

Subscribe to DeepL Pro to edit this document.  
Visit [www.DeepL.com/pro](https://www.deepl.com/pro?cta=edit-document) for more information.

**Introduction knowledge graph :**

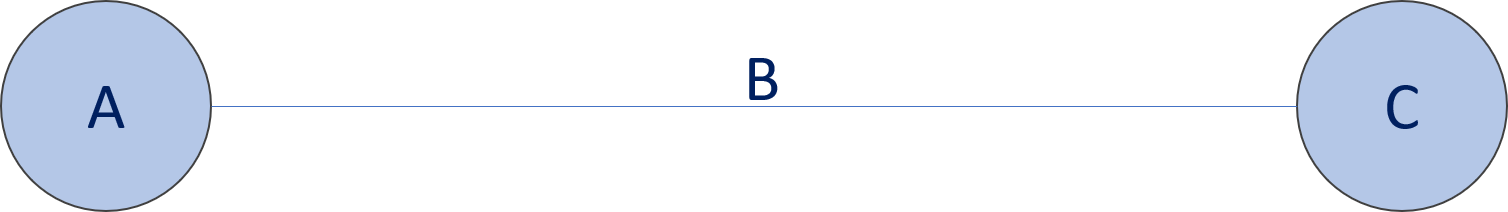
For the past few years, knowledge graphs have been all around us... Whether it's through personalized shopping experiences via online recommendations on websites such as Amazon, [Zalando](https://zeenea.com/fr/comment-le-big-data-a-contribue-au-succes-de-zalando/), or through our favorite search engine [Google](https://zeenea.com/fr/google-goods-outil-de-gestion-des-donnees-de-google/).

However, this concept is still often a challenge for most data and analytics managers who are trying to aggregate and link their enterprise assets to leverage them like those web giants.

**What is a knowledge graph?**

A knowledge graph, also known as a semantic network, represents a network of real-world entities, i.e. objects, events, situations or concepts, and illustrates the relationship between them. This information is usually stored in a graph database and visualized in the form of a graph structure, prompting the use of the term knowledge "graph".

A knowledge graph consists of three main components: nodes, edges and labels. Any object, place or person cannot be a node. An edge defines the relationship between nodes. For example, a node can be a client, such as IBM, and an agency, such as Ogilvy. An advantage would be to categorize the relationship as a client relationship between IBM and Ogilvy.



A is the subject, B is the predicate, C is the object

According to GitHub, a knowledge graph is a type of ontology that describes knowledge in terms of entities and their relationships in a dynamic and automated way. Unlike static ontologies, which are very difficult to maintain.

Here are other definitions of a knowledge graph proposed by different experts:

A "way of storing and using one's data that allows people and machines to better exploit the connections in their datasets." (Datanami)

* A "database that stores information in a graphical format - and, importantly, can be used to generate a graphical representation of the relationships between any of its data points." (Forbes)

"Encyclopedias of the Semantic World". (Forbes)

Thanks to machine learning algorithms, a knowledge graph provides a structure for all your data and allows the creation of multilateral relationships in all your data sources.

The fluidity of this structure increases as new data is introduced, creating more relationships and adding more context, and helping your data teams make informed decisions with connections you might never have found.

The idea of a knowledge graph is to build a network of objects and, more importantly, to create semantic or functional relationships between the different assets.

In a data catalog, a knowledge graph is therefore what represents different concepts and links objects together by semantic or static links.

