How Large Language Models and Knowledge Graphs Can Transform Enterprise Search

Published 28 March 2023 - ID G00784270 - 32 min read

By Analyst(s): Darin Stewart

Initiatives: Digital Workplace and CRM for Technical Professionals

> Search users expect answers to questions, not lists of links to resources that might contain the answers. Application technical professionals can use large language models and knowledge graphs to transform enterprise search to meet these demands.

Overview

Key Findings

- Search users expect a Google-like experience that includes natural language queries and answers, knowledge panels and recommended resources. Lexical and traditional term frequency-inverse document frequency (TF-IDF) search are not sufficient to meet these expectations.
- Large language models (LLMs), such as Generative Pretrained Transformer (GPT), Bidirectional Encoder Representations from Transformers (BERT) and Language Model for Dialogue Applications (LaMDA), used in conjunction with knowledge graphs, are key to providing modern search capabilities.
- LLMs and knowledge graphs can greatly enhance search platforms, but they are not a replacement for search platforms. Most major search vendors now support integration with these technologies.

Recommendations

To improve the usability and effectiveness of enterprise search by integrating LLMs and knowledge graphs into their search environments, application technical professionals responsible for digital workplaces and CRM for technical professionals should:

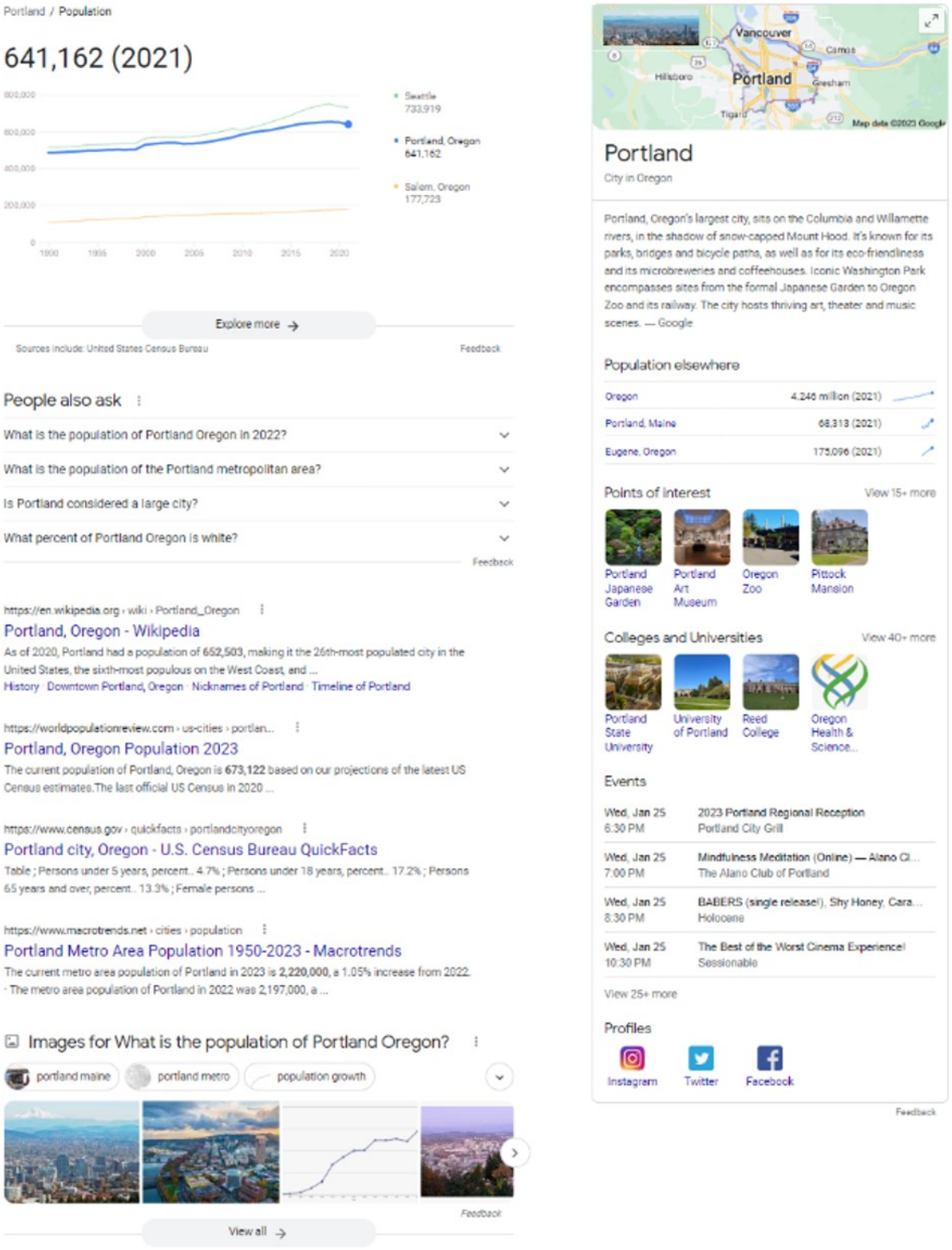
- Use an LLM to identify and disambiguate common entities as part of the index creation pipeline. Supplement it with a knowledge graph to create context-aware vector embeddings that are available to the search engine.

- Start with existing enterprise knowledge graphs, when available. Use the output of the vector-embedding process to further enrich these knowledge graphs and to seed new knowledge graphs.
- Create personal knowledge graphs for staff members as the basis of access control, personalization and expertise discovery.

Analysis

Siri has spoiled us. So have Alexa, Google and even Cortana. We ask them questions. They give us answers. They do not demand keywords or make us hunt through lists of links in the hope that one of the myriad pages or documents referenced might contain the information we are looking for. We just ask. They just answer. If I ask Google "What is the population of Portland, Oregon?" Google responds with "641,162." I am also provided with a "knowledge panel" containing additional information about Portland. Lastly, I get the traditional list of links pointing at webpages pertinent to the population of Portland (see Figure 1). This is the type of search result that users have come to expect on the web. Within the enterprise, however, they are usually disappointed.

Figure 1: Google Search Results



Source: Google

Most enterprise search implementations continue to rely on some variation of basic crawl and index. Documents, webpages and other resources are parsed into their basic elements, organized according to some schema, and then ranked by whatever proprietary magic their chosen platform has to offer.

This generic approach suffers from two fundamental flaws. First, it treats the query as a string of characters or a list of words, rather than as a proper question capturing the user's intent. Second, the index does not capture the context of the information it contains. Items in the index are undifferentiated and unrelated to other elements of the index.

