IMFM Ljubljana
[3] UL FMF Ljubljana
[4] vladimir.batagelj@fmf.uni-lj.si
[5] https://github.com/bavla/netsJSON

version: March 17, 2025 at 05 : 29

Abstract. The relationship between the concepts of network and knowledge graph is explored.

Keywords: First keyword · Second keyword · Third keyword

1 Introduction

Open data plays a crucial role in ensuring the computational reproducibility and verifiability of published results. Collections of similar datasets are also important for developing methods to analyze specific types of data. When preparing such data, it is essential to adhere to the FAIR principles – Findability, Accessibility, Interoperability, and Reusability. To facilitate ease of use, data should ideally be stored in a text format that preserves the structure of data and includes relevant metadata.

Network analysis is an area where data is often stored in diverse file formats. Adopting a common format for storing network data would be highly beneficial. Such a format would allow us to obtain the specific descriptions required by various network analysis programs using relatively simple scripts.

2 Graphs and networks

2.1 Unit identification

The fundamental task in transforming data about the selected topic into a structured dataset to be used in further analyses is the *identification of units* (entity recognition). Often the source data are available as unstructured or semi-structured text. In this case, the transformation is a task of the *computer-assisted text analysis* (CaTA). *Terms* considered in TA are collected in a *dictionary* (it can be fixed in advance, or built dynamically). The main two problems with terms are

– *equivalence* – different words representing the same term, and
– *ambiguity* – same words representing different terms.

Because of these the *coding* – transformation of raw text data into formal *description* – is done often manually or semiautomaticly.

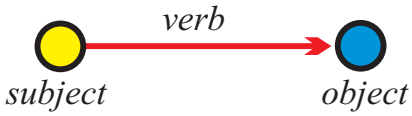
We assume that unit identification assigns a unique identifier (ID) to each unit. For some types of units, such IDs are standardized: ISO 3166-1 alpha-2 two-letter country codes, ISO 9362 Bank Identifier Codes (BIC), ORCID Open Researcher and Contributor ID, ISSN International Standard Serial Number, DOI Digital Object Identifier, URI Uniform Resource Identifier, etc.

In data displays, often IDs are replaced by corresponding (short) labels/names.

Besides the semantic units or *concepts* related to the selected topic we can identify in the raw data also syntactic units – parts of the text. As *syntactic units* of TA we usually consider clauses, statements, paragraphs, news, messages, etc.

In thematic TA the units are coded as rectangular matrix *Syntactic units* × *Concepts* which can be considered as a two-mode network.

In semantic TA the units (often clauses) are encoded according to the S-V-O (*Subject-Verb-Object*) model or its improvements.



This coding can be directly considered as network with *Subjects* ∪ *Objects* as nodes and links (arcs) labeled with *Verbs*.

2.2 Networks

A *network* is based on two sets – set of *nodes* (vertices), that represent the selected *units*, and set of *links* (lines), that represent *ties* between units. They determine a *graph*. A link can be *directed* – an *arc*, or *undirected* – an *edge*.

Additional data about nodes or links can be known – their *properties* (attributes). For example: name/label, type, value, etc.

Network = Graph + Data

The data can be measured or computed.

2.3 Networks Formally

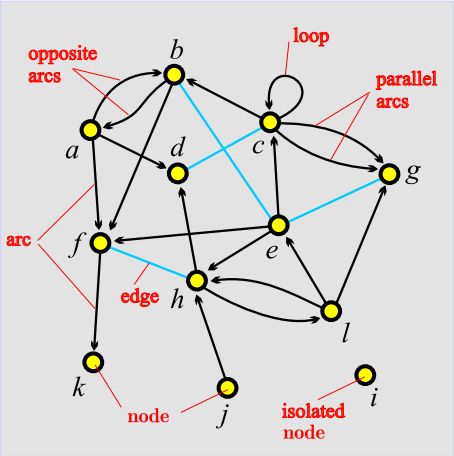
A *network* $\mathcal{N} = (\mathcal{V}, \mathcal{L}, \mathcal{P}, \mathcal{W})$ consists of:

– a *graph* $\mathcal{G} = (\mathcal{V}, \mathcal{L})$, where $\mathcal{V}$ is the set of nodes, $\mathcal{A}$ is the set of arcs, $\mathcal{E}$ is the set of edges, and $\mathcal{L} = \mathcal{E} \cup \mathcal{A}$ is the set of links.
$n = |\mathcal{V}|$, $m = |\mathcal{L}|$
– $\mathcal{P}$ *node value functions* / properties: $p : \mathcal{V} \to A$
– $\mathcal{W}$ *link value functions* / weights: $w : \mathcal{L} \to B$

Additional information/data about values:

– How can we compute with values – algebraic structures semigroup, monoid, group, semiring, etc.
– Properties of values – in a molecular graph an atom is assigned to each node; properties of relevant atoms are such additional data.

2.4 Some terminology



unit, actor – node, vertex
tie, line – link, edge, arc

arc = directed link, (a, d)
a is the *initial* node,
d is the *terminal* node.

edge = undirected link, $(c\colon d)$
c and d are *end* nodes.

3 Types of networks

Besides ordinary (directed, undirected, mixed) networks some extended types of networks are also used:

– *2-mode networks*, bipartite (valued) graphs – networks between two disjoint sets of nodes.
– *multi-relational networks*.
– *linked networks* and *collections of networks*.
– *temporal networks*, dynamic graphs – networks changing over time.
– specialized networks: representation of genealogies as *p-graphs*; *Petri's nets*, etc.

Network (input) file formats should provide a means of expressing all of these types of networks. All interesting data should be recorded (respecting privacy).

3.1 Two-mode networks

In a *two-mode* network $\mathcal{N} = ((\mathcal{U}, \mathcal{V}), \mathcal{L}, \mathcal{P}, \mathcal{W})$ the set of nodes consists of two disjoint sets of nodes $\mathcal{U}$ and $\mathcal{V}$, and all the links from $\mathcal{L}$ have one end node in $\mathcal{U}$ and the other in $\mathcal{V}$. Often also a *weight* $w : \mathcal{L} \to \mathbb{R} \in \mathcal{W}$ is given; if not, we assume $w(u, v) = 1$ for all $(u, v) \in \mathcal{L}$.

A two-mode network can also be described by a rectangular matrix $\mathbf{A} = [a_{uv}]_{\mathcal{U}\times\mathcal{V}}$.

$$a_{uv} = \begin{cases} w(u,v) & (u,v) \in \mathcal{L} \\ 0 & \text{otherwise} \end{cases}$$

Examples: (persons, societies, years of membership), (buyers/consumers, goods, quantity), (parliamentarians, problems, positive vote), (persons, journals, reading), (authors, works, is an author). A classical example of a two-mode network is the Southern women (Davis 1941) Freeman's overview. .

3.2 Multi-relational networks

A *multi-relational network* is denoted by

$$\mathcal{N} = (\mathcal{V}, (\mathcal{L}_1, \mathcal{L}_2, \ldots, \mathcal{L}_k), \mathcal{P}, \mathcal{W})$$

and contains different relations $\mathcal{L}_i$ (sets of links) over the same set of nodes. Also the weights from $\mathcal{W}$ are defined on different relations or their union.

Examples of such networks are the transportation system in a city (stations, lines); WordNet (words, semantic relations: synonymy, antonymy, hyponymy, meronymy, etc.), KEDS networks (states, relations between states: Visit, Ask information, Warn, Expel person, etc,).

3.3 Linked networks and collections of networks

In a *linked* or *multimodal* network

$$\mathcal{N} = ((\mathcal{V}_1, \mathcal{V}_2, \ldots, \mathcal{V}_j), (\mathcal{L}_1, \mathcal{L}_2, \ldots, \mathcal{L}_k), \mathcal{P}, \mathcal{W})$$

the set of nodes $\mathcal{V}$ is partitioned into subsets (*modes*) $\mathcal{V}_i$, $\mathcal{L}_s \subseteq \mathcal{V}_p \times \mathcal{V}_q$, and properties and weights are usually partial functions.

A set of networks $\{\mathcal{N}_1, \mathcal{N}_2, \ldots, \mathcal{N}_k\}$ in which each network shares a set of nodes with some other network is called a *collection* of networks.