**Main features**

Knowledge graphs combine features of several data management paradigms:

**Database**, as the data can be explored via structured queries;

**Graph**, as they can be analyzed like any other network data structure;

**Knowledge base**, as they carry formal semantics, which can be used to interpret data and deduce new facts.

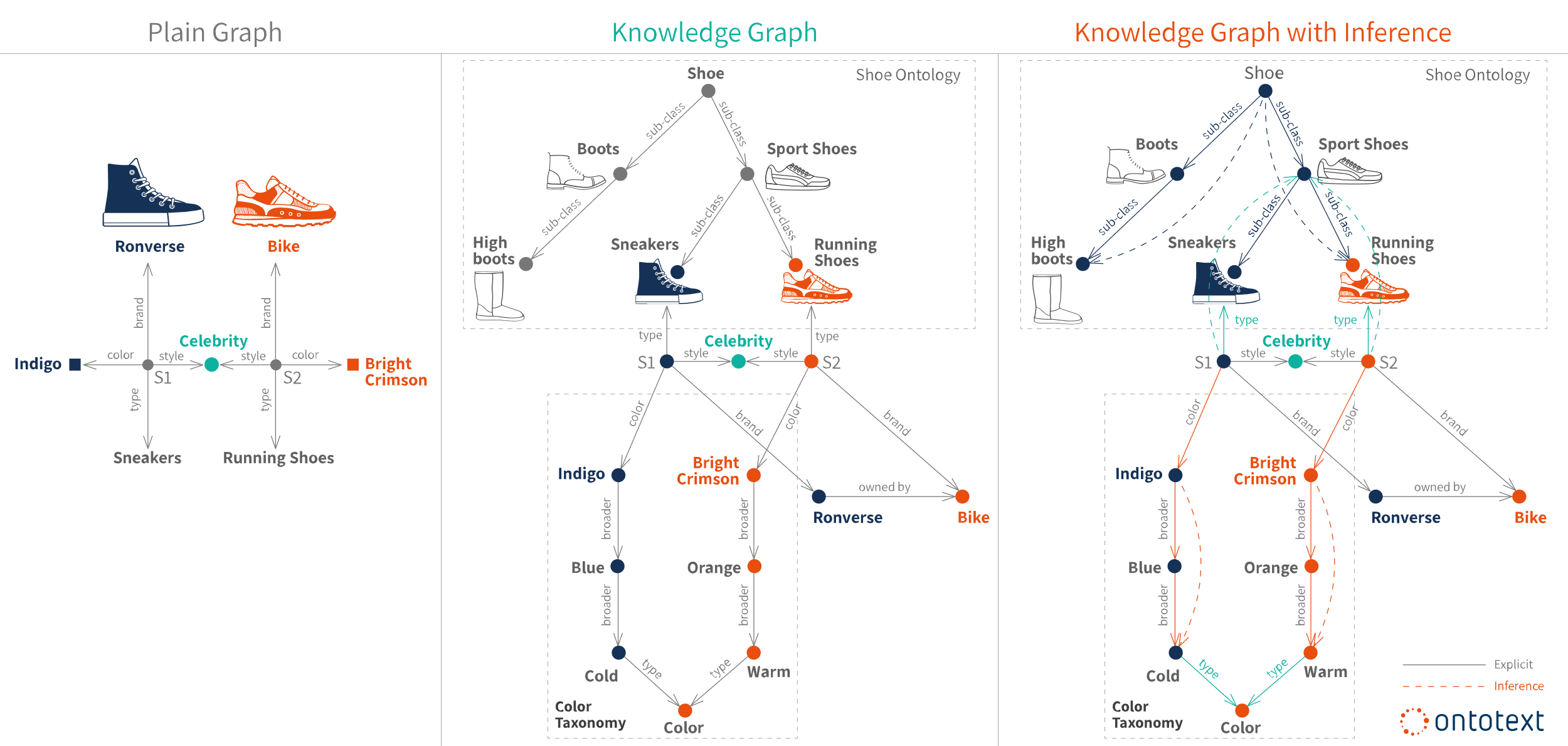
Knowledge graphs, represented in [RDF](https://www.ontotext.com/knowledgehub/fundamentals/what-is-rdf/) , provide the best framework for data integration, unification, linking and reuse, as they combine :

**Expressiveness**: The Semantic Web stack standards - RDF(S) and OWL - allow for the seamless representation of various types of data and content: data schemas, taxonomies and vocabularies, all kinds of [metadata](https://www.ontotext.com/knowledgehub/fundamentals/metadata-fundamental/), reference and reference [data.](https://www.ontotext.com/knowledgehub/fundamentals/metadata-fundamental/) The [RDF extension\*](http://graphdb.ontotext.com/documentation/9.2/free/devhub/rdf-sparql-star.html) facilitates the modeling of provenance and other structured metadata.