These weaknesses are occasionally slightly mitigated by rudimentary natural language processing or basic metadata and content enrichment. Even so, the elements of both the query and the index remain little more than atomized text to be matched to keywords. This approach will never be enough to meet users' demands to "just make it like Google!"

Google distinguishes its search capabilities by addressing both limitations of traditional search. Google understands queries and, when appropriate, treats them as questions. Google also understands the information space being searched. In other words, Google searches for "things not strings."

There are countless innovations that go into the "secret sauce" of Google search, but two in particular are relevant to enterprise search: large language models (LLMs) and knowledge graphs. Individually, these are powerful technologies for improving enterprise search. When used together they can be transformational.

Large Language Models for Search

LLMs have recently captured the public's imagination with the debut of ChatGPT, but they are not a particularly new innovation. Google introduced and adopted BERT in 2019 for English language queries. [1] Within two months, BERT was used to understand queries in more than 70 languages. Google has recently announced a new conversational search service, BARD, based on a different LLM called LaMDA. [2] Similarly, Microsoft has introduced a new version of Bing powered by GPT and has invested over $10 billion in this LLM's creator, OpenAI. [3] Needless to say, the big players in search are betting big on LLMs.

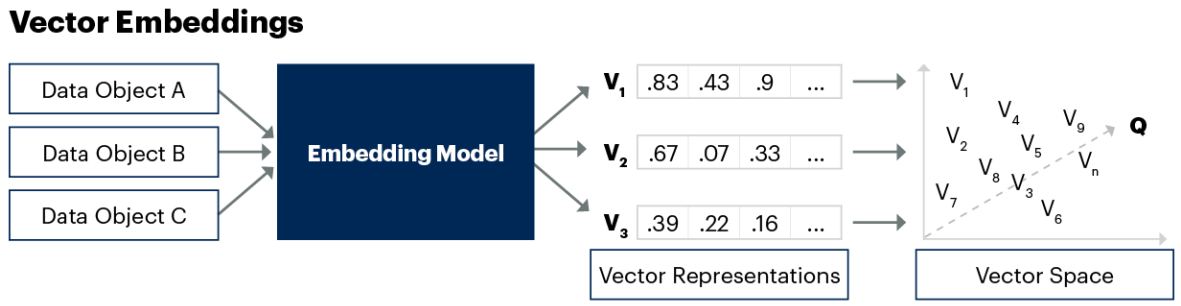
An LLM is a specialized type of artificial intelligence (AI) that has been trained on vast amounts of text, typically billions of words, resulting in tens (or hundreds) of billions of parameters. This enables an LLM both to interpret textual input and generate human-like textual output. In other words, an LLM can help a search engine understand a question and formulate an answer.

A primary mechanism enabling this ability is vector embedding, which is central to semantic vector search (see Figure 2). This approach is currently being trumpeted by several search vendors, including Elastic, Lucidworks, Pinecone, Sinequa and others. A vector embedding is a fixed-length, high-dimensional, numeric representation of a data item that captures its features. In other words, a vector embedding is a list of numbers that captures something's meaning and context. Vector embeddings are simple to generate from a pretrained model, requiring only a few lines of code. For example, generating vector embeddings from the universal sentence encoder family of pretrained models, available from TensorFlow Hub, requires only a bit of Python:

```
import tensorflow_hub as hub


embed = hub.load("https://tfhub.dev/google/universal-sentence-encoder/4")

embeddings = embed([

"The quick brown fox jumps over the lazy dog.."

"I am a sentence for which I would like to get its embedding"])

print(embeddings)
```

This approach works on any type of data — including audio, images, video and so on — and is not restricted to text. Determining how closely related two things are is a matter of measuring the distance between their vectors. The best match for some vectors is basically the approximate nearest neighbor (ANN). This is particularly useful for search.

Figure 2: Vector Embeddings



A vector embedding is a fixed-length, high-dimensional, numeric representation of a data item that captures its features and facilitates semantic search.

Source: Gartner
Note: Q = Query
784270_C

Gartner

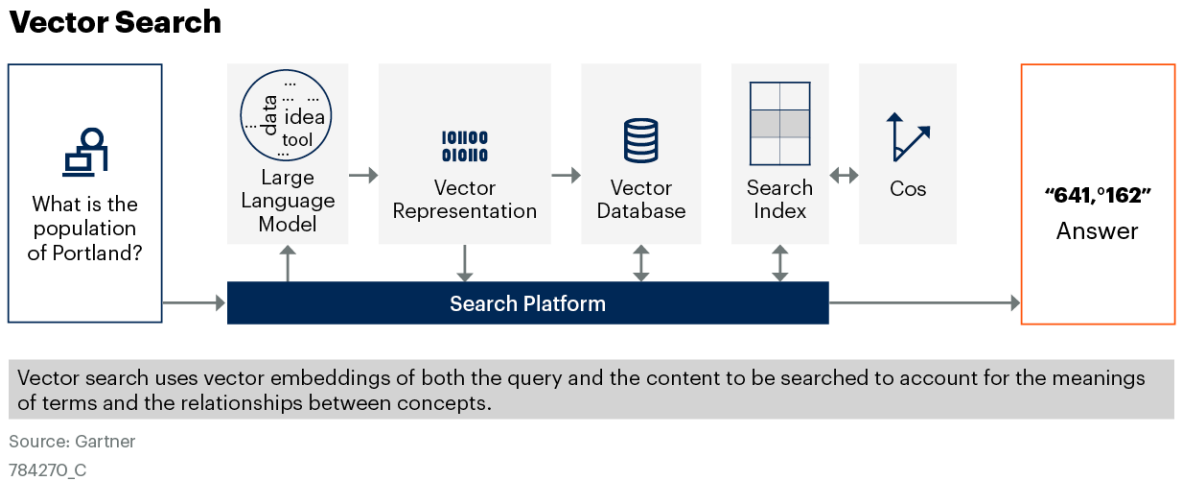
Traditional search engines treat content lexically. They match terms in a query with terms in a document and count how often those terms appear (this is an oversimplification, but the practice remains a foundation of lexical approaches to search). Vector search treats content *semantically* by using vector embeddings of both the query and the content to be searched to account for the meanings of terms and the relationships between concepts. Generating vector embeddings is a critical step in this process as deep-learning models, central to modern search, do not directly utilize unstructured content. That content must first be converted to a numeric form or vector. When both a query and the content to be searched are vectorized, finding the right answer boils down to finding the query vector's nearest neighbor in the vectorized corpus. This enables better queries, more precise search, and much richer and more accurate answers.

