DOI:10.1145/3331166

Article development led by acmqueue
queue.acm.org

**Five diverse technology companies show how it's done.**

BY NATASHA NOY, YUQING GAO, ANSHU JAIN, ANANT NARAYANAN, ALAN PATTERSON, AND JAMIE TAYLOR

# Industry-Scale Knowledge Graphs: Lessons and Challenges

KNOWLEDGE GRAPHS ARE critical to many enterprises today: They provide the structured data and factual knowledge that drive many products and make them more intelligent and "magical."

In general, a knowledge graph describes objects of interest and connections between them. For example, a knowledge graph may have nodes for a movie, the actors in this movie, the director, and so on. Each node may have properties such as an actor's name and age. There may be nodes for multiple movies involving a particular actor. The user can then traverse the knowledge graph to collect information on all the movies in which the actor appeared or, if applicable, directed.

Many practical implementations impose constraints on the links in knowledge graphs by defining a *schema* or *ontology*. For example, a link from a movie to its director must connect an object of type Movie to an object of type Person. In some cases the links themselves might have their own properties: a link connecting an actor and a movie might have the name of the specific role the actor played. Similarly, a link connecting a politician with a specific role in government might have the time period during which the politician held that role.

Knowledge graphs and similar structures usually provide a shared substrate of knowledge within an organization, allowing different products and applications to use similar vocabulary and to reuse definitions and descriptions that others create. Furthermore, they usually provide a compact formal representation that developers can use to infer new facts and build up the knowledge—for example, using the graph connecting movies and actors to find out which actors frequently appear in movies together.

This article looks at the knowledge graphs of five diverse tech companies, comparing the similarities and differences in their respective experiences of building and using the graphs, and discussing the challenges that all knowledge-driven enterprises face today. The collection of knowledge graphs discussed here covers the breadth of applications, from search, to product descriptions, to social networks:

- Both Microsoft's Bing knowledge graph and the Google Knowledge Graph support search and answering questions in search and during conversations. Starting with the descriptions and connections of people, places, things, and organizations, these graphs include general knowledge about the world.
- Facebook has the world's largest social graph, which also includes information about music, movies, celebrities, and places that Facebook users care about.