**Performance**: All specifications have been designed and tested in practice to allow efficient management of graphs of billions of facts and properties.

**Interoperability**: There is a range of specifications for data serialization, access (SPARQL Protocol for end-points), management (SPARQL Graph Store) and federation. The use of globally unique identifiers facilitates data integration and publication.

**Standardization**: All of the above is standardized through the W3C community process, to ensure that the requirements of the various stakeholders are met, from software developers to enterprise data management professionals to system operations teams.

****

**Ontologies and formal semantics**

[Ontologies](https://www.ontotext.com/knowledgehub/fundamentals/what-are-ontologies/) represent the backbone of the formal semantics of a knowledge graph. They can be seen as the data schema of the graph**.** They serve as a formal contract between the developers of the knowledge graph and its users regarding the meaning of the data it contains. A user can be another human being or a software application that wishes to interpret the data reliably and accurately. Ontologies provide a shared understanding of the data and its meanings.

When formal semantics is used to express and interpret the data in a knowledge graph, there are a number of representation and modeling tools available:

**Classes.** Most often, an entity description contains a classification of the entity with respect to a hierarchy of classes. For example, when dealing with business information, there may be *Person*, *Organizations*, and *Location* classes. Persons and Organizations may have a common superclass *agent.* Location usually has many subclasses, for example *Country*, *Populated place*, *City*, etc. The notion of class is borrowed from object-oriented design, where each entity usually belongs to exactly one class.

**Relationship types**. Relationships between entities are usually labeled with types, which provide information about the nature of the relationship, e.g., *friend*, *relative*, *competitor*, etc. Relationship types can also have formal definitions, for example, that *parent-of is* the inverse relationship of *child-of*, they are both special cases of *relative-of*, which is a symmetric relationship. Or define that *subregion* and *subsidiary* are transitive relationships.

**Categories**. An entity can be associated with categories, which describe certain aspects of its semantics, for example *"the four great consultants"* or *"19th century composers"*. A book can belong to all these categories simultaneously: *"Books on Africa"*, *"Bestseller"*, *"Books by Italian authors"*, *"Children's books"*, etc. The categories are described and arranged in taxonomy.

**Free text descriptions**. Often, a "friendly text" description is provided to further clarify the design intent of the entity and enhance the search.

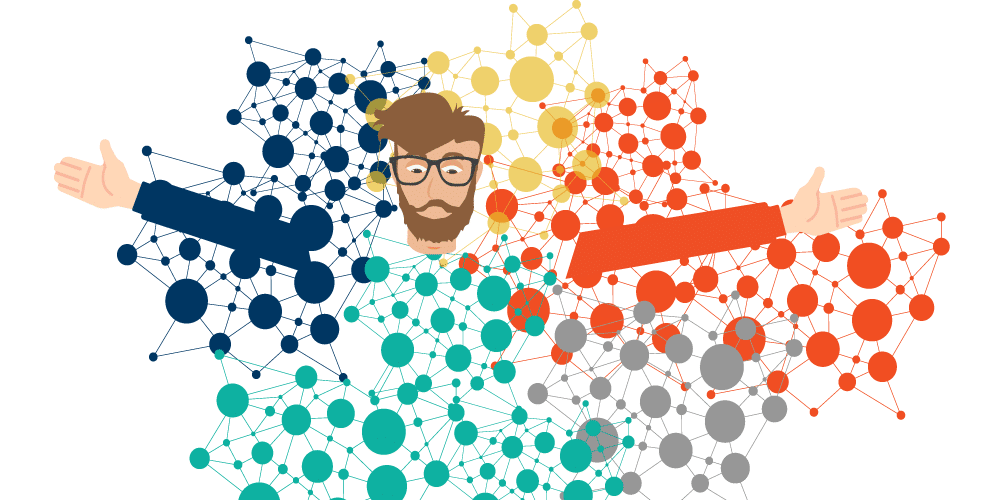
**Example of large knowledge graphs**

**Google Knowledge Graph.**

Google's algorithm uses this system to collect and provide end users with information relevant to their queries. Google's knowledge graph contains more than 500 million objects, as well as more than 3.5 billion facts about these different objects and the relationships between them.