Vector embeddings can be created in several ways. Feature engineering relies on domain knowledge to identify, extract and encode salient characteristics of data. This approach can be applied to specialized areas, such as medical imaging or fraud detection, but is too dependent on human expertise and does not scale. A better approach is to use pretrained embeddings generated by models such as word2vec for text or VGG for images. A wide range of such models for an equally wide range of data types is available at the Hugging Face model hub. A drawback of this approach is that the character and context of the data being vectorized is limited to what is explicitly present in the corpus being processed and the features used to train the model. A better approach is to use an LLM trained on a radically larger dataset to generate vector embeddings.

Think of this as if two people were describing a particular wine. The first person enjoys wine but does not know much about the topic. They may describe a wine as red or white, dry or sweet, and not have much else to say about it. While accurate, these basic characteristics are not particularly helpful for identifying a specific vintage. The second person is a professional sommelier, trained to detect and articulate all the subtleties that distinguish one wine from another. They would describe a wine with a wide variety of characteristics and very precise terminology, which would identify a particular wine not only as separate and distinct from others but perhaps as being unique.

In a similar way, LLMs can generate much richer vector embeddings than simpler, more limited pretrained models (see Figure 3). A vector generated by a model trained on a specific corpus, such as enterprise content, will typically incorporate several hundred descriptive dimensions. While this may seem impressive, a vector generated by an LLM will have access to several *billion* descriptive dimensions. As with selecting a wine, richer and deeper descriptive dimensions will be much more likely to find the desired information than a few basic descriptors drawn from a local and limited corpus.

Figure 3: Vector Search



Vector search uses vector embeddings of both the query and the content to be searched to account for the meanings of terms and the relationships between concepts.

Source: Gartner
784270_C

Gartner

LLMs are powerful tools for enhancing search, but they are difficult to develop and expensive to maintain. Most enterprises do not have the capability to "bootstrap" their own, purpose-built LLM. As a result, most search platforms will integrate existing LLMs, such as GPT, BERT and BLOOM. While dramatically improving search overall, these general-use LLMs can sometimes yield unpredictable results when applied to specific domains.

The problem is that these general-purpose LLMs are trained on massive amounts of public content (from the web). They are very good at understanding language and intent, but do not always get their facts right. This was evidenced when Google debuted its BARD service. When asked "What new discoveries from the James Webb Space Telescope can I tell my 9-year-old about?", Bard responded that the telescope "took the very first pictures of a planet outside of our own solar system." This was factually incorrect. The first image of an exoplanet was actually produced almost 14 years before the Webb telescope was even launched. [4] Why did BARD make such a clear error (albeit with eloquence and confidence)? It did not have the focused, curated expertise in astronomy and space exploration history necessary to answer the question correctly.

This presents a problem. If feature engineering, based on domain expertise, does not scale and general-purpose LLMs lack the necessary expertise, how do we get the benefits of applying LLMs to search without the costly and embarrassing mistakes they may introduce? (Google's parent company, Alphabet, lost over $100 billion in market value as a result of Bard's very public mistake.) The answer is to utilize LLMs trained in the specific domain of the search application.

As an example, consider how the BERT LLM has been adapted to specific domains. The original BERT models were trained on two public, English language sources, namely the BooksCorpus (800 million words) and the English language version of Wikipedia (2.5 billion words). This LLM has been partitioned into subgraphs and infused with specific knowledge using lightweight adapters to create domain-specific LLMs for life sciences and biomedical applications (SciBERT, BioBERT, PubMedBERT), [5] and for the legal industry (LawBERT). [6]

Creating *enterprise-specific* variations of generalized LLMs still requires expertise and resources beyond the reach of many, if not most, organizations. Vendors are beginning to offer LLM development services to meet this need. For example, NVIDIA has introduced the NeMo LLM Service, based on the Megatron LLM, which claims 530 billion dimensions, to provide a "path to customizing and deploying LLMs."

The hurdles to creating purpose-built, domain-specific LLMs or customizing generalized LLMs are beginning to fall. Google researchers have recently published a method called SKILL (Structured Knowledge Infusion for Large Language Models) for training models on domain-specific information. LLMs developed with SKILL have demonstrated threefold improvement on search tasks. [7] This approach, however, requires a very specific approach to information modeling and data management, namely a knowledge graph. Fortunately, knowledge graphs have applications in enterprise search far beyond training LLMs.

Knowledge Graphs for Search

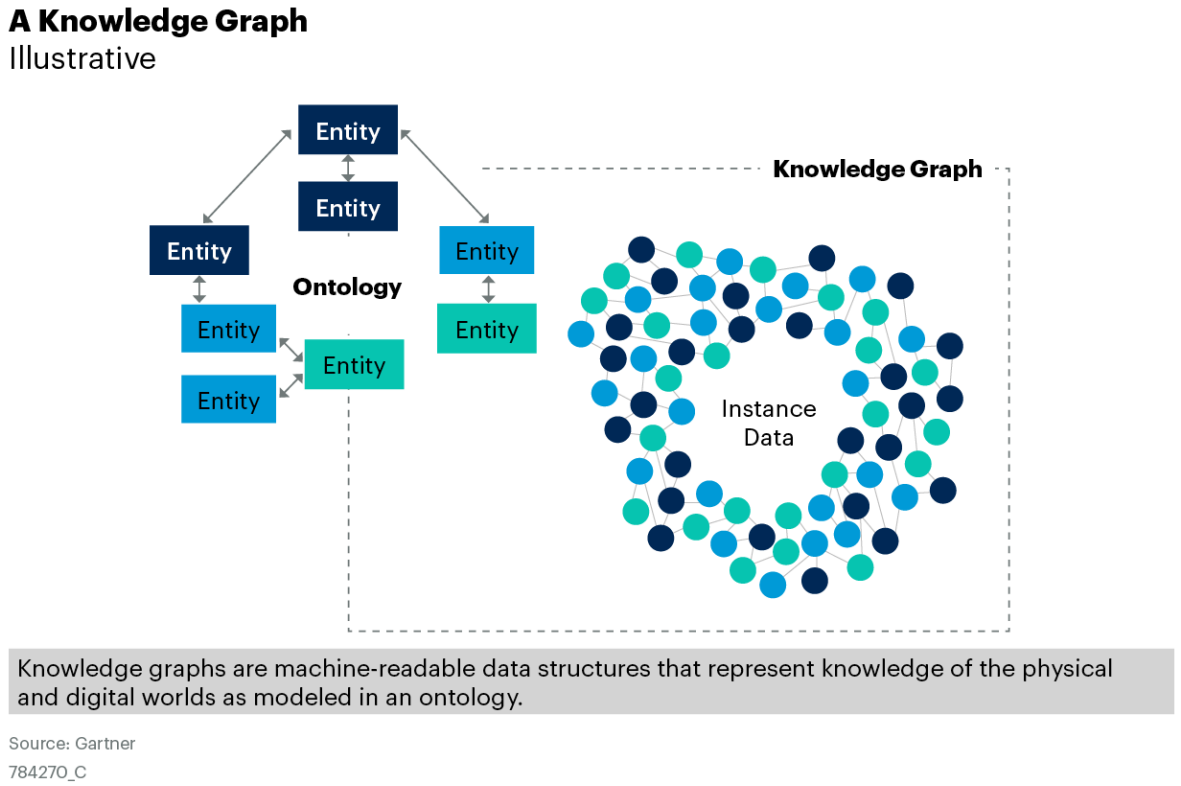
In 2012, Google made a fundamental change to its approach to search by introducing the Google Knowledge Graph, which enabled the answer-oriented search users now expect. [8] The Google Knowledge Graph is essentially an enormous knowledge base containing over 5 billion entities drawn from many sources. More importantly, it also contains relationships between those entities. It knows, for example, that Portland is a part of Oregon, which is a part of the U.S., which is a part of North America. It also relates other information to the entity called Portland, such as population, weather, identity of the current mayor, and the fact that the city is known for its breweries. It treats "Portland" as a thing, rather than just a string of characters. This idea of searching for "things not strings" is the key to effective, answer-oriented search, and knowledge graphs, used in conjunction with LLMs, are what make it possible both for Google and enterprises.