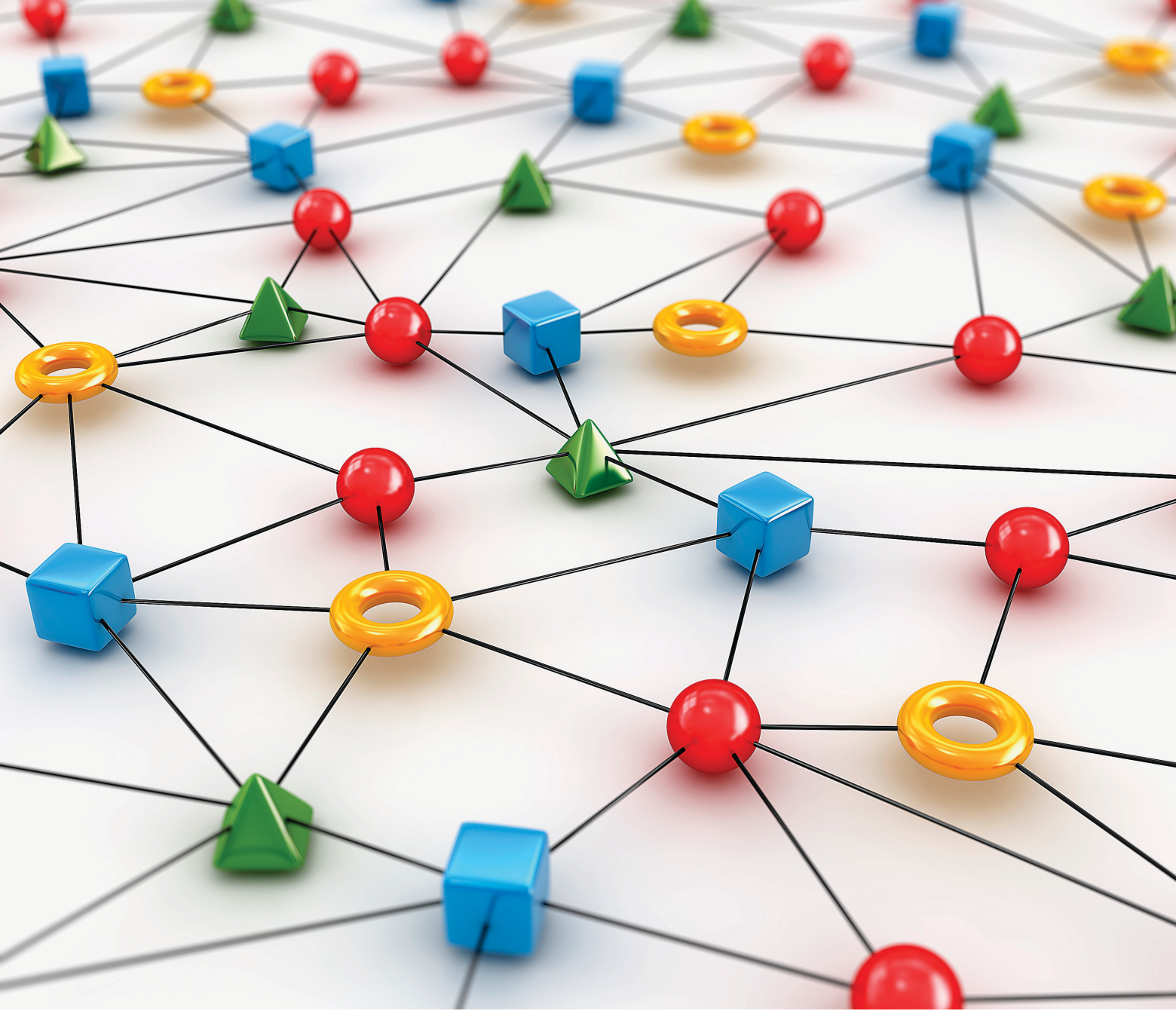

IMAGE BY ADEMPERCEM/SHUTTERSTOCK

▸ The Product Knowledge Graph at eBay, currently under development, will encode semantic knowledge about products, entities, and the relationships between them and the external world.

▸ The Knowledge Graph Framework for IBM's Watson Discovery offerings addresses two requirements: one focusing on the use case of discovering nonobvious information, the other on offering a "Build your own knowledge graph" framework.

The goal here is not to describe these knowledge graphs exhaustively, but rather to use the authors' practical experiences in building knowledge graphs in some of the largest technology companies today as a scaffolding to highlight the challenges that any enterprise-scale knowledge graph will face and where some innovative research is needed.

## What's In a Graph? Design Decisions

Let's start by describing the five knowledge graphs and the decisions that went into each design and determining the scope of each graph. The different applications and product goals for each one resulted in different approaches and architectures, though many of the challenges are shared by all the enterprises. The accompanying table summarizes the properties of these knowledge graphs.

**Microsoft.** Engineers and scientists at Microsoft have been working on large-scale graphs for many years. This work included building the end-to-end system from the underlying research, as well as a global-scale service for hundreds of millions of users. Across the company, there are several major graph systems, each bringing specific challenges around creating the graph and keeping it up to date. Many different products can use a knowledge graph to bring value to consumers. The following are some of the graphs at Microsoft:

▸ The Bing knowledge graph contains information about the world and powers question answering on Bing. It contains entities such as people, places, things, organizations, locations, and so on, as well as the actions that a user might take (for example, to play a video or buy a song). This is the largest knowledge graph at Microsoft, as its

aim is to contain general knowledge about the entire world.

▸ The Academic graph is a collection of entities such as people, publications, fields of study, conferences, and locations. It allows a user to see connections between researchers and pieces of research that may otherwise be hard to determine.

▸ The LinkedIn graph contains entities such as people, jobs, skills, companies, locations, and so on. The LinkedIn Economic graph is based on 590 million members and 30 million companies, and is used to find economy-level insights for countries and regions.

The Bing search engine displays a knowledge panel from the Bing knowledge graph when there is additional useful information. For example, a search for the film director James Cameron reveals information such as his date of birth, height, movies and TV shows he directed, previous romantic partners, TED Talks he gave, and Reddit "Ask Me Anything" questions and answers (through partnership with Reddit). A search for a different type of entity returns completely different information—for example, searching for "Woodblock restaurant" results in an extract from the menu, professional critic and user reviews, as well as the option to book a table.

All of these graph systems—as would probably be the case with any large graph system—have three key determinants of quality and usefulness:

▸ *Coverage.* Does the graph have all the required information? The answer is always effectively no, because developers are always looking for new ways to provide value to users and for new sources of information.