The knowledge graph improves Google search in three main ways:

* **Finding the right result**: a search not only based on keywords but also on their meaning.
* **Get the best summary**: gather the most relevant information from various sources based on the user's intent.
* **Deepen and broaden your search**: discover more than you expected with relevant suggestions.



**How a knowledge graph works**

Knowledge graphs are typically made up of data sets from various sources, often with different structures. Schemas, identities and context work together to provide structure to various data. Schemas provide the framework for the knowledge graph, identities classify the underlying nodes appropriately, and context determines the framework in which that knowledge exists. These components allow words with multiple meanings to be distinguished. This allows products, such as Google's search engine algorithm, to determine the difference between Apple, the brand, and Apple, the fruit.

Knowledge graphs, powered by machine learning, use natural language processing (NLP) to build a complete view of nodes, edges, and labels via a process called semantic enrichment. When data is ingested, this process allows knowledge graphs to identify individual objects and understand the relationships between different objects. This actionable knowledge is then compared and integrated with other datasets that are relevant and similar in nature. Once a knowledge graph is complete, it allows question answering and search systems to retrieve and reuse complete answers to given queries. While consumer products demonstrate their time-saving capabilities, the same systems can also be applied in a business environment, eliminating manual data collection and integration work to support business decision making.

Data integration efforts around knowledge graphs can also support the creation of new knowledge, making connections between data points that may not have been realized before.

**Use cases for knowledge graphs**

There are a number of popular consumer knowledge graphs that define user expectations for search systems across all businesses. Some of these knowledge graphs include:

* DBPedia and Wikidata are two different knowledge graphs for data on Wikipedia.org. DBPedia is composed of data from Wikipedia infoboxes while Wikidata focuses on secondary and tertiary objects. Both generally publish in RDF format.
* Google Knowledge Graph is represented by Google's search engine results pages (SERPs), providing information based on what people are searching for. This knowledge graph is made up of over 500 million objects, drawing on data from Freebase, Wikipedia, the CIA World Facebook, and more.

However, knowledge graphs also have applications in other industries, such as :

* **Retail:** Knowledge graphs were used for up-selling and cross-selling strategies, recommending products based on individual buying behaviour and popular buying patterns across all demographic groups.
* **Entertainment:** knowledge graphs are also used for artificial intelligence (AI) based recommendation engines for content platforms, such as Netflix, SEO or social media. Based on clicks and other online engagement behavior, these providers recommend users to read or watch new content.
* **Finance:** This technology has also been used for Know Your Customer (KYC) and anti-money laundering initiatives within the financial sector. They assist in the prevention and investigation of financial crime, enabling banking institutions to understand their customers' money flows and identify non-compliant customers.
* **Health:** Knowledge graphs also benefit the health sector by organizing and categorizing relationships within medical research. This information helps providers by validating diagnoses and identifying treatment plans based on individual needs.

**Knowledge graphs and RDF databases**

Years ago, we went from the buzzword Big Data to Smart Data. Having unprecedented amounts of data has driven the need to have a data model that reflects our own complex understanding of information.

To make data intelligent, machines no longer had to be bound by inflexible data schemas defined "a priori". We needed data repositories that could represent the "real world" and the entangled relationships that flow from it. All of this had to be done in a machine-readable way and have formal semantics to allow for automated reasoning that complemented and facilitated our own.