Knowledge Graphs in a Nutshell

Knowledge graphs are machine-readable data structures. They represent knowledge of the physical and digital worlds, including entities (people, organizations, digital assets and so on) and the relationships between them. These entities and relationships adhere to a graph data model, which is a network of nodes (vertices) and links (edges/arcs). The entities and relationships allowed in the knowledge graph are defined in an ontology, which is a formal model of the domain being represented.

These models consist of numerous three-part (or in some cases four-part) statements in the form "subject, predicate, object," such as "Person," "hasName," "Name." An ontology defines which entities are allowed and how they relate to each other. Ontologies are usually defined and represented using a standard such as Resource Description Framework (RDF), Terse RDF Triple Language (TURTLE), Simple Knowledge Organization System (SKOS) or Web Ontology Language (OWL). A knowledge graph is an instantiation of an ontology, using the definition to represent actual entities and their relationships. A knowledge graph consists of the instance data mapped to the ontology (see Figure 4). For example, an instance of the "Person" entity could be something along the lines of (person-462, name, "Charles Dexter Ward").

Figure 4: A Knowledge Graph



The ontology governing a particular knowledge graph provides an additional capability unavailable in other approaches to data management. Information not explicitly present in the dataset can be inferred. For example, if we know that Stockholm is the capital of Sweden and we know that Lars was born in Stockholm, we can infer that Lars is a Swedish citizen. This ability to infer facts and represent meaningful entities and relationships is extraordinarily powerful in the context of search.

Unfortunately, the power of knowledge graphs comes at the cost of complexity. Defining an ontology is a specialized skill that requires not only data modeling expertise but domain knowledge of every area of the enterprise that falls within the scope of the model being developed. Once defined, mapping and maintaining data sources can require fundamental changes to enterprise data management procedures and policies. Once the knowledge graph and supporting processes are established, much of the data extraction and mapping can be automated, but getting to that point can be a daunting undertaking.

For a full explanation of the ontology and knowledge graph development process, see Building Knowledge Graphs.

Fortunately, the search engineer often does not need to start from scratch when incorporating knowledge graphs into their search program. Many organizations have already introduced knowledge graphs into their data management and data science practices, and the rate of adoption is accelerating. Organizations such as Goldman Sachs, Novartis, Morgan Stanley, AstraZeneca and Wells Fargo are adopting knowledge graphs for a range of use cases including data integration, data discovery, identity management and security. [9] Enterprise search should be added to that list and should utilize the knowledge graphs already developed.

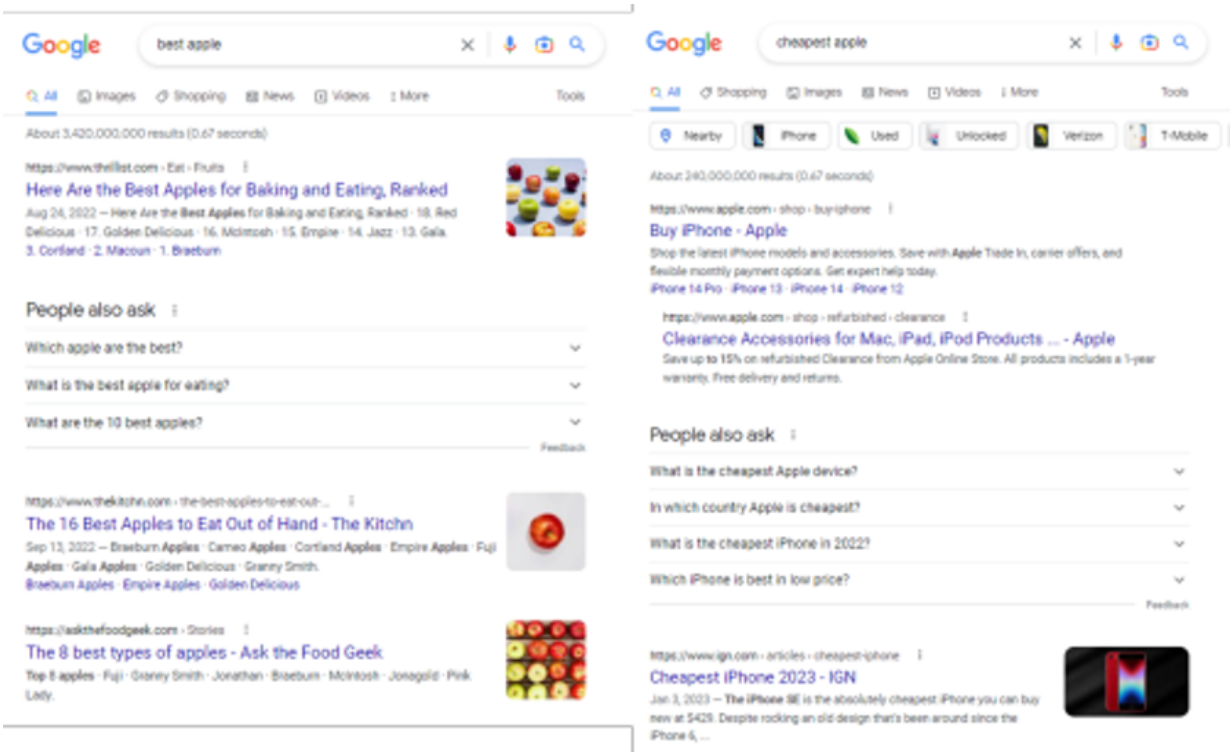
There are also many public knowledge graphs, such as Wikidata, KBpedia and DBpedia, for general world knowledge. Much of the Google Knowledge Graph is populated from these resources. In addition, there are many domain-specific ontologies that can be used to generate a variety of knowledge graphs from enterprise data. These range in subject matter from the FIBO Viewer to Chemical Entities of Biological Interest. Several directories and repositories provide access to ontologies and knowledge graphs that can be used as a starting point. Among these are BioPortal, Vocabulary of Interlinked Datasets (VoID) and Ontology Design Patterns.