A linked network can be transformed into a collection of networks and vice versa.

Bibliographical information is usually represented as a collection of bibliographical networks $\{\mathbf{WA}, \mathbf{Cite}, \mathbf{WK}, \mathbf{WC}, \mathbf{WI}, \ldots\}$ (W – works, A – authors, K – keywords, C – countries, I – institutions).

3.4 Multilevel networks

A *multilevel network* organizes nodes into hierarchical levels, where each level represents a different scale or granularity of interaction or connectivity. It is a special case of a linked network.

Key characteristics of multilevel networks are: hierarchical levels, inter-level links, scale-specific dynamics, and modularity.

Each level may have its unique dynamics, rules, or behaviors. For example, in a biological network, one level might represent protein-protein interactions, while another level represents cellular interactions. They often exhibit modularity, where nodes within a level are more densely linked to each other than to nodes in other levels. This modularity can help in understanding the functional or structural organization of the network.

Example: a multilevel network in the context of a university: Level 1: Individual students and faculty members, Level 2: Departments or academic units, Level 3: The entire university.

3.5 Temporal networks

In a temporal network, the presence/activity of a node/link can change through time $\mathcal{T}$.

Temporal quantities. To describe changes, we introduce a notion of a *temporal quantity* (TQ)

$$a(t) = \begin{cases} a'(t) & t \in T_a \\ \text{⌘} & t \in \mathcal{T} \setminus T_a \end{cases}$$

where T_a is the *activity time set* of a and $a'(t)$ is the value of a in an instant $t \in T_a$, and ⌘ denotes the value *undefined*.

We assume that the values of temporal quantities belong to a set A which is a *semiring* $(A, +, \cdot, 0, 1)$ for binary operations $+ : A \times A \to A$ and $\cdot : A \times A \to A$. $A_{\text{⌘}} = A \cup \{\text{⌘}\}$.

Let $A_{\text{⌘}}(\mathcal{T})$ denote the set of all temporal quantities over $A_{\text{⌘}}$ in time $\mathcal{T}$. To extend the operations to networks and their matrices we first define the *sum* (parallel links) $a + b$ as

$$(a + b)(t) = a(t) + b(t) \quad \text{and} \quad T_{a+b} = T_a \cup T_b.$$

The *product* (sequential links) $a \cdot b$ is defined as

$$(a \cdot b)(t) = a(t) \cdot b(t) \quad \text{and} \quad T_{a \cdot b} = T_a \cap T_b.$$

Let us define TQs $\mathbf{0}$ and $\mathbf{1}$ with requirements $\mathbf{0}(t) = \text{⌘}$ and $\mathbf{1}(t) = 1$ for all $t \in \mathcal{T}$. Again, the structure $(A_{\text{⌘}}(\mathcal{T}), +, \cdot, \mathbf{0}, \mathbf{1})$ is a semiring.

To produce software support for computation with TQs we limit it to TQs that can be described as a sequence of disjoint time intervals with a constant value

$$a = [(s_i, f_i, v_i)]_{i \in 1..k}$$

where s_i is the starting time and f_i the finishing time of the i-th time interval $[s_i, f_i)$, $s_i < f_i$ and $f_i \leq s_{i+1}$, and v_i is the value of a on this interval. Outside the intervals the value of TQ a is undefined, ⌘.

For example $a = [(1, 5, 2), (6, 8, 1), (11, 12, 3), (14, 16, 2), (17, 18, 5), (19, 20, 1)]$. Assuming discrete time, the TQ a has in time points 1, 2, 3, 4 value 2; it is undefined in the time point 5; has in time points 6, 7 value 1; etc.

A *temporal network*

$$\mathcal{N}_T = (\mathcal{V}, \mathcal{L}, \mathcal{P}, \mathcal{W}, T)$$

is obtained if the *time* T is attached to an ordinary network. T is a linearly ordered set of *time points* $t \in T$. In a temporal network some node properties and/or link weights are assigning TQs.