▸ *Correctness.* Is the information correct? How do you know if two sources of information are actually about the same fact, and what do you do if they conflict? Answering these questions is a huge area of study and investment by itself.

▸ *Freshness.* Is the content up to date? It may have been correct at one time but gone stale. Freshness will vary for something that changes almost constantly (a stock price) compared with something that changes rarely (the capital of a country), with many different kinds of information in between.

To generate knowledge about the world, data is ingested from multiple sources, which may be very noisy and contradictory, and must be collated into a single, consistent, and accurate graph. The final fact that a user sees is the tip of an iceberg—a huge amount of work and complexity is hidden below. For example, there are 200 Will Smiths in Wikipedia alone, and the Bing knowledge result for the actor Will Smith is composed from 108,000 facts taken from 41 websites.

**From search to conversation.** Knowledge graphs power advanced AI, allowing single queries to be turned into an ongoing conversation. Specifically, this allows a user to have a conversation with the system and to have the system maintain the context through each turn of the conversation. For example, in a future scenario a user could say to Bing, "Show me all the countries in the world where it's over 70 degrees Fahrenheit right now," and once the system returns the answer, the user could say, "Show me those within a two-hour flight."

You can take the same idea further to enable a full conversational experience. For example, a user could say, "I want to travel to NYC two days before Thanksgiving and stay for a week," and the system would use the underlying knowledge graph to make sense of the query and then request missing pieces of information. In this example, the system needs to know that "NYC" could mean "JFK Airport" and that Thanksgiving is November 22. It then must know how to carry out a flight search, which requires a start location and a destination location. The system would then have to know the next line of the conversation must determine the start location, so it would say, "Okay, booking a flight to JFK from November 20 to 27. Where will you be flying from?"

**Google.** With more than 70 billion assertions describing a billion entities, the Google Knowledge Graph covers a wide swath of subject matter and is the result of more than a decade of data-contribution activity from a diverse set of individuals, most of whom have never had experience with knowledge-management systems.

Perhaps more important, Knowledge Graph serves as a long-term, stable source of class and entity identity that many Google products and features use behind the scenes. Outside users and developers can observe these features when they use services such as YouTube and Google Cloud APIs. This focus on identity has allowed Google to transition to "things not strings." Rather than simply returning the traditional "10 blue links," Knowledge Graph helps Google products interpret user requests as references to concepts in the world of the user and to respond appropriately.

Google's Knowledge Graph is perhaps most visible when users issue queries about entities and the search results include an array of facts about the entities that are served from Knowledge Graph. For example, a query for "I.M. Pei" produces a small panel in the search results with information about the architect's education, awards, and

**Common characteristics of the knowledge graphs.**

| | Data model | Size of the graph | Development stage |
|---|---|---|---|
| **Microsoft** | The types of entities, relations, and attributes in the graph are defined in an ontology. | ~2 billion primary entities, ~55 billion facts | Actively used in products |
| **Google** | Strongly typed entities, relations with domain and range inference | 1 billion entities, 70 billion assertions | Actively used in products |
| **Facebook** | All of the attributes and relations are structured and strongly typed, and optionally indexed to enable efficient retrieval, search, and traversal. | ~50 million primary entities, ~500 million assertions | Actively used in products |
| **eBay** | Entities and relation, well-structured and strongly typed | Expect around 100 million products, >1 billion triples | Early stages of development and deployment |
| **IBM** | Entities and relations with evidence information associated with them. | Various sizes. Proven on scales documents >100 million, relationships >5 billion, entities >100 million | Actively used in products and by clients |

the significant structures he designed.

The Knowledge Graph also recognizes that certain kinds of interactions can take place with different entities. A query for "The Russian Tea Room" provides a button to make a reservation, while a query for "Rita Ora" provides links to her music on various music services.

At the scale of the Google Knowledge Graph, a single individual can not remember, let alone manage, the detailed structures used throughout the graph. To ensure the system remains consistent over time, Google built its Knowledge Graph from a basic set of low-level structures. It replicated similar structures and reasoning mechanisms at different levels of abstraction, conceptually bootstrapping the structure from a number of basic assertions. For example, to check specific invariant constructions, Google leveraged the idea that types were themselves instances of types to introduce the notion of metatypes. It could then reason about the metatypes to verify the finer-grained types did not violate the invariants it was interested in. It can validate that time-independent identities are not subclasses of structures, which are time-dependent. This scalable level of abstraction was relatively easy to add in a manner that worked out of the box because it was built upon the same low-level entailments on which the rest of the system was based.