Integrating either public or enterprise knowledge graphs into a search architecture, along with LLMs, makes many new search capabilities possible. Google-like natural language queries, answer panels, extractive answers, and inferred or calculated answers all come within the reach of search engineers and their customers. To deliver these capabilities successfully, the search platform must understand both the question being asked and the enterprise that is to provide the answer.

Understanding the Question

To accurately answer any question or respond to any query, a search engine must first understand what the user is after. Discerning user intent begins by interpreting the words chosen by the user to express an information need. Isolated keywords are notoriously difficult to interpret. For example, if a user searches for “best apple” or “cheapest apple,” are they looking for a nice, inexpensive piece of fruit or a cheap smartphone? Without context, it is difficult to know which interpretation better matches the user’s intent. Even Google sometimes struggles to make this determination (see Figure 5).

Figure 5: Google Search Results for “best apple” and “cheapest apple”

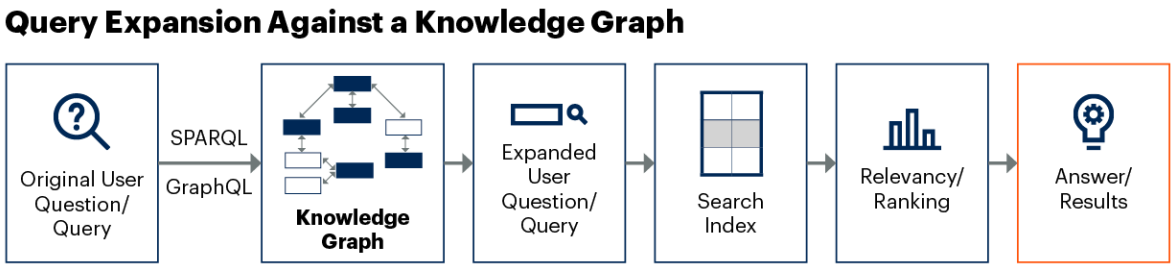


Source: Google

Users struggle to articulate their information need clearly and adequately. This is true whether they formulate that need as questions or queries (keywords and operators). Their terminology may not align with the content. They may be unaware of related and critical topics. In short, users simply “don’t know what they don’t know.” Querying a knowledge graph with the user question before querying the full search index can provide the necessary context to help fill in the gaps.

Query expansion is not a new technique. Comparing search keywords to a thesaurus and adding synonyms to the query is a common way to reconcile terminology between users and content. Early security binding adds user access privileges to the query before the index is interrogated to ensure that only content the user is entitled to see is returned. These techniques are simple, effective and easy to implement, but they still treat the search terms as isolated keywords. That is, the search terms are evaluated lexically rather than semantically. Comparing these search terms to a knowledge graph will provide the context and relationships necessary both to establish user intent and identify the information required to respond to that intent (see Figure 6).

Figure 6: Query Expansion Against a Knowledge Graph



Comparing search terms to a knowledge graph will establish user intent and identify the information required to respond to that intent.

Source: Gartner
784270_C

Gartner

For example, a simple search on the term "PTO" can return content related to "paid time off," the "Patent and Trademark Office," or the "Professional Triathletes Organization," among other potential expansions of the acronym. Normally, we simply resolve the ambiguity with an authority file that defines the acronym for the organization. A more sophisticated approach can use the proximity of other terms to disambiguate the meaning. In some cases, however, things get trickier. What if the user is a leading shoe designer for NIKE or Adidas? Perhaps she wants to take time off work to file a patent for a new triathlete shoe design she has developed on the side? In this case any or all of the three meanings of PTO could be intended. Comparing the query to a knowledge graph can help detect the user's intent.

If additional terms in the query or question are related to PTO in a knowledge graph — "trainers," "filing" or "vacation," for example — the intended meaning becomes much more apparent. This also allows us to add important related concepts to the query before it is issued, such as "noncompete clause." This enables us both to improve results when the final query is issued and include related information that the user should see, whether they are aware of it or not. If we have information on users themselves, graph-driven query expansion becomes much more powerful.

Personal knowledge graphs (PKGs) are a declared research and development priority for Google. A PKG is "a source of structured knowledge … where the entities and the relations between them are of personal, rather than general, importance. The graph has a particular "spiderweb" layout, where every node in the graph is connected to one central node: *the user*." [10]

Google must collect this information from public sources and users' interactions with its tools and services. An enterprise can take a similar approach, but with the advantage that all relevant systems are under your control and the knowledge graph can, as it were, be populated with the digital exhaust of the user's daily work activities. This capability underlies much of the graph capabilities of Microsoft 365, but is largely inaccessible to the administrator and therefore cannot be tailored to specific enterprise needs and goals. If an enterprise creates and maintains PKGs for its employees (using a platform such as Neo4j or Stardog, for example), the PKG can be designed to meet the needs of both the enterprise and the individual user.

Beyond basic permission and access control information, query expansion can enhance a search with user specific signals and information. Adding the user's role, geography, current assignment, past projects, collaborators, preferences, and past search behaviors will enrich the search at hand and can form the basis of meaningful search personalization. Those query enrichments will also guide and reinforce the machine-learning-based operation of the search platform.