RDF databases (also called RDF triplestores), such as Ontotext's GraphDB, can seamlessly integrate heterogeneous data from multiple sources and store hundreds of billions of facts about any concept imaginable. The RDF graph structure is very robust (it can handle massive amounts of data of all kinds and from various sources) and flexible (it does not need to redefine its schema every time we add new data).

Another important feature of RDF databases is their inference capability where new knowledge can be created from already existing facts. When such new facts are materialized and stored in an RDF database, our search results become much more relevant, opening new avenues for actionable information.

**Recent applications of Knowledge Graphs**

There are many applications of knowledge graphs in both research and industry. In computer science, there are many uses of a directed graph, for example, data flow graphs, binary decision diagrams, state charts, etc. For our discussion here, we have chosen to focus on two concrete applications that have led to a recent surge in popularity of knowledge graphs: organizing information on the Internet and data integration.

**Knowledge graphs for organizing knowledge on the Internet**

We will explain the use of a knowledge graph on the web by taking the concrete example of Wikidata. Wikidata acts as the central storage of structured data for Wikipedia. To show the interaction between the two and to motivate the use of the Wikidata knowledge graph, let us consider the city of Winterthur in Switzerland which has a page in Wikipedia. Winterthur's Wikipedia page lists its twin cities: two in Switzerland, one in the Czech Republic and one in Austria. The city of Ontario, California, which has a Wikipedia page entitled *Ontario, California*, lists Winterthur as a sister city. The relationship between sister cities and twin cities is identical and reciprocal. Thus, if a city A is a sister city to another city B, then B must be a sister city to A. This inference should be automatic, but because this information is listed in English in Wikipedia, it is not easy to detect this discrepancy. . On the other hand, in the Wikidata representation of Winterthur, there is a relationship called *sister administrative body* that lists the city of Ontario. Since this relationship is symmetric, the Wikidata page for the city of Ontario automatically includes Winterthur. Thus, when the Wikidata knowledge graph is fully integrated with Wikipedia, such discrepancies will naturally disappear.

Wikidata includes data from several independent providers, for example, the Library of Congress which publishes data containing information about Winterthur. Using the Wikidata identifier for Winterthur, information published by the Library of Congress can be easily linked to information available from other sources. Wikidata facilitates such linking by publishing the definitions of the relationships used in *Schema.Org*.

The relationship vocabulary in *Schema.Org* gives us at least three advantages. First, it is possible to write queries spanning multiple datasets, which would not otherwise be possible. An example of such a query is: Display on a map the cities of birth of people who died in Winterthur? Second, with such a query capability, it is possible to easily generate structured information boxes within Wikipedia. Third, the structured information returned by queries can also appear in search results, which is now a standard feature for major search engines.

A recent version of Wikidata contained over 80 million objects, with over a billion relationships between these objects. Wikidata makes connections across more than 4872 different catalogues in 414 different languages published by independent data providers. According to the recent estimate, 31% of websites and more than 12 million data providers publishing Schema.Org annotations currently use the *Schema.Org* vocabulary.

Let's look at several key features of the Wikidata knowledge graph. First, it is a graph of unprecedented scale, and is the largest knowledge graph available today. Second, it is jointly created by a community of contributors. Third, some of the data in Wikidata may come from automatically extracted information, but it must be easily understood and verified according to Wikidata's editorial policies. Fourth, there is an explicit effort to provide semantic definitions of various relation names through the *Schema.Org* vocabulary. Finally, the main use case for Wikidata is to improve web search. While Wikidata has several applications that use it for analysis and visualization tasks, its use on the Web continues to be the most compelling and easy to understand application.

**Knowledge graphs for enterprise data integration**

Data integration is the process of combining data from different sources and providing the user with a unified view of the data. Much of the data in organizations resides in relational databases. One approach to data integration is based on a global schema that captures the interrelationships between the data elements represented in these databases. Creating a global schema is an extremely difficult process because there are many tables and attributes; the experts who created these databases are usually not available; and due to the lack of documentation, it is difficult to understand the meaning of the data. Because of the challenges of creating a global schema, it is practical to work around this problem and convert the relational data into a database with the generic triplet schema, i.e., a knowledge graph. Mappings between attributes are created as needed, for example, in response to specific business questions, and can themselves be represented in a knowledge graph. We illustrate this process with a concrete example.