This meta-level schema also allows validation of data at scale. For example, you can validate that painters existed before their works of art were created by identifying the painters as the "origin" of their painted work "products" and applying a general check on all relations between these metaclasses.

At a slightly higher conceptual level, Knowledge Graph "understands" that authors are distinct from their creative works, even though these entities are frequently conflated in colloquial expressions. Similarly, creative works may have multiple expressions that are themselves distinct. This ontological knowledge helps maintain the identity of entities as the graph grows.

Building the Knowledge Graph through these self-describing layers not only simplifies consistency checking by machines, but also makes the Knowledge Graph easier for internal users to understand. Once new developers have been trained on the fundamentals of Knowledge Graph organization, they can understand the full extent of its inventory of structures. Similarly, by keeping the structure of the graph tied to a few core principles and exposing meta-relations explicitly in schemas, finding and comprehending new schema structures is simplified for internal developers.

**The system would use the underlying knowledge graph to make sense of the query and then request missing pieces of information.**

**Facebook** is known for having the world's largest social graph. Facebook engineers have built technology over the past decade to enable rich connections between people. Now they are applying the same technology to building a deeper understanding of not just people, but also the things that people care about.

By modeling the world in a structured manner and at scale, Facebook engineers were able to unlock use cases that a social graph by itself could not fulfill. Even seemingly simple things, such as structured understanding of music and lyrics when combined with software that detects when people are referencing them, can enable serendipitous moments between individuals. Many experiences in Facebook's products today, such as helping people plan movie outings on Messenger, are powered by the knowledge graph.

Facebook's knowledge graph focuses on the most socially relevant entities, such as those that are most commonly discussed by its users: celebrities, places, movies, and music. As the Facebook knowledge graph continues to grow, developers focus on those domains that have the greatest chance of delivering utility and delightful user experiences.

Coverage, correctness, structure, and constant change all drive the design of the Facebook knowledge graph:

▸ Coverage means being exhaustive in a domain that is being modeled. The default stance is multiprovider, which means that the entire graph-production system is built with the assumption that data will be received from multiple sources, all providing (sometimes conflicting) information about overlapping sets of entities. The Facebook knowledge graph deals with the conflicting information in one of two ways: the information is deemed to be sufficiently low confidence to justify dropping it; or conflicting views are incorporated into the entity by retaining

provenance and an inferred confidence level about the assertion.

▸ Correctness does not mean the knowledge graph always knows the "right" value for an attribute, but rather that it is always able to explain why a certain assertion was made. Therefore, it keeps provenance for all data that flows through the system, from data acquisition to the serving layer.

▸ Structure means the knowledge graph must be self-describing. If a piece of data is not strongly typed or does not fit the schema describing the entity, then the graph attempts to do one of the following: convert the data into the expected type (for example, performing simple type coercion, handling incorrectly formatted dates); extract structured data that matches the type (for example, run natural language processing, NLP) on unstructured text such as user reviews to convert into typed slots); or leave it out entirely.

▸ Lastly, the Facebook knowledge graph is designed for constant change. The graph is not a single representation in a database that is updated when new information is received. Instead, the graph is built from scratch, from the sources, every day, and the build system is idempotent—producing a complete graph at the end of it.

An obvious place for a Facebook knowledge graph to start is the Facebook pages ecosystem. Businesses and people create pages on Facebook to represent a huge range of ideas and interests. Furthermore, having the owner of an entity make assertions about it is a valuable source of data. As with any crowd-sourced data, however, it is not without its challenges.

Facebook pages are very public facing, and millions of people interact with them every day. Thus, the interests of a page owner don't always align with the requirements of a knowledge graph.

Most commonly, pages and entities do not have a strict 1:1 mapping, as pages can represent collections of entities (for example, movie franchises). Data can also be incomplete or very unstructured (blobs of text), which makes it more difficult to use in the context of a knowledge graph.

Facebook's biggest challenge has been to leverage data found on its pages and to combine it with other more structured sources of data to achieve the goals of a clean, structured knowledge graph. A useful tool for Facebook has been to think of the graph as the model and a Facebook page as the view—a projection of an entity or collection of entities that reside in the graph.