PKGs are not isolated, stand-alone structures but rather subgraphs of a larger, broader knowledge graph that describes an enterprise. The entities within a PKG can even link to external entities. For example, if a developer is also a committer for an open-source project, such as Solr, that relationship should be captured explicitly, even if it is not formally an enterprise resource. This forms a basis for practical enterprise expertise discovery and management. In addition to enhancing information discovery and retrieval with presearch query expansion, the aggregate of PKGs begins to provide an overall picture of an enterprise and its capabilities.

Understanding the Enterprise

Knowledge graphs and their supporting platforms are powerful tools for describing and managing information, but they do not replace search engines. A knowledge graph can provide a semantic integration layer across data sources that reconciles content regardless of where it is originally created and currently hosted. This provides the best mechanism for understanding an enterprise and all its component domains, topics and entities. Knowledge graph platforms provide the means to query and explore this reconciled and described information. Unfortunately, directly querying a knowledge graph does not scale or perform to a level acceptable for enterprise search.

Knowledge graphs enhance enterprise search platforms and insight engines, but cannot replace them.

When querying a knowledge graph to enrich or rewrite a query, the search is focused on the structure of the knowledge graph and how it can be exploited to enhance the actual search operation. This narrow scope makes directly querying the knowledge graph practical and the performance acceptable. When the actual search is executed, the full-instance graph must be considered, which increases search time to a point that is unacceptable to users. To gain the benefits of a knowledge graph in search, while still satisfying impatient information seekers, a proper search index is still necessary. A knowledge graph can be used to enrich the content being indexed by guiding natural language processing and text analytics to generate and add metadata as part of the ingestion process. In addition, the knowledge graph itself — both entities and relationships — should be indexed.

Content Processing and Index Enrichment

Creating a modern search index involves much more than simply parsing documents into individual terms and logging term occurrences. Any additional information that can facilitate the tasks required of the search engine and application should also have a place in the index. Most commonly, a document is parsed into index fields indicating how a particular term is used. This metadata explicitly denotes that a term or phrase represents a person, organization, product, project or any other useful concept that can be captured. This is a common application of text analytics in the indexing pipeline, but it remains challenging.

Consider the sentence “Actor Eugene Levy flew from Paris to Portland on Alaska Airlines flight 2022 in October.” First, the parser must detect words that might represent specific concepts or entities of interest. These could include, “Eugene,” “Levy,” “Paris,” “Portland,” “Alaska” and “flight.” It must then determine how these terms are being used and what entities or concepts they may each represent.

Three of these terms, “Eugene,” “Paris” and “Portland,” could represent specific cities. The nearby term “Alaska” might support this, as it is also the name of a geographic entity. However, the terms “Actor” and “Levy,” adjacent to “Eugene,” might also indicate that “Eugene” is the first name of a performer with the last name “Levy.” The presence of an actor might point to “Paris” being part of the name of another formerly famous person, “Paris Hilton.” Then again, “Eugene” and “Portland” are both cities in Oregon, so maybe they are locations after all. “Alaska” is part of the U.S., like “Portland” and “Eugene,” so it probably means the northernmost state, but then again, it is next to “Airlines,” so maybe it is actually the name of a business. “2022” looks like part of a date, but why is the word “in” separating the year from the month? All of this must be figured out before content is added to a search index.

Extracting meaningful, indexable information from text is a complicated and messy task. There are many analytical tricks that can tease out much of the meaning in this sentence and parse its terms into appropriate search index fields. Most of them depend on statistical analysis, but increasingly they also use machine learning to suss out semantics and slot the results into index fields. Results are often less than optimal, however. Knowledge graphs and LLMs can accelerate and increase the accuracy of this process.

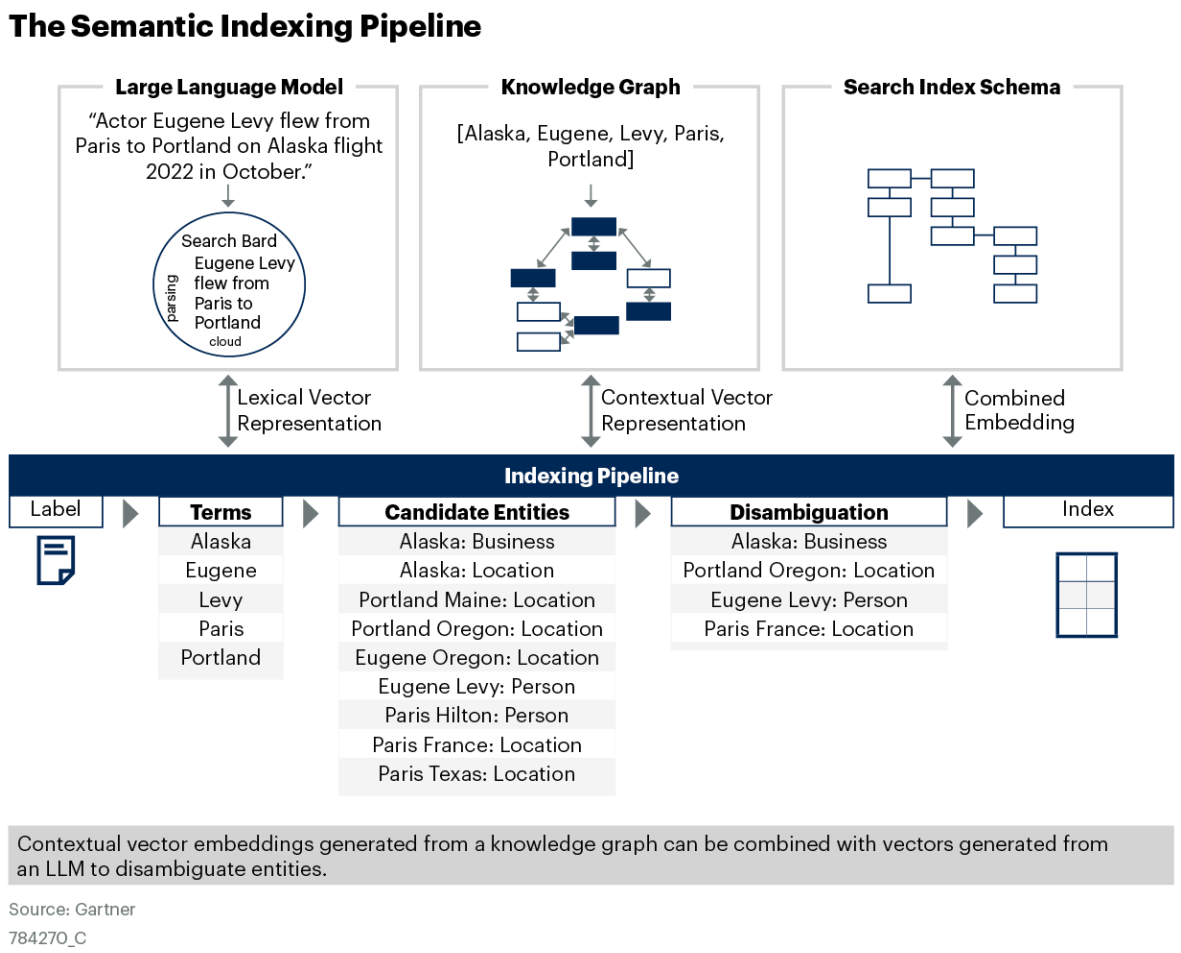
> **By 2024, companies that use graphs and semantic approaches for natural language technology projects will have 75% less artificial intelligence technical debt than those that do not.**
>
> *— How to Build Knowledge Graphs That Enable AI-Driven Enterprise Applications*