In a temporal network nodes $v \in \mathcal{V}$ and links $l \in \mathcal{L}$ are not necessarily present or active in all time points. If a link $l(u, v)$ is active in time point t then also its end nodes u and v should be active in time t.

We will denote the network consisting of links and nodes active in time $t \in T$ by $\mathcal{N}(t)$ and call it a *time slice* in time point t.

Program `Pajek` also supports descriptions of temporal networks based on *events* – a sequence of transformations (add, remove, change, time point, etc.).

3.6 Multi-relational temporal network – KEDS/WEIS

The types of networks can be combined.

```
890402  YUG     KSV     224  (RIOT)  RIOT-TORN
890404  YUG     ETHALB  212  (ARREST PERSON) ALB ETHNIC JAILED IN YUG
890407  ALB     ETHALB  224  (RIOT)  RIOTS
890408  ETHALB  KSV     123  (INVESTIGATE)   PROBING
...
030731  GER     CYP     042  (ENDORSE)       GAVE SUPPORT
030731  UNWCT   BOSSER  212  (ARREST PERSON) SENTENCED TO PRISON
030731  VAT     EUR     043  (RALLY) RALLIED
030731  UNWCT   BOSSER  013  (RETRACT)       CLEARED
030731  UNWCT   BAL     121  (CRITICIZE)     CHARGES
030731  SER     UNWCT   122  (DENIGRATE)     TESTIFIED
030731  BOSSER  UNWCT   121  (CRITICIZE)     ACCUSED
...
```

Kansas Event Data System *KEDS*

4 Knowledge graphs

4.1 Knowledge graph

A knowledge graph is a structured representation of knowledge that captures entities, relationships, and attributes in a graph-based format. New entities, relationships, and attributes can be added dynamically. Knowledge graphs have evolved from early semantic networks (1960s) and ontologies to become powerful tools for representing and reasoning about complex knowledge. The Austrian linguist Edgar W. Schneider coined the term "knowledge graph" in 1972.

A knowledge graph is a graph of data intended to accumulate and convey knowledge of the real world, whose nodes represent entities of interest and whose edges represent potentially different relations between these entities [9]. It acquires and integrates information into an ontology and applies a reasoner to derive new knowledge [5].

Knowledge graphs are widely used in applications like semantic search, recommendation systems, natural language processing, and artificial intelligence. They are often built using standards like RDF (Resource Description Framework) and queried using languages like SPARQL.

4.2 Knowledge graphs formally

There is no generally accepted definition of a knowledge graph. Most approaches are based on is a set of *facts* $\mathcal{F}$, where each fact is a triple of the form (e_1, r, e_2) or (e, a, v), with: $e, e_1, e_2 \in E$, $r \in R$, $a \in A$, and v (value, which can be a literal or another entity). E is a set of *entities* (nodes), representing real-world objects, concepts, or instances. R is a set of *relationships* (arcs), representing directed links between entities. A is a set of *attributes*, representing properties or characteristics of entities or relationships.

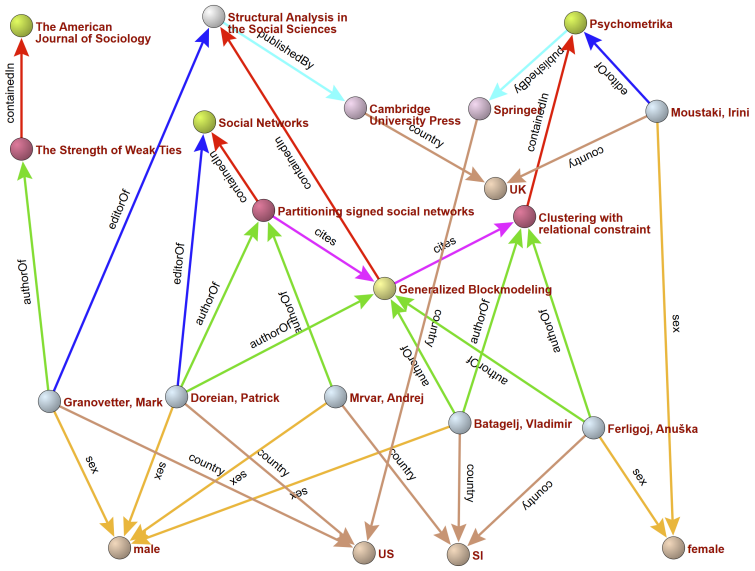


Fig. 1. Knowledge graph

The set of facts $\mathcal{F}$ can be represented with a *knowledge graph* (network)