**The Discovery use case creates new knowledge that is not directly present in domain documents or data sources.**

**eBay** is building its Product Knowledge Graph, which will encode semantic knowledge about products, entities, and their relationships with each other and the external world. This knowledge will be key to understanding what a seller is offering and a buyer is looking for and intelligently connecting the two, a key part of eBay's marketplace technology.

For example, eBay's knowledge graph can relate products to real-world entities, defining the identity of a product and why it might be valuable to a buyer. A basketball jersey for the Chicago Bulls is one product, but if it is signed by Michael Jordan, it is a very different product. A postcard from 1940 in Paris might be just a postcard; knowing that Paris is in France and that 1940 is during World War II changes the product entirely.

Entities in the knowledge graph can also relate products to each other. If a user searches for memorabilia of Lionel Messi and the graph indicates that Lionel Messi plays for Futbol Club Barcelona, then, maybe, merchandise for that club is of interest, too. Perhaps memorabilia for other famous Barcelona players will be of interest to this shopper. Related merchandise should include soccer-based products such as signed shirts, strips, boots, and balls. This idea can extend from sports to music, film, literature, historical events, and much more.

Just as important as entity relations is understanding the products themselves and their relationships. Knowing that one product is an iPhone and another is a case for an iPhone is obviously important. But the case might fit some phones and not others, so eBay needs to model the parts and accessory sizes. Knowing the many variants and relationships of products is also important: Which products are manufacturer variants of one product? Do they come in different sizes, capacities, or colors? Which are comparable—meaning they have mostly the same specifications but perhaps different brands or colors? The system also needs to understand

products that go together as a set, say in bundles, kits, or even fashion outfits.

As with other knowledge graphs, eBay must cope with scale. At any one time there may be more than one billion active listings across thousands of categories. These listings might include hundreds of millions of products and tens of billions of attributes specified for those products.

There are several different users of the eBay Knowledge Graph, and these users have very different service-level requirements. When the search service needs to understand a user's query, the knowledge graph must power an answer that takes milliseconds. At the other end of the scale, large graph queries could take hours to run.

To cope with these challenges, eBay engineers have designed an architecture that provides them with flexibility, while ensuring that the data is consistent. The knowledge graph uses a replicated log for all writes and edits to the graph. The log provides a consistent ordered view of the data. This approach enables multiple back-end data stores that meet different use cases. Specifically, there is a flattened document store for serving search queries with low latency and a graph store for doing long-running graph analysis. Each of these stores simply appends its operations to the write log and gets the additions and edits to the graph in a guaranteed order. As a result, each store will be consistent.

**IBM** developed its Knowledge Graph Framework, used by Watson Discovery Services and its associated offerings, which have been deployed in many industry settings outside of IBM. IBM Watson uses the Knowledge Graph Framework in two distinct ways: First, the framework directly powers Watson Discovery, which focuses on using structured and unstructured knowledge to discover new, nonobvious information, and the associated vertical offerings on top of Discovery; second, the framework allows others to build their own knowledge graphs with the prebuilt knowledge graph as the core.

The Discovery use case creates new knowledge that is not directly present in domain documents or data sources. This new knowledge can be surprising and anomalous. While search and exploration tools access knowledge that is already available in the sources available to the system, they are necessary but not sufficient for Discovery. Nonobvious discovery includes new links between entities (for example, a new side effect of a drug, an emerging company as an acquisition target or sales lead), a potential new important entity in the domain (for example, a new material for display technologies, a new investor for a particular investment area), or changing significance of an existing entity (an increasing stake by an investor in an organization, or increasing interaction between a person of interest and some criminal in an intelligence-gathering scenario).

Given its wide enterprise customer base applying cognitive technologies in various domains, IBM focused on creating a framework for clients and client teams to build their own knowledge graphs. Industry teams at IBM leverage this framework to build domain-specific instances. Clients exist in several domains ranging from consumer-oriented research in banking and finance, insurance, IT services, media and entertainment, retail, and customer service, to industries focused almost entirely on deep discovery—especially scientific domains such as life sciences, oil and gas, chemicals and petroleum, defense, and space exploration. This breadth requires the framework have all of the machinery that clients need to build and manage a knowledge graph themselves. Some of the key technologies built into the framework include document conversion, document extraction, passage storage, and entity normalization.