The first step in the semantic indexing processes is to identify and extract entities that may be of interest. These represent people, places, organizations, products or any other *thing*. This initial extraction step is a standard function of text analytics, which actually produces candidate entities rather than final, definitive entities. This is because there will still be ambiguity as to what type of entity a term represents. “Paris” could be a person or a city. “Alaska” could be a state or a business.

A publicly trained LLM, such as GPT or BERT, can be used to generate vector embeddings and differentiate common entities. A domain-specific LLM could differentiate specialized terms for a particular field such as medicine, law or finance. However, even these specialized LLMs have limitations and are only as rich as the information available in the information used to train them. For example, the Unified Medical Language System (UMLS), a primary vocabulary used to train BioBERT, has relatively sparse relationship information. A majority of the entities in the UMLS are represented by just 10 semantic types, and over 90% of the entities do not have a description. [11] This severely limits the LLM's ability to support fine-grained disambiguation.

To overcome these limitations, contextual vector embeddings for candidate entities should be generated from a knowledge graph, taking advantage of its relationships and structural information. These vectors can be combined with the lexical embeddings generated from the LLM to produce a disambiguated entity that can be indexed and even added to the knowledge graph, along with any other facts extracted from the source content (see Figure 7). As an added bonus, these combined vectors can also be used to train additional machine learning models, including those used internally by the search engine. Platforms from GraphAware, Ontotext and Pureinsights have each been used to develop search architectures supporting this type of entity disambiguation cycle.

Figure 7: The Semantic Indexing Pipeline



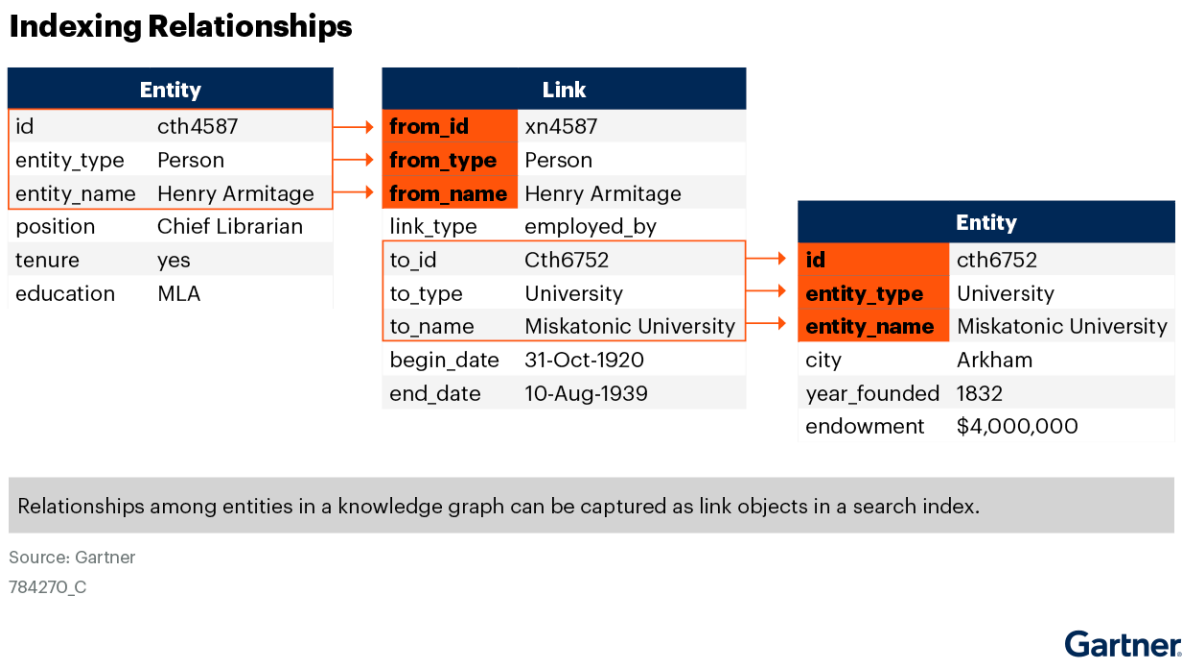
Contextual vector embeddings generated from a knowledge graph can be combined with vectors generated from an LLM to disambiguate entities.

Source: Gartner
784270_C

Gartner

In addition to the source content to be searched, the knowledge graph itself should be indexed. The domain-specific entities, along with the contextual and structural information captured in the graph, are valuable resources for search. As previously discussed, directly querying the graph is not sufficiently performant to use for general search at enterprise scale. As a result, graph entities and their associated properties should become entries in the search index. Of particular value are the relationships between entities, as captured in the graph. These should be indexed. By including relationship properties as part of the index entry, many queries can be answered directly by querying only these link objects. For example, if you want to know how long a particular person has worked for a particular university, you can identify this information from just the relationship or link object between the person and the university. This is very efficient, but it only works if the necessary information is captured and properly structured (see Figure 8).

Figure 8: Indexing Relationships



Finally, useful information that is not explicitly present in the source content but can be inferred from the graph and its governing ontology should be generated and indexed. Again, inferences can be made directly from the graph, but they involve costly operations and are unsuitable for query-time operations. Preinferring facts and adding them to the index avoids this problem. Having inferred facts available at query time is particularly helpful for answering questions that rely on multiple relationships. It is easy to get carried away with inference, however. If the process is not limited to only those inferred facts that are directly relevant to the search application being supported, the index can quickly become bloated, which damages both performance and accuracy.