Many financial institutions want to create an enterprise knowledge graph that combines internal customer data with licensed third-party data. Some examples of these third party data sets include Dunn & Bradstreet, S&P 500, etc. An example of using an enterprise knowledge graph is to assess risk while making lending decisions. External data contains information such as a company's suppliers. If a company is experiencing financial difficulties, this increases the risk of lending to that company's suppliers. To combine this external data with internal data, the external schemas must be linked with the internal schema of the company. In addition, the company names used in the external sources must be linked to the corresponding customer IDs used by the financial institutions. When using a knowledge graph approach to data integration, the determination of such relationships can be delayed until they are actually needed.

**Knowledge graphs in artificial intelligence**

Knowledge graphs, known as semantic networks, have been used as a representation of artificial intelligence since the early days of the field. Over the years, semantic networks have evolved into different representations such as conceptual graphs, description logics and rule languages. To capture uncertain knowledge, probabilistic graphical models have been invented.

A widely known application of representation languages from semantic networks is the capture of ontologies. An ontology is a formal specification of the conceptualization of a domain. An ontology plays an important role in information exchange and in capturing the basic knowledge of a domain that could be used to reason about and answer questions.

The World Wide Web Consortium (W3C) has standardized a family of knowledge representation languages that are now widely used to capture knowledge on the Internet. We will consider such a language known as the Resource Description Frame (RDF) in the next chapter. This family of languages also includes the Web Ontology Language (OWL) and the Semantic Web Rule Language (SWRL).

Orthogonal to knowledge representation, a central challenge in AI is the knowledge acquisition bottleneck, i.e. how to capture knowledge in the chosen representation in an economically scalable way. Early approaches were based on knowledge engineering. Efforts to automate parts of knowledge engineering have led to techniques such as inductive learning and the current generation of machine learning.

Therefore, it is natural that knowledge graphs are used as the representation of choice for storing automatically learned knowledge. There is also a growing interest in leveraging domain knowledge expressed in knowledge graphs to improve machine learning.

**Knowledge Graphs as a result of Machine Learning**

We will examine how graphs are used as a target output representation for natural language processing and computer vision algorithms.

Feature extraction and relation extraction from text are two fundamental tasks in natural language processing. Information extracted from several parts of the text must be correlated, and knowledge graphs provide a natural support to achieve such a goal. For example, from the sentence shown on the left, we can extract the entities *Albert Einstein*, *Germany*, *Theoretical Physicist*, and *Theory of Relativity*; and the relationships *born in*, *occupation*, and *developed*. Once this extract from the knowledge graph is embedded in a larger knowledge graph, we get additional links (indicated by dotted edges) such as a theoretical physicist is *a kind of* physicist who *practices* physics, and this theory of relativity is a *branch of* physics.

|  |  |  |
| --- | --- | --- |
| |  |  | | --- | --- | | **Albert Einstein** was a **German born theoretical physicist** who developed the **theory of relativity**. |  | |
|  |

The Holy Grail of computer vision is complete image understanding, that is, creating a model that can name and detect objects, describe their attributes and recognize their relationships. Scene understanding would enable important applications such as image retrieval, question answering and robotic interactions. Many advances have been made in recent years towards this goal, including image classification and object detection.

|  |  |
| --- | --- |
| |  | | --- | |  | |
|  |

For example, from the image above, an image understanding system should produce a knowledge graph shown on the right. The nodes of the knowledge graph are the outputs of an object detector. Current research in computer vision focuses on developing techniques to correctly infer relationships between objects, such as a man *holding* a bucket and a horse *feeding* from the bucket, etc. The knowledge graph shown at right is an example of a knowledge graph...

