Chapter 1
What Is a Knowledge Graph?

1.1 Introduction

In recent years, *knowledge graphs* (KGs) have emerged as a major area in Artificial Intelligence (AI) [139]. Graphs have always been pervasive in the broader AI literature, but with the advent of large quantities of data on the Web ('Big Data') and in the broader commercial sphere, there emerged a need to enable machines to 'understand' and make use of this data in some productive analytical way. The inability of machines to truly understand English, and other 'natural' languages like it, with all their irregularities and nuances, has also been largely evident in the (unsuccessful) quest to achieve general AI and commonsense reasoning. Although much progress has been made in all of these domains, it is still very much the case that machines have an easier time processing structured data in the form of graphs, dictionaries and tables than in natural language.

In modern history, Google was among the first big companies to recognize and couple this ability with that of providing richer search capabilities on the Web. In fact, the use of the term 'Knowledge Graph' in recent Computer Science articles, papers and posts, can be traced back to the *Google Knowledge Graph*, which was described in an influential blog post in the early 2010s. The basic motto behind the Google Knowledge Graph was to make search about *things not strings* [164]. In other words, it would allow search to evolve from simple string searching (with all its bells and whistles), to one that involved reasoning about entities, attributes and relationships. The effort can be argued to have been very successful. While the full size and scope of the Google Knowledge Graph is not known, it has grown considerably in size and many search results on Google now involve *knowledge panels* (Fig. 1.1), which are elaborate, yet condensed, information sets about entities that the user might have been searching for. This is in contrast to the previous status quo, which was a list of webpages, ordered by predicted relevance to the user's search query. Beyond Google, other companies have also now started investing in knowledge graphs, and a number of KG-centric startups have emerged in multiple

M. Kejriwal, *Domain-Specific Knowledge Graph Construction*, SpringerBriefs in Computer Science, https://doi.org/10.1007/978-3-030-12375-8_1

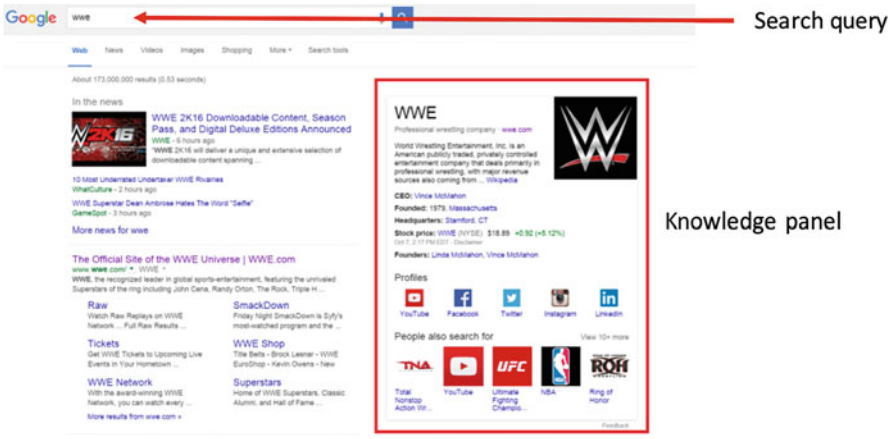

Fig. 1.1 An illustration of a knowledge panel rendered in Google for the search query 'wwe'. At least in part, the panel is powered by KG-centric technologies

countries and continents. There are also applications in non-profit, government and academia. We cover an exciting range of current and growing KG ecosystems in Chap. 5.

Defined *abstractly*, a knowledge graph is a graph-theoretic representation of human knowledge such that it can be ingested *with semantics* by a machine. In other words, it is a way to express 'knowledge' using graphs, in a way that a machine would be able to conduct *reasoning and inference* over this graph to answer queries ('questions') in some meaningful way. However, this definition is not very operational. The simplest *functional definition* of a knowledge graph is that it is a set of *triples*, with each triple intuitively representing an 'assertion'. If the KG was constructed correctly (with 100% accuracy) over a trustworthy data source, we could also think of assertions as *facts*. Formally, a triple is a 3-tuple (h, r, t) where h represents a head entity, t represents a tail entity, and r expresses a relationship between the two entities. Many, though not all, statements in natural language (e.g., English) can be expressed conveniently in this form. Consider, for example, the sentence *Fido the dog stole a bone from Mary's backyard*, which can be expressed as a set of triples[1] {(Fido, is-a, Dog), (Fido, stole, bone_1), (bone_1, is-a, Bone), (bone_1, located-in, yard_1), (yard_1, is-a, Yard), (yard_1, belongs-to, Mary), (Mary, is-a, Person)}.