The following are some of the key insights and lessons that IBM engineers learned from both building the knowledge graph for Watson Discovery and deploying the system in other industries.

▸ *Polymorphic stores offer a solution.* The IBM Watson Knowledge Graph uses a polymorphic store, supporting multiple indices, database structures, in-memory, and graph stores. This architecture splits the actual data (often redundantly) into one or more of these stores, allowing each store to address specific requirements and workloads. IBM engineers and researchers addressed a number of challenges such as keeping these multiple stores in sync, allowing communication between the stores through microservices, and allowing ingestion of new knowledge or reprocessing raw data in a way that does not require reloading or rebuilding the entire graph.

▸ *Evidence must be primitive to the system.* The main link between the real world (which developers often try to model) and the data structures holding the extracted knowledge is the "evidence" of the knowledge. This evidence is often the raw documents, databases, dictionaries, or image, text, and video files from which the knowledge is derived. When it comes to making pointed and useful contextual queries during a discovery process, the metadata and other associated information often play a role in inference of the knowledge. Thus, it is critical not to lose the linkage between the relationships stored in the graph and where those relationships come from.

▸ *Push entity resolution to runtime through context.* Resolving ambiguous references to entities referenced by partial names, surface forms, or multiple entities having the same names is a classic problem in understanding natural language. In the field of knowledge discovery, however, developers often look for the nonobvious patterns where an entity is not behaving in its well-understood form or appears in a novel context. Thus, a disambiguation of an entity too early in the process of knowledge-graph creation conflicts with the very goal of discovery. It is better to leave those utterances unresolved or disambiguate them to multiple entities, and then during runtime use the context of the query to resolve the entity name.

## Challenges Ahead

The requirements, coverage, and architectures of the knowledge graphs discussed here differ quite a bit, but many of the challenges appear consistently across most implementations. These include challenges of scale, disambiguation, extraction of knowledge from heterogeneous and unstructured sources, and managing knowledge evolution. These challenges have been at the forefront of research for years, yet they continue to baffle industry practitioners. Some of the challenges are present in some of the systems but may

be less relevant in other settings.

**Entity disambiguation and managing identity.** While entity disambiguation and resolution is an active research area in the semantic Web, and now in knowledge graphs for several years, it is almost surprising that it continues to be one of the top challenges in the industry almost across the board. In its simplest form, the challenge is in assigning a unique normalized identity and a type to an utterance or a mention of an entity. Many entities extracted automatically have very similar surface forms, such as people with the same or similar names, or movies, songs, and books with the same or similar titles. Two products with similar names may refer to different listings. Without correct linking and disambiguation, entities will be incorrectly associated with wrong facts and result in incorrect inference downstream.

While these problems might seem obvious in smaller systems, when identity management must be done with a heterogeneous contributor base and at scale, the problem becomes much more challenging. How can identity be described in a way that different teams can agree on it and know what the other teams are describing? How can developers be sure to have enough human-readable information to adjudicate conflicts?

**Type membership and resolution.** Most current knowledge-graph systems allow each entity to have multiple types, and the specific type may matter in different circumstances. For example, Barack Obama is a person, but also a politician and actor—a vastly more popular politician and not a very well-known actor. Cuba can be a country or may refer to its government. In some cases, knowledge-graph systems defer the type assignment to runtime: Each entity describes its attributes, and the application uses a specific type and collection of attributes depending on the user task.

While criteria for class membership might be straightforward early on, as the universe of instances grows, enforcing these criteria while maintaining semantic stability becomes challenging. For example, when Google defined the category for "sports" in its knowledge graph, e-sports did not exist. So, how does Google maintain the category identity for sports while also including e-sports?

**Managing changing knowledge.** An effective entity-linking system also needs to grow organically based on its ever-changing input data. For example, companies may merge or split, and new scientific discoveries may break an existing entity into multiples. When a company acquires another company does the acquiring company change identity? What about a division being spun out? Does identity follow the acquisition of the rights to a name?

While most knowledge-graph frameworks are becoming efficient at storing a point-in-time version of a knowledge graph and managing instantaneous changes to the knowledge graphs to evolve the graph, there is a gap in being able to manage highly dynamic knowledge in the graphs.[4] A fundamental understanding of temporal constructs, history, and change with history is needed to capture these changes. Furthermore, the ability to manage updates through multiple stores (for example, IBM's polymorphic stores) is necessary.