**Knowledge graphs as input for machine learning**

Popular deep machine learning models rely on numerical input which requires that any symbolic or discrete structure be first converted to a numerical representation. *Integrations* that transform a symbolic input into a vector of numbers have emerged as a representation of choice for inputs to machine learning models. We will explain this concept and its relation to knowledge graphs by taking the example of *word plots* and *graph plots*.

*Word plots* have been developed to compute the similarity between words. To understand word plunges, we consider the following set of sentences.

|  |  |
| --- | --- |
| I like knowledge graphs. |  |
| I like databases. |  |
| I like to run. |  |

In the above set of sentences, we will count how often a word appears next to another word and record the counts in a matrix shown below. For example, the word *I* appears next to the word *like* twice, and next to the word *enjoy* once, and therefore its counts for these two words are 2 and 1 respectively, and 0 for every other word. We can calculate the number of other words in the same way to fill the table.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Account | I | Like | Having fun | knowledge | graphics | Databases | operation | . |
| I | 0 | 2 | 1 | 0 | 0 | 0 | 0 | 0 |
| Like | 2 | 0 | 0 | 1 | 0 | 1 | 0 | 0 |
| Having fun | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| Knowledge | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 |
| Graphics | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 |
| Databases | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 |
| Operation | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 |
| . | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 |

The above table is a matrix which is often called the number of *word co-occurrences*. We say that the meaning of each word is captured by the vector in the row corresponding to that word. To compute the similarity between words, we can simply compute the similarity between the vectors that correspond to them. In practice, we are interested in a text that may contain millions of words, and a more compact representation is desired. Since the above matrix is sparse, we can use linear algebra techniques (e.g. singular value decomposition) to reduce its dimensions. The resulting vector corresponding to a word is known as *a word embedding*. Typical word embeddings used today are based on vectors of length 200. There are many variations and extensions of the basic idea presented here. Techniques exist to automatically learn word embeddings for a given text.

The use of word embeddings has been shown to improve the performance of many natural language processing tasks, including feature extraction, relation extraction, parsing, passage retrieval, etc. One of the most common applications of word embeddings is automatic completion of search queries. Word embeddings provide us with a simple way to predict which words are likely to follow the partial query a user has already typed.

Since a text is a sequence of words and word nestings compute the co-occurrences of words in it, we can think of the text as a knowledge graph in which each word is a node, and there is a directed edge between each word and another word immediately following it. Graph dives generalize this notion for the general structure of the network. The goal and approach, however, remain the same: to represent each node in a knowledge graph by a vector, so that the similarity between nodes can be computed as a difference between their corresponding vectors. The vectors of each node are also called graph plots.

To compute the knowledge graph plunges, we define a method to encode each node of the graph into a vector, a function to compute the similarity between the nodes, and then optimize the encoding function. Encoding a node into a vector is also known as *node* embedding. One possible encoding function is to use *random walks* of the knowledge graph (typically 32 to 64 such random walks) and compute the number of node co-occurrences on the knowledge graph producing a matrix similar to the numbers of word co-occurrences in the text. There are many variations of this basic method for computing knowledge graph plots. Just as we encoded a node into a vector, we can also encode the entire graph into a vector called a graph embedding. There are many approaches to computing graph embeddings, but perhaps the simplest approach is to add the vectors for each of the nodes in the graph and obtain a vector representing the entire graph.

We have chosen to explain graph plunges by first explaining word plunges because they are easy to understand and their use is common. Graph plots are a generalization of the word plots. They are a way to input domain knowledge expressed in a knowledge graph into a machine learning algorithm. Graph dives do not induce a knowledge representation, but are a means of transforming a symbolic representation into a numerical representation for consumption by a machine learning algorithm.

Once we have computed knowledge graph embeddings, they can be used for a variety of applications. One obvious use of knowledge graph integrations computed from a friendship graph is to recommend new friends. A more advanced task involves link prediction (i.e. the probability of a link between two nodes). Link prediction in a business graph could be used to identify potential new customers.