Why does it make sense to call such a set a 'graph'? For a long time, in fact, it was not conventional to do so and what we are referring to as a knowledge graph here used to be known (and is still known, in many papers) as a knowledge *base*. One of

[1]For reasons that will become clear throughout the book, we use identifiers such as bone_1 and yard_1 to refer to *instances* of *concepts* (also called *classes*) such as Bone and Yard. The convention adopted herein is to use capitalized initials for concepts.

the main reasons why knowledge bases slowly morphed into knowledge graphs can be attributed to the influence and success of the Google Knowledge Graph. However, there was also pervasive influence from both the knowledge discovery, and Semantic Web, communities, both of which have always been closely associated with graph-theoretic innovations. For a large part of this millennium, the database community was also studying graph databases, algorithms and data structures in detail.

This fascination (both industrial and academic) with graphs aside, there was another good reason to think of knowledge bases as graphs. First, if one takes the step of visualizing the first and third elements (i.e. h and t) of a triple as *nodes*, and the second element r as a labeled, directed edge pointing from h (the *head* entity) to t (either a *tail* entity or an *attribute*), an intuitive data model emerges (Fig. 1.2). In fact, many people would find it easier to draw the kind of diagram shown in Fig. 1.2 (with a few examples for guidance) than thinking carefully about sets of triples. In a certain sense, the KG can be said to serve as a lingua franca between machines and humans, in that it is structured enough for machines to process and ingest with semantics, but is intuitive enough for humans to make sense of, at least if represented and drawn using common-sense mnemonics. In fact, the Freebase knowledge graph, and more recently, Wikidata, allow the crowdsourced acquisition of such structured knowledge, as opposed to Wikipedia, where the crowdsourced knowledge is acquired mostly in natural language.

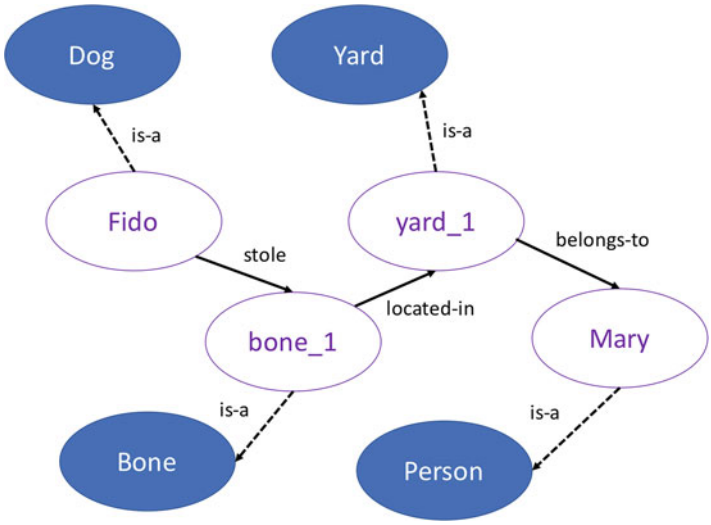

Fig. 1.2 The Knowledge Graph (KG) representation of the information expressed in the Fido the dog example. Filled ovals (i.e. concepts) are parts of the ontology, while the unfilled ovals are part of the KG itself (the instances) is-a relationships (dashed edges) mediate between instances (in the KG) and concepts (in the ontology). Other relationships are defined in the ontology, but used in the KG. Constraints on how the relationships may be used are considered part of the ontology, typically defined using formal declarations

Although the simple definition (which we shall refer to as the 'knowledge base' or KB-definition, where relevant) has many advantages, not the least of which is its simplicity and ease of reading into, and serializing from, machine learning and other data analytics programs, it is also unsatisfactory for certain applications. Just like we do not want a database to have an 'open' schema, we do not always want a knowledge graph to be unconstrained in terms of the data it contains, and the ways in which that data is modeled. This leads to the notion of an *ontology*, which (put simply) defines (and imposes constraints) on the concepts and relationships that are permissible in a KG. For example, considering the earlier example of Fido the dog, it is clear that the ontology contains concepts such as *Dog*, *Bone*, *Yard* and *Person* and defines relationships as well. An example of a defined constraint is that the 'belongs-to' relation must have a *Person* instance (e.g., Mary) as its *target*. Considering the example in Fig. 1.2, the is-a relationship mediates between the KG, which contains instances, and the ontology, which contains concepts. Although the example makes it look straightforward, it can sometimes become a point of contention as to what is a concept and what is an instance in the real world.

Beyond the Google KG, most KGs are domain-specific and have some kind of underlying ontology. This is because there is typically no 'one-size fits all' schema or ontology that is well-suited for solving all problems or answering all queries. Deciding what makes for a good ontology is a controversial topic that is outside the scope of this book. However, once an ontology is given, the expectation is that the KG will conform to the ontology. The more complex the ontology, the harder it is to make the KG conform, but the stronger are its semantics and the complexity of the queries that can be posed on the knowledge graph. As the community has moved towards statistical, data-rich methods, ontologies (designed for knowledge graphs) have generally become less complex over time since it makes the publishing and checking of data easier. Knowledge graphs that contain encyclopedic information have also fueled this trend, since it is not clear if it is even possible to design deep, sophisticated ontologies for 'broad' domains.