Strengths

The introduction of LLMs and knowledge graphs into a search architecture offers many strengths and advantages. These include:

- **Google-like search experience:** Traditional approaches to search are incapable of providing the type of search experience that users have come to expect. LLMs and knowledge graphs enable the capabilities that will provide the desired search experience. Natural language questions replace formulaic queries, and direct answers replace lists of links. Knowledge panels can be generated from graph relationships that not only provide extended information about the subject of the question, but also related and recommended topics.

- **Improved search accuracy:** LLMs and knowledge graphs enable semantic vector search, yielding superior search results to those available from purely lexical search. In addition, vector embeddings generated from LLMs are contextual, rather than context-independent like those generated by simpler models such as word2vec. This context improves disambiguation of entities significantly. These vectors are further refined and enriched by graph information, resulting in both more precise queries and a richer index. Together, this results in better search results.
- **Enhanced data management:** Knowledge graphs are an increasingly common mechanism in enterprise data management. Search engineers can utilize existing knowledge graphs and their supporting data management cycles to enhance their search products. As LLMs are incorporated into the index pipeline, the output of that enrichment process can be fed back into enterprise knowledge graphs or form the basis of altogether new graphs. This creates a virtuous cycle for both search and data management, with each improving the other.

Weaknesses

The process of enhancing search with LLMs and knowledge graphs is not without its challenges, which include:

- **Complexity:** Enterprise search is an inherently complex undertaking. Various repositories, each with its own content types, vocabularies and idiosyncrasies, must be identified, reconciled and indexed. While integrating an LLM can help in the reconciliation and disambiguation process, they add an additional layer of integration, processing and dependency. Add a knowledge graph and the complexity grows significantly, especially if the output of the content processing and indexing process is to be fed back into the data management life cycle. Proper governance and close coordination between all parties concerned is essential to maintain the integrity of the data and search environments.

- **Newness:** Although LLMs and knowledge graphs are not themselves new, their application to the enterprise environment is a fairly recent development. Many organizations have limited experience with these technologies in general and no experience of their application to search. Introducing these technologies involves a significant learning curve and may require more resources than are readily available. In many cases, engaging with a vendor or professional services organization is necessary to bootstrap a knowledge graph program or an LLM integration. This is especially the case if the enterprise does not have a mature search practice. In addition, the LLM space is evolving rapidly. The size and complexity of available models increases roughly tenfold annually. [12] Many of the ramifications of this rapid growth have yet to be fully identified or understood.

- **Bias and inaccuracy:** LLMs are primarily trained on public information. These internet and web resources have the potential to introduce both bias and false information into the model. This can have a detrimental impact on search results and has the potential to perpetuate and even spread misinformation. Organizations will need to monitor for this and take corrective steps if bias and inaccuracies are detected. This places an additional burden on enterprise search and security teams, but it will be unavoidable as machine-learning-based technologies become increasingly common and critical to operations.

Guidance

To improve the usability and effectiveness of enterprise search by integrating LLMs and knowledge graphs into their search environments, application technical professionals should:

- **Generate vector embeddings from a general LLM:** Use vector embeddings from a general LLM to enhance your existing search engine. Such integrations are possible with most modern search platforms as part of the indexing pipeline. Several vendors have these connectors and processes established and available as part of their product offerings. This is the most direct path to semantic vector search, and can both improve search results and provide experience with LLMs. While GPT and LaMDA are currently receiving the most attention, earlier models such as BERT and its variants have a broader body of prior art and a larger community of practice to draw from. If this is the search team's first use of LLMs, adopting and utilizing a more established LLM will typically be easier and faster simply because there are more implementation examples to draw from.

- **Capitalize on existing enterprise knowledge graphs:** Knowledge graphs are complicated and expensive undertakings. It may be difficult to justify instituting a knowledge graph practice to exclusively support search. If knowledge graphs are already a feature of your organization's data management practice, identify which existing graph is most pertinent to your search use case and integrate that graph with your search platform. Create a virtuous cycle by feeding the product of your content processing and indexing pipeline back into the source knowledge graph and data management life cycle. If knowledge graphs are not currently a part of enterprise data management, but the organization is curious about them, utilizing an existing public knowledge graph to enhance search could be an attractive proof of concept.

- **Engage a vendor or experience integrator:** LLM and knowledge graph support of semantic vector search is currently the state of the art. Each model, graph and search platform has its own idiosyncrasies in how it behaves and interacts. Getting the full benefit of this approach will require deep experience with each element involved. Most enterprises do not have this expertise in each area, and much less in the interaction of all three. Most of the search vendors and integrators mentioned in this document have prioritized semantic vector search and its supporting architectures as part of their platform offerings. Engaging their professional services to bootstrap the project will be more cost-effective than going it alone. This is especially true if knowledge transfer is built into the engagement. Have the search team work closely with the vendor to understand the dependencies and interactions. This will both modernize your search practice and establish a sustainable foundation for future innovation.

Notes

[1] P. Nayak, Understanding Searches Better Than Ever Before, The Keyword, 25 October 2019.

[2] S. Pichai, An Important Next Step on Our AI Journey, The Keyword, 6 February 2023.

[3] Y. Mehdi, Reinventing Search With a New AI-powered Microsoft Bing and Edge, Your Copilot for the Web, Microsoft, 7 February 2023.

[4] 2M1207 b - First Image of an Exoplanet, NASA.

[5] Z. Meng and others, Mixture-of-Partitions: Infusing Large Biomedical Knowledge Graphs Into BERT, *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, 2021, pp. 4672-81.

[6] E Zhang, LawBERT: Towards a Legal Domain-Specific BERT?, Towards Data Science, 21 August 2020.

[7] F. Moiseev and others, SKILL: Structured Knowledge Infusion for Large Language Models, *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies* (2022), pp. 1581-88.

[8] A. Singhal, Introducing the Knowledge Graph: Things, Not Strings, The Keyword, 16 May 2022.

[9] J. L. Schenker, New Report Details Industry's Use of Knowledge Graphs, Medium, 7 May 2021.

[10] K. Balog and T. Kenter, Personal Knowledge Graphs: A Research Agenda, *Proceedings of the ACM SIGIR International Conference on the Theory of Information Retrieval (ICTIR)*, ACM (2019), pp. 217-20.

[11] M. Varma and others, Cross-Domain Data Integration for Named Entity Disambiguation in Biomedical Text, arXiv, 2021.

[12] J. Simon, Large Language Models: A New Moore's Law?, Hugging Face, 26 October 2021.

Recommended by the Author

Some documents may not be available as part of your current Gartner subscription.

Building Knowledge Graphs

How to Build Knowledge Graphs That Enable AI-Driven Enterprise Applications

Emerging Use Cases for Natural Language Technology