There are a lot of considerations around the integrity of the update process, eventual consistency, conflicting updates, and, simply, runtime performance. There may be an opportunity to think of different variations of existing distributed data stores designed to handle incremental cascade updates. It is also critical to manage changing schemas and type systems, without creating inconsistencies with the knowledge already in the system. Google, for example, addresses this problem by conceptualizing the metamodel layer into multiple layers. The basic lower layers remain fairly constant and higher levels are built through the notion of metatypes (which are really instances of types), which can be used to enrich the type system.

**Knowledge extraction from multiple structured and unstructured sources.** Despite the recent advances in natural language understanding, the extraction of structured knowledge (which includes entities, their types, attributes, and relationships) remains a challenge across the board. Growing the graphs at scale requires not only manual approaches, but also unsupervised and semi-supervised knowledge extraction from unstructured data in open domains.

For example, in the eBay Product Knowledge Graph, many graph relationships are extracted from unstructured text in listings and seller catalogs; the IBM Discovery knowledge graph relies on documents as evidence for the facts represented in the graphs. Traditional supervised machine-learning frameworks require labor-intensive human annotations to train knowledge-extraction systems. This high cost can be alleviated or eliminated by adopting fully unsupervised approaches (clustering with vector representations) or semi-supervised techniques (distant supervision with existing knowledge, multi-instance learning, active learning, and so on). Entity recognition, classification, text, and entity embeddings all prove useful tools to link our unstructured text to entities we know about in the graph.[3]

**Managing operations at scale.** It is probably not surprising that all of the knowledge-graph systems described here face the challenge of managing the graphs at scale. This dimension often makes the problems that have been addressed in multiple forms in the academic and research community (such as disambiguation and unstructured data extraction) present new challenges in industry settings. Managing scale is the underlying challenge that affects several operations related to performance and workload directly. It also manifests itself indirectly as it affects other operations, such as managing fast incremental updates to large-scale knowledge graphs as at IBM or managing consistency on a large evolving knowledge graph as at Google.[1]

## Other Key Challenges

In addition to these truly pervasive challenges, the following challenges will be critical to the efforts described in this article. These are interesting and intriguing subjects for research and academic communities.

**Knowledge-graph semantic embeddings.** With a large-scale knowledge graph, developers can build high-dimensional representations of entities and relations. The resulting embeddings will greatly benefit many machine-learning, NLP, and AI tasks as sources of features and constraints, and

can form the basis for more sophisticated inferences and ways to curate training data. Deep-learning techniques can be applied to problems of entity deduplication and attribute inference.[2]

**Knowledge inference and verification.** Making sure that facts are correct is a core task in constructing a knowledge graph, and with a huge scale it is not remotely possible to verify everything manually. This requires an automated approach: advances in knowledge representation and reasoning, probabilistic graphical models, and natural language inferences can be used to construct an automatic or semi-automatic system for consistency checking and fact verification.

**Federation of global, domain-specific, and customer-specific knowledge.** In a case like IBM clients, who build their own custom knowledge graphs, the clients are not expected to tell the graph about basic knowledge. For example, a cancer researcher is not going to teach the knowledge graph that skin is a form of tissue, or that St. Jude is a hospital in Memphis, Tennessee. This is known as "general knowledge," captured in a general knowledge graph.

The next level of information is knowledge that is well known to anybody in the domain—for example, carcinoma is a form of cancer or NHL more often stands for non-Hodgkin lymphoma than National Hockey League (though in some contexts it may still mean that—say, in the patient record of an NHL player). The client should need to input only the private and confidential knowledge or any knowledge that the system does not yet know. Isolation, federation, and online updates of the base and domain layers are some of the major issues that surface because of this requirement.

**Security and privacy for personalized, on-device knowledge graphs.** Knowledge graphs by definition are enormous, since they aspire to create an entity for every noun in the world, and thus can only reasonably run in the cloud. Realistically, however, most people do not care about all entities that exist in the world, but rather a small fraction or subset that is personally relevant to them. There is a lot of promise in the area of personalizing knowledge graphs for individual users, perhaps even to the extent that they can shrink to a small enough size to be shippable to mobile devices. This will allow developers to keep providing user value in a privacy-respecting manner by doing more on-device learning and computation, over local small knowledge-graph instances. (We are eager to collaborate with the research community in pursuit of this goal.)