$$G = (\mathcal{V}, \mathcal{A}, \lambda)$$

such that for every arc $\ell(u, v) \in \mathcal{A}$ the triple $(\lambda(u), \lambda(\ell), \lambda(v)) \in \mathcal{F}$ and there is an arc in $\mathcal{A}$ for each triple from $\mathcal{F}$. There is no isolated node in $\mathcal{V}$. The function $\lambda : \mathcal{V} \cup \mathcal{A} \to \mathcal{L}$ where $\mathcal{L}$ is a set of *labels*, provides semantic meaning to entities, relationships, and attributes.

In a network, the labels on nodes are considered standard node property; arcs with the same label determine the corresponding relation. Therefore the knowledge graph G is a multi-relational network. Usually, the attribute arcs and attribute value nodes are transformed into functions (node properties).

Example. As an example, let us describe a part of the network determined by the following works:

Generalized blockmodeling, Clustering with relational constraint, Partitioning signed social networks, The Strength of Weak Ties

There are nodes of different types (modes): persons, papers, books, series, journals, publishers; different relations between them: author_of, editor_of, contained_in, cites, published_by; and attributes: sex and country. See Figure 1.

4.3 Limitations

An S-V-O triple on which the knowledge graph is based corresponds in logic to a binary predicate $V(S, O)$. In sentences describing facts predicates of larger -arity can appear. For example ***

An approach would be to use hypergraphs instead of graphs.

Another approach is to express k-ary predicate as a binary predicate. Consider a 4-ary predicate $P(x, y, t, z)$. The quadruple (x, y, t, z) can be expressed as a pair in three ways $(x, (y, t, z))$, $(x, y), (t, z))$, and $((x, y, t), z)$. Defining a binary predicate $P'(x, (y, t, z)) = P(x, y, t, z)$ we get the triple $(x, P', (y, t, z))$.

Reification in knowledge representation is the process of turning a predicate or statement into an addressable object. Reification allows the representation of assertions so that they can be referred to or qualified by other assertions WP.

***

Introduction to RDF

RDF 1.1 Resource Description Framework; OWL 2 Web Ontology Language; URI; URN; Turtle; n-ary Relations

RDF 1.2 draft;

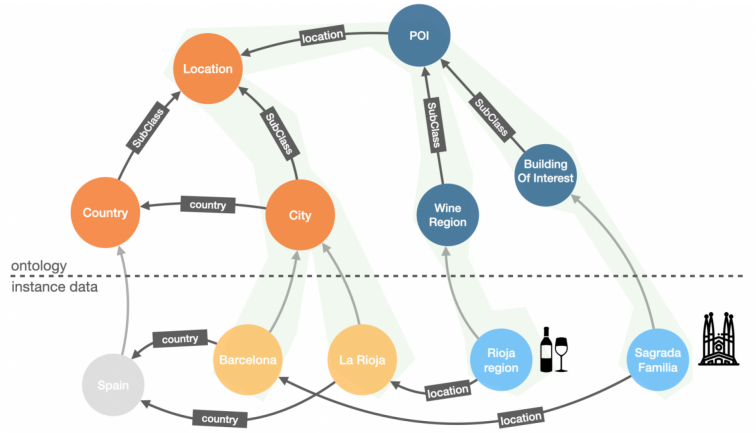


Fig. 2. Barcelona

5 RDF Graphs

In [6] we learn "There can be four kinds of nodes in an RDF graph: IRIs (Internationalized Resource Identifier), literals, blank nodes, and triple terms" (see "1.2 Resources and Statements"). And in "3. RDF Graphs" a definition of an RDF graph is given:

An *RDF graph* is a set of RDF triples. An RDF triple is said to be *asserted* in an RDF graph if it is an element of the RDF graph.