In the next few sections, we detail some concrete examples of knowledge graphs in various domains. These examples were selected to express both how intuitive, and expressive, a knowledge graph can be in representing diverse information sets across multiple domains. The examples also illustrate the importance of the domain in modeling and representing knowledge graphs. Some domains, like the event domain, require lots of classes and properties in their underlying ontologies while others can be modeled with only a few classes and properties. Often, there is a choice in how expressive to make the ontology.

1.2 Example 1: Academic Domain

As our first example of a domain-specific KG, let us consider the academic *publication* domain (Fig. 1.3). The two purple nodes in the center of the KG represent different publications, named mnemonically by their publication titles.

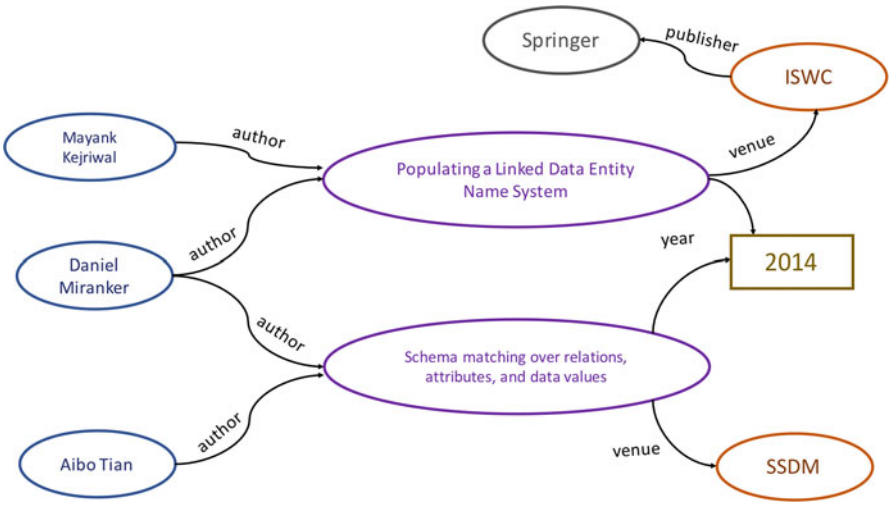

Fig. 1.3 An illustration of a *Publication* knowledge graph, showing two different publications sharing authors. Rectangles are typically used diagrammatically to represent *literals* while oval nodes represent *resources* or *entities*

Some important details concerning the publications are also shown, including their authors, dates of publication and venues.

Despite its simplicity, the KG in Fig. 1.3 illustrates some of the expressiveness in *representation*, an issue that becomes extremely important in communities such as the Semantic Web (SW). The oval nodes in the figure represent *entities* or *resources*, and are generally referred to (in the SW community) as Internationalized Resource Identifiers (IRIs), a generalized form of *Uniform* Resource Identifiers (URIs). In this book, we do not define these concepts, since they are community-specific, but focus more on the overall distinction between entities and literals (also known as *attributes*). Entities can have relationships with other entities (such as between authors and their publications) or attributes (such as the year of a publication). The distinction can be expressed by the fact that in a triple (h, r, t), t is either a literal (for the latter) or an entity (for the former). Note that h is always an entity.

1.3 Example 2: Products and Companies

In the second example, inspired by the *products and e-commerce* domain, we expand upon the notions presented in academic domain. Once again, we see the distinction between literals and entities, but as illustrated in Fig. 1.4, there are numerous degrees of freedom even when modeling the most basic structures in KGs. In this case, we see the same product, represented and modeled in two different ways. The choice of modeling can have implications both for upstream tasks (such as information

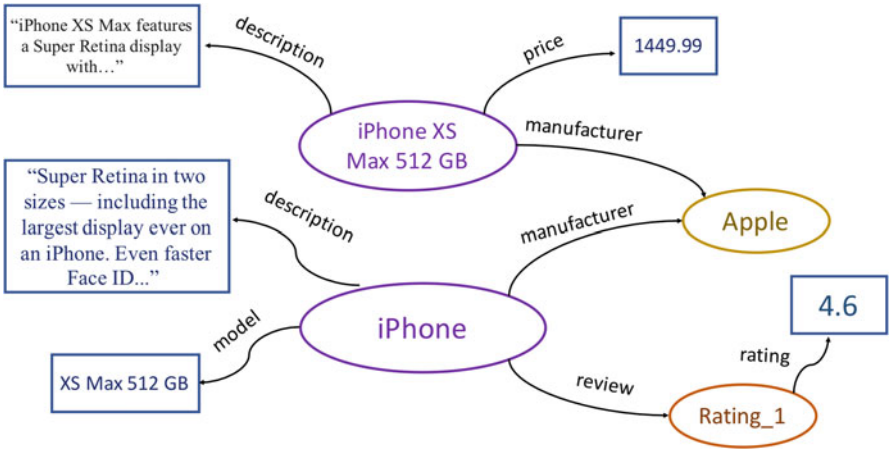

Fig. 1.4 An illustration of a *Product* knowledge graph, showing the same product but represented in different ways. The problem of linking the same underlying entity nodes (Entity Resolution) will be covered in detail in Chap. 3