**Multilingual knowledge systems.** A comprehensive knowledge graph must cover facts expressed in multiple languages and conflate the concepts expressed in those languages into a cohesive set. In addition to the challenges in knowledge extraction from multilingual sources, different cultures may conceptualize the world in subtly different ways, which poses challenges in the design of the ontology as well.

## Conclusion

The natural question from our discussion in this article is whether different knowledge graphs can someday share certain core elements, such as descriptions of people, places, and similar entities. One of the avenues toward sharing these descriptions could be to contribute them to Wikidata as a common, multilingual core. In the nearer term, we hope to continue sharing the results of research that each of us may have done with researchers and practitioners outside of our companies.

Knowledge representation is a difficult skill to learn on the job. The pace of development and the scale at which knowledge-representation choices impact users and data do not foster an environment in which to understand and explore its principles and alternatives. The importance of knowledge representation in diverse industry settings, as evidenced by the discussion in this article, should reinforce the idea that knowledge representation should be a fundamental part of a computer science curriculum—as fundamental as data structures and algorithms.

Finally, we all agree that AI systems will unlock new opportunities for organizations in how they interact with customers, provide unique value in their space, and transform their operations and workforces. To realize this promise, these organizations must figure out how to build new systems that unlock knowledge to make them truly intelligent organizations.

*The article summarizes and expands on a panel discussion the authors conducted at the International Semantic Web Conference in Asilomar, CA, in Oct. 2018* (https://bit.ly/2ZYVLJh). *The discussion is based on practical experiences and represents the views of the authors and not necessarily their employers.*

**Related articles on queue.acm.org**

**Schema.org: Evolution of Structured Data on the Web**
*R.V. Guha, D. Brickley, and S. Macbeth*
https://queue.acm.org/detail.cfm?id=2857276

**Hazy: Making it Easier to Build and Maintain Big-data Analytics**
*A. Kumar, F. Niu, and C. Ré*
https://queue.acm.org/detail.cfm?id=2431055

**A Primer on Provenance**
*L. Carata, et al.*
https://queue.acm.org/detail.cfm?id=2602651


**References**

1. Höffner, K., Walter, S., Marx, E., Usbeck, R., Lehmann, J. and Ngonga Ngomo, A.C. Survey on challenges of question answering in the semantic Web. *Semantic Web 8*, 6 (2017), 895–920.
2. Lin, Y., Liu, Z., Sun, M., Liu, Y. and Zhu, X. Learning entity and relation embeddings for knowledge graph completion. In *Proceedings of the Assoc. Advancement of Artificial Intelligence 15*, (2015), 2181–2187.
3. Nickel, M., Murphy, K., Tresp, V. and Gabrilovich, E. 2016. A review of relational machine learning for knowledge graphs. In *Proceedings of the IEEE 104*, 1 (2016), 11–33.
4. Paulheim, H., Knowledge graph refinement: a survey of approaches and evaluation methods. *Semantic Web 8*, 3 (2017), 489–508.



**Natasha Noy** is a scientist at Google, where she works on making structured data accessible and leads Google Dataset Search. Previously, she worked on ontology engineering and semantic Web at Stanford University, Stanford, CA, USA.

**Yuqing Gao** is the general manager of Microsoft's Artificial Intelligence – Knowledge Graph organization. She has been a key leader behind intelligent features for Microsoft Office products, Bing Entity Search, and other prominent AI knowledge-driven Microsoft technologies.

**Anshu Jain** works at IBM Watson, where he is responsible for the architecture of the core knowledge and language capabilities, including Knowledge Graph, natural language understanding, and Watson Knowledge Studio, among others.

**Anant Narayanan** is an engineering manager at Facebook, where he helps build knowledge platforms to develop a deeper understanding of entities and relationships. Previously, he led the development of large-scale data pipelines at Ozlo to support conversational AI systems.

**Alan Patterson** is a Distinguished Engineer at eBay, heading up eBay's efforts to build a product knowledge graph that contains eBay's knowledge of products, as well as organizations, brands, people, places, and standards. Previously, he worked at the startup True Knowledge (also Evi.com).

**Jamie Taylor** manages the Schema Team for Google's Knowledge Graph. The team's responsibilities include extending KG's underlying semantic representation, growing coverage of the ontology, and enforcing semantic policy. Previously, he worked for Metaweb Technologies.