An *RDF triple* (often simply called "triple") is a 3-tuple that is defined inductively as follows:

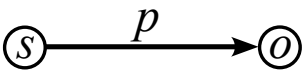
1. If s is an IRI or a blank node, p is an IRI, and o is an IRI, a blank node, or a literal, then (s, p, o) is an RDF triple.
2. If s is an IRI or a blank node, p is an IRI, and o is an RDF triple, then (s, p, o) is an RDF triple.

The three components (s, p, o) of an RDF triple are respectively called the subject, predicate, and object of the triple.

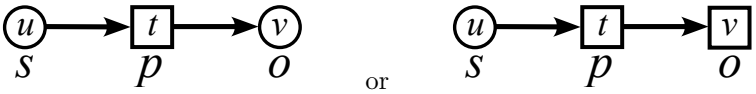
Note: The definition of triple is recursive. That is, a triple can itself have an object component which is another triple. However, by this definition, cycles of triples cannot be created.

IRIs, literals, blank nodes, and triple terms are collectively known as RDF terms.

Instead of traditional RDF graph triple representation with labeled nodes and arcs



we propose a representation based on the following alternative



In the proposed approach an RDF graph is a network $\mathcal{K} = ((\mathcal{S}, \mathcal{T}), \mathcal{A}, \lambda)$. $\mathcal{S}$ is the set of simple nodes, $\mathcal{T}$ is the set of triple nodes, and $\mathcal{V} = \mathcal{S} \cup \mathcal{T}$ is the set of nodes. $\mathcal{A}$ is the set of arcs (directed links) and λ is the label function – for the example triple $\lambda(u) = s$, $\lambda(v) = o$, and $\lambda(t) = p$.

Labels are from three sets $\mathcal{L}$ – literals, $\mathcal{B} = \{\lrcorner\}$ – blank, and $\mathcal{I}$ – IRI. The inductive definition of an RDF graph can be transformed as follows. Let $\mathbb{K}$ be the set of RDF networks. Then

RDFb. Empty network $\mathcal{K} = ((\emptyset, \emptyset), \emptyset, \emptyset) \in \mathbb{K}$.

Let $\mathcal{K} \in \mathbb{K}$ and $\mathcal{K}' = ((\mathcal{S}', \mathcal{T}'), \mathcal{A}', \lambda')$ be the expanded network. Then for every $u \in \mathcal{V}$ we have $\lambda'(u) = \lambda(u)$.

RDFs. If $\lambda'(u) \in \mathcal{I} \cup \mathcal{B}$, $\lambda'(v) \in \mathcal{I} \cup \mathcal{B} \cup \mathcal{L}$, $t \in S$, and $\lambda'(t) \in \mathcal{I}$ then $\mathcal{S}' = \mathcal{S} \cup \{u, v\}$, $\mathcal{T}' = \mathcal{T} \cup \{t\}$, $\mathcal{A}' = \mathcal{A} \cup \{(u, t), (t, v)\}$, and the network $\mathcal{K}' \in \mathbb{K}$.

RDFt. If $v \in \mathcal{T}$, $u \in \mathcal{S}$, $\lambda'(u) \in \mathcal{I} \cup \mathcal{B}$ then $\mathcal{S}' = \mathcal{S} \cup \{u\}$, $\mathcal{T}' = \mathcal{T} \cup \{t\}$, $\mathcal{A}' = \mathcal{A} \cup \{(u, t), (t, v)\}$, and the network $\mathcal{K}' \in \mathbb{K}$.

Let $n_S = |\mathcal{S}|$, $n_T = |\mathcal{T}|$, and $m = |\mathcal{A}|$. For the initial empty network $n_S = n_T = m = 0$. For each application of the rules **RDFs** and **RDFt** we have $n'_T = n_T + 1$ and $m' = m + 2$. Therefore, after k applications of these rules, it holds $n_T = k$ and $m = 2k$.

Questions

Is the transformation done OK?

Can a graph $\mathcal{K} = ((\mathcal{S}, \mathcal{T}), \mathcal{A})$ be recognized as a graph of an RDF network? The oriented K_4 with $n_S = 1$ and $n_T = 3$ is not in $\mathbb{K}$.

Can we obtain the corresponding construction sequence?