extraction) and downstream tasks, such as entity resolution and querying, that occur after the initial KG has been extracted and stored. We also see that the availability of information can vary, usually depending on the source from which the KG nodes were extracted to begin with. Also, because the two *product mentions* have not been resolved into a single underlying entity, it is not straightforward to compute an *aggregation* (e.g., the number of unique products) over such KGs and expect reasonable or correct answers. Because it is often the case that the same entity is extracted independently from multiple raw sources, one has to perform *Entity Resolution* on the extracted KG. We cover this step in more detail in Chap. 3.

1.4 Example 3: Geopolitical Events

Finally, we consider the most complex, and cutting-edge, example of a geopolitical event KG. In addition to the usual artifices that we saw before (entities vs. literals etc.), the graph illustrates how 'second-order' entities like events can be represented in a KG. Events are second-order because they have first-order entities like locations and times as their *arguments*; in turn, these first-order entities have attributes describing them further (Fig. 1.5). Events can also directly have attributes, and similar to first-order entities, have relationships between themselves. The notion of what separates first-order from second-order entities is not completely well-understood and is more of a semantic rather than a syntactic issue. In practice, the difference is very real. Extracting and resolving events, for example, have become areas of research in their own right, and performance on them continues to be poor

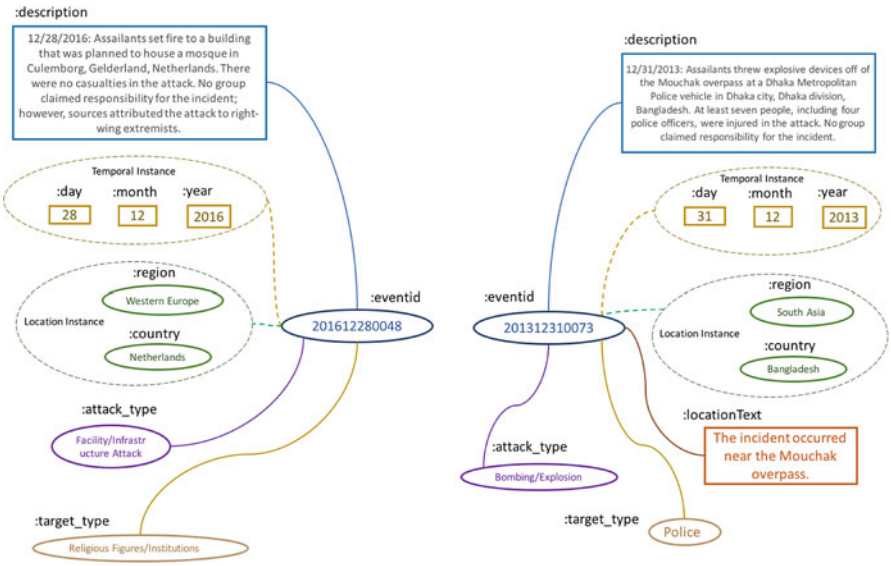

Fig. 1.5 An illustration of an *Event* knowledge graph, showing two disparate geopolitical events

in comparison to performance on extracting and resolving first-order entities like persons and locations. A good example of an event knowledge graph is GDELT [101].

1.5 Conclusion

Knowledge graphs have become a popular data representation that sits at the intersection of knowledge discovery, data mining, Semantic Web and Natural Language Processing. Each of these communities has had dealings with knowledge graphs and their applications. In fact the term is so broad that there is no real 'survey' of knowledge graphs, making it difficult to attribute the invention to a specific paper or author. Generally, the focus is on 'generic' knowledge graphs without too much emphasis on the domain, and domain-specific constraints, that girds the construction and representation of the knowledge graph. An increasing amount of evidence suggests that there is no one size fits all model for knowledge graph construction and inference that can be used across all domains; rather, special domain-specific techniques must be used to obtain state-of-the-art performance. In the rest of this book, we cover domain-specific knowledge graph construction in detail. Although the area is continuing to evolve, some trends have been established and are built on prior research developed over multiple decades. At the time of writing, knowledge graph-powered applications continue to proliferate (Chap. 5).