6 Conclusions

1.
2.
3.

6.1 Acknowledgments

The computational work reported in this paper was performed using a collection of R functions `bibmat` and the program Pajek for analysis of large networks [**?**]. The code and data are available at Github/Bavla [**?**].

This work is supported in part by the Slovenian Research Agency (research program P1-0294, research program CogniCom (0013103) at the University of Primorska, and research projects J5-2557, J1-2481, and J5-4596), and prepared within the framework of the COST action CA21163 (HiTEc).

References

1. Arroyuelo, D., Hogan, A., Navarro, G., Reutter, J., Vrgoč, D.: Tackling challenges in implementing large-scale graph databases. Communications of the ACM **67**(8), 40–44 (2024)
2. Batagelj, V.: Social network analysis, large-scale (2009)
3. Batagelj, V.: Analysis and visualization of large networks (20/21 October 2005), slides
4. Chen, L., Zhang, H., Chen, Y., Guo, W.: Blank nodes in RDF. J. Softw. **7**(9), 1993–1999 (2012)
5. Ehrlinger, L., Wöß, W.: Towards a definition of knowledge graphs. SEMANTiCS (Posters, Demos, SuCCESS) **48**(1-4), 2 (2016)
6. Hartig, O., Champin, P.A., Kellogg, G., Seaborne, A. (eds.): RDF 1.2 Concepts and Abstract Syntax (25 February 2025), `https://www.w3.org/TR/rdf12-concepts/`, W3C Working Draft
7. Hayes, J., et al.: A graph model for RDF. Darmstadt University of Technology/University of Chile (2004), `https://users.dcc.uchile.cl/~cgutierr/papers/rdfgraphmodel.pdf`

8. Hogan, A., Blomqvist, E., Cochez, M., d'Amato, C., de Melo, G., Gutiérrez, C., Kirrane, S., Labra Gayo, J.E., Navigli, R., Neumaier, S., Ngonga Ngomo, A.C., Polleres, A., Rashid, S.M., Rula, A., Schmelzeisen, L., Sequeda, J.F., Staab, S., Zimmermann, A.: Knowledge Graphs. No. 22 in Synthesis Lectures on Data, Semantics, and Knowledge, Springer (2021). https://doi.org/10.2200/S01125ED1V01Y202109DSK022, https://kgbook.org/
9. Hogan, A., Blomqvist, E., Cochez, M., d'Amato, C., Melo, G.D., Gutierrez, C., Kirrane, S., Gayo, J.E.L., Navigli, R., Neumaier, S., et al.: Knowledge graphs. ACM Computing Surveys (Csur) **54**(4), 1–37 (2021)
10. Hogan, A., Dong, X.L., Vrandečić, D., Weikum, G.: Large language models, knowledge graphs and search engines: A crossroads for answering users' questions. arXiv preprint arXiv:2501.06699 (2025)
11. Nguyen, V., Leeka, J., Bodenreider, O., Sheth, A.: A formal graph model for RDF and its implementation. arXiv preprint arXiv:1606.00480 (2016), https://arxiv.org/abs/1606.00480
12. Paulheim, H.: Knowledge graph refinement: A survey of approaches and evaluation methods. Semantic web **8**(3), 489–508 (2016)
13. Tomaszuk, D., Hyland-Wood, D.: RDF 1.1: Knowledge representation and data integration language for the web. Symmetry **12**(1), 84 (2020)
14. Wilkinson, M.D., Dumontier, M., Aalbersberg, I.J., Appleton, G., Axton, M., Baak, A., Blomberg, N., Boiten, J.W., da Silva Santos, L.B., Bourne, P.E., et al.: The fair guiding principles for scientific data management and stewardship. Scientific data **3**(1), 1–9 (2016)
15. Zhong, L., Wu, J., Li, Q., Peng, H., Wu, X.: A comprehensive survey on automatic knowledge graph construction. ACM Computing Surveys **56**(4), 1–62 (2023)
16. Zloch, M., Acosta, M., Hienert, D., Conrad, S., Dietze, S.: Charaterizing RDF graphs through graph-based measures–framework and assessment. Semantic Web **12**(5), 789–812 